Swarm Robots Foraging under Vision and Communication Uncertainties

Simon Obute Obute

University of Leeds
School of Computing

Submitted in accordance with the requirements for the degree of
Doctor of Philosophy

College of Engineering August 2020

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

The work in Chapter 3 and 4 of the thesis has appeared in a publication as follows:
Obute, S.O., Kilby, P., Dogar, M.R., Boyle J. H. (Accepted: 2020) RepAtt: Achieving Swarm Coordination through Chemotaxis. In: 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE 2020).
I was responsible for conducting all of the research (except for Adaptive Large Neighbourhood Search algorithm), performing the data analysis and writing the paper.
Kilby, P. was responsible for generating the foraging paths from his implementation of the Adaptive Large Neighbourhood Search algorithm and describing its implementation.
Dogar, M.R. and Boyle J.H. supervised the research and reviewed the paper before submission

The work in Chapter 6 of the thesis has appeared in a publication as follows:
Obute, S.O., Dogar, M.R., Boyle, J.H. (2019). Chemotaxis Based Virtual Fence for Swarm Robots in Unbounded Environments. In: Martinez-Hernandez U. et al. (eds) Biomimetic and Biohybrid Systems. Living Machines 2019. Lecture Notes in Computer Science, vol 11556. Springer, Cham.
I was responsible for conducting all of the research, performing the data analysis and writing the paper.
Dogar, M.R. and Boyle J.H. supervised the research and reviewed the paper before submission

Acknowledgements

I thank God for the grace to conduct this doctoral research. My gratitude also goes to my supervisors, Dr. Jordan Boyle and Dr. Mehmet Dogar, for their guidance throughout the research. They were instrumental in helping me focus on the research goal and not getting dragged into the many ‘rabbit holes’ on the way. My coming to Leeds was made possible by three years of sponsorship from the National Information Technology Development Agency, Nigeria. I am grateful to the agency for covering my tuition and living allowance expenses for the first three years of the research. To my wife, Rita, I am grateful for all the support, encouragement and endless prayers for me to finish well in the PhD research. Thanks to the School of Computing Robotics Lab, Kenny, Felix, Seun, Opeyemi, Peter, Ezekiel and Ephraim for the friendship and engaging discussions we had. You have made my PhD journey an enjoyable one. A special thanks to Nick for helping out during the litter recording experiments.

Abstract

The foraging of items and returning them to a specific deposit site is a canonical task for investigating the collective behaviour of a swarm of robots. In addition to being a suitable task for investigating robot capabilities such as vision, inter-robot communication and task allocation, it has many real-world applications in search and rescue, planetary exploration, hazardous waste clean up, agriculture and litter picking that make it attractive for swarm robots deployment. The state-of-the-art in swarm robotics research has done little to study in detail the impact that imperfections in robot vision and communication have on the collective behaviour of a swarm of robots. This research changes that by extensively studying the impact of such imperfections and proposing a novel foraging algorithm that is robust to them.

The Repulsion-Attraction (RepAtt) algorithm proposed in this thesis takes inspiration from the chemotaxis search behaviour observed in biological organisms. The design approach for RepAtt emphasises simplicity of the algorithm to demonstrate the effectiveness of this minimalist bio-inspired chemotaxis-based search, reduce the hardware requirements of robots and remove complexities in inter-robot communication. RepAtt utilises temporal gradients of attractant and/or repellent signals to increase the probability of tumbles when conditions are unfavourable (increasing repulsion or decreasing attraction intensities), or suppressing tumbles as conditions progressively improve. Hardware experiments using sound were conducted to model the signals the robots used for their chemotaxis search and also to quantify a realistic noise level in the signal. Using this communication model and noise levels, extensive simulation studies were conducted for the swarm under ten different foraging environments. The results indicate that with realistic noisy communication, the RepAtt algorithm significantly improved the swarm's collective behaviour by reducing the foraging time by up to 70% in comparison to the Random Walk algorithm. The RepAtt algorithm also scaled well with increasing swarm size and was robust to changes in the distribution of targets and world sizes.

An elaborate study was also conducted to model the swarm robots' vision system using state-of-the-art deep neural networks object detection algorithms. The outcome of this aspect of the research was a probabilistic vision model that is representative of the performance of object detection algorithms and can easily be applied to a wider variety of swarm tasks that involve target detection. This probabilistic vision model was studied using extensive simulations on Random Walk and RepAtt swarm foraging algorithms. The results showed that, despite the negative influence of imperfect vision system, RepAtt still exhibited superior foraging performance in comparison to Random Walk even for object detection probabilities as low as 0.2. At fast vision update rates, the negative impact of the imperfect vision on the swarms' collective behaviour when using the RepAtt algorithm was most significant. This was because increasing the vision update rate increased the rate robots switched between broadcasting repulsion and attraction signals. The level of this impact increased as the robots' detection accuracy approached 0.5 and was most noticeable for the high vision update rate of 40 Hz. The results showed that minimising these fluctuations between repulsion and attraction broadcasting improved the swarm's robustness to the imperfections in the robots' vision system, and consequently reduced the foraging time.

The chemotaxis-based algorithm was also extended to implement a virtual fencing mechanism to prevent a swarm of exploring robots from drifting away from their work area near a stationary or mobile nest. This aspect of the research was important for handling swarm deployment in the numerous unbounded environments they encounter in real-world applications, which is a less studied area in swarm robotics. The results of extensive simulation and hardware validation showed that the virtual fence was a practical means of keeping the swarm within the desired work area and was as effective as a physical wall, which is only applicable in controlled environments.

Table of Contents

List of Figures **xi**

List of Tables **xxiii**

1 Introduction **1**

1.1 Background 1

1.2 Motivation 3

1.3 Aim and Objectives 5

1.4 Contributions Made 5

1.5 Thesis Outline 6

2 Literature Review **7**

2.1 Introduction 7

2.2 Recruitment Strategies in Swarms 7

2.2.1 Recruitment in ants 7

2.2.2 Recruitment in bees 9

2.3 Biological Search Strategies 10

2.4 Swarm Robots Foraging Algorithms 12

2.5 Recruitment in Swarm Robots 19

2.5.1 Stigmergy-based communication 19

2.5.2 Direct communication 21

2.5.3 Impact of noise on recruitment 21

2.6 Detecting Target Objects 22

2.6.1 Individual target detection 23

2.6.2 Deep neural network for object detection 23

2.6.3 Deploying deep neural network based object detectors on robots . . 26

2.7 Robots for Picking Litter 27

2.8 Swarm Foraging Metrics 29

2.8.1 Metrics measurement 29

2.8.2 Baseline algorithms 30

2.9 Gaps in Body of Knowledge 31

2.10 Summary 33

3 The RepAtt Swarm Foraging Algorithm **35**
3.1 Introduction 35
3.2 The Robot Model 35
3.3 The Communication Model 36
3.4 Algorithm Description 39
3.5 Chemotaxis Parameters 41
3.6 Swarm Foraging Simulation Setup 44
3.7 Parameter Optimisation 45
3.8 Swarm Foraging Time Analysis 53
3.9 Robustness of RepAtt 56
3.10 Scalability of RepAtt 57
3.11 Summary 59

4 Noisy Communication **61**
4.1 Introduction 61
4.2 Noise and Filter Modelling 61
4.3 Parameter Optimisation 65
4.4 Foraging with Noisy Communication 71
4.5 Robustness of RepAtt 75
4.6 Scalability of RepAtt 76
4.7 Summary 76

5 Uncertainty in Foraging Robot Vision **79**
5.1 Introduction 79
5.2 Data Collection 79
5.3 Dataset Preparation 81
5.4 Training and Testing 82
5.4.1 Testing phase 1: preparing YOLOv3 as baseline 83
5.4.2 Testing phase 2: extracting metrics data for smaller networks 85
5.5 Model Development 88
5.6 Model Validation 92
5.7 Simulation using Network Models 93
5.7.1 Full frame rate model 94
5.7.2 Raspberry Pi 4 model 96
5.7.3 Raspberry Pi 3 model 97
5.7.4 Effect of vision update rate on swarm foraging 98
5.8 Exploring the Effects of Model Probabilities 99

5.8.1 Effects of P_{u2s} and P_{s2s} on P_s 99
5.8.2 Effects of P_{u2s}, P_{s2s} and vision update rate on RepAtt 100
5.8.3 Effects of P_{u2s}, P_{s2s} and vision update rate on swarm foraging . . . 105
5.9 Summary 107

6 Chemotaxis-Based Virtual Fence 109
6.1 Introduction 109
6.2 Exploration in Unbounded Environments 109
6.3 Simulation Setup 112
6.4 Determining the Chemotactic Region 112
6.5 Effects of Probability Multiplier and Divisor 113
6.6 Investigating Exploration Effectiveness 117
6.7 Real Robot Experiments 122
6.8 Summary 125

7 Conclusions and Future Directions 127
7.1 Summary 127
7.2 Conclusion 130
7.3 Further Work 131

References 133

Appendix A RepAtt: Noiseless Communication 147
A.1 Parameters Optimisation 147
A.2 Scalability of RepAtt 161

Appendix B Noisy Communication 165
B.1 Parameters Optimisation (N100-Q1) 165
B.2 Parameters Optimisation (N100-Q40) 169
B.3 Scalability of RepAtt 183

Appendix C Imperfect Vision 187
C.1 False Positive Data 187
C.2 Effects of P_{u2s}, P_{s2s} and Vision Update Rate on Swarm Foraging 187
C.2.1 Algorithms foraging performance 188
C.2.2 Effects of update rates 193

List of Figures

1.1 A test model representation of a group of swarm robots tasked with collecting litter in a residential area. 4

2.1 Three types of random walk. Blue and green circles represent starting and stopping locations respectively, while the dotted lines represent the path of the agent. In (c), the brown gradient represents change in concentration of an attractant signal the agent measures to inform its biased search toward the signal source. 11

3.1 An experimental test model of white noise sound signal intensity degradation with distance. 37
3.2 Sound intensity output from two white noise sources. 38
3.3 One-dimensional Environment 42
3.4 One-dimensional robot simulation using chemotaxis to move toward attractive signal. 43
3.5 Plot of initial target distributions, nest and robot locations for 10 test environments. (a) - (e) and (f) - (j) are 50m × 50m and 100m × 100m world boundary dimensions respectively. The 200 targets to forage are purple, black '+' is central nest and yellow blob represent the robots. 45
3.6 Sorted foraging times normalised using the mean time for A1m1d-R1m1d for the corresponding world setup. 46
3.7 Effect of the chemotaxis parameters on the foraging performance of a swarm of 36 robots foraging 180 out of 200 targets in the One100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 2$, $r_d = 1$. 48
3.8 Enhanced view of the heat map highlighting the location of the best and worst parameter combinations in the One100m distribution. The annotations show the normalised foraging time with the corresponding ranking in brackets. 49
3.9 Effect of the chemotaxis parameters on the foraging performance of a swarm of 36 robots foraging 180 out of 200 targets in the Uniform50m targets distribution. Best combination for this distribution is $a_m = 1$, $a_d = 1$, $r_m = 1$, $r_d = 1000$. 50
3.10 Enhanced view of the heat map where the best and worst parameter combinations occur in the Uniform50m distribution. 51

3.11 Effect of the chemotaxis parameters on the foraging performance of a swarm of 36 robots foraging 180 out of 200 targets in the ten environment setups. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 1$, $r_d = 100$. 52
3.12 Enhanced view of the heat map where the best and worst parameter combinations occur when sorted according to rank scores across the 10 environment setups. 53
3.13 Scatter plot of simulation progress of a swarm of 36 robots foraging 200 targets in the One100m world setup. 54
3.14 Scatter plot of simulation progress of a swarm of 36 robots foraging 200 targets in the Uniform50m world setup. 55
3.15 Time taken by swarm of 36 robots to pick up 90% of targets for different world setups, normalised using the time taken by Random Walk. 56
3.16 Change in swarm foraging times across the ten target distributions. Foraging time was normalised based on distribution with shortest time, which was for the Uniform50m world, for the corresponding algorithms. 57
3.17 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 58

4.1 (a) Line fitted to every one metre segment of the experimental data. (b) Difference between sound measurements and corresponding line in the segment. (c) Standard deviation of the sound variations from the fitted line. (d) Ratio of the deviation of each segment to mean of sound data for the corresponding segment is fairly consistent. 62
4.2 The modelled noise data fits strongly with the level of noise expected from experiment data. 63
4.3 Queue size variation of the average filter. Increasing size of queue reduces number of negative gradient when traversing from 15m to 1m 64
4.4 Queue size versus percentage increase/change of the experimental (raw) and modelled data. Error bars represent standard deviation across the five experiments. 65
4.5 Sorted foraging times for 900 combinations of chemotaxis parameters normalised using the mean time for Random Walk for the corresponding world setup. The swarm of 36 robot foraged used noisy communication without any noise filtering system (i.e. N100-Q1) 66

4.6 Effect of the chemotaxis parameters on the performance of a swarm of 36 robots foraging 180 out of 200 targets in the One100m and Uniform100m targets distributions and using N100-Q1 version of RepAtt. Best combination from the scoring system is $a_m = 1$, $a_d = 1000$, $r_m = 1$, $r_d = 1000$. 67
4.7 Enhanced view to heat map where the best (a) and worst (b) parameter combinations occur when sorted according to rank scores across the 10 environment setups. 68
4.8 Sorted foraging times for 900 chemotaxis parameter combinations normalised using the mean time for Random Walk for the corresponding world setup. 68
4.9 Effect of the chemotaxis parameters on the foraging performance of a swarm of 36 robots foraging 180 out of 200 targets in the ten test worlds. Best combination for this distribution is $a_m = 4$, $a_d = 100$, $r_m = 1$, $r_d = 10$. . . . 70
4.10 Heat map where the best and worst parameter combinations occur when sorted according to rank scores across the ten environment setups. 71
4.11 Scatter plots for One100m simulation snapshots. The optimized chemotaxis parameters for N100-Q40 $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ for all the simulations in this table. 72
4.12 Scatter plots for Uniform50m simulation snapshots. The optimised chemotaxis parameters for N100-Q40 $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ for all the simulations in this figure. 73
4.13 Varying queue size and its effect on foraging time. The optimised chemotaxis parameters combination of $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ was used for all simulations. Each bar shows the mean of 30 simulation repetitions and error bars represent 95% confidence interval. 74
4.14 Change in swarm foraging times across the ten target distributions. Each simulation was repeated 30 times, with y-axis showing mean time to pick up 90% of 200 targets (normalised based on distribution with shortest mean time for each algorithm, which is Uniform50m in all cases). 76
4.15 Relative efficiency computed based on $p = 180$ out of 200 total targets (using Equation 3.6). Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ were used. 77

5.1 Development workflow for the robot vision model and the investigation of the effects of imperfect vision on a swarm of foraging robots. 80
5.2 Robot setup for collecting data of litter objects in St. George's Park. 81

5.3 Sample images of images used as training dataset with litter objects annotated using red bounding boxes. The first two rows are images downloaded from ImageNET [1] and the third row are sample images from litter recording in St. George's Park. 82
5.4 Detecting litter using YOLOv3 and applying five-metre horizon to it. 83
5.5 Sample image showing the different tags a litter can have in a frame. Pink box represents the new litter with ID L169, blue boxes are for litters that were tracked using IOU, while red boxed litter is tracked using optical flow. 84
5.6 The path of each litter detected by YOLOv3 within the videos. Black spots are first appearance of litter, red are positions tracked using optical flow, blue are positions tracked using IOU and green are last location of litter within the frame. 85
5.7 A sample detection using two sizes of Tiny-YOLOv3 ((a) and (b)) and MobileNet-SSD ((c) and (d)). The larger input sizes are more successful in detecting litter, however, they usually have longer inference time because of the increased computational requirement. 87
5.8 The data from Tiny-YOLOv3 and MobileNet-SSD are compared with corresponding data from YOLOv3. All detections that match (shown in blue) are True Positives; False Negative detections are shown in pink; while the red box represents False Positive detections. 88
5.9 Representation of the matching accuracy of detections in the smaller networks against those of YOLOv3. Red spots are missed detections (FN) while green spots are matching detections (TP). The data was generated from the positions of all 679 unique litter IDs across all video frames used for testing the networks. 89
5.10 The swarm foraging performance and corresponding P_s shown in brackets. Foraging times were normalised by the performance of Random Walk algorithm, where robots used pure vision to forage. Full frame rate model probabilities were used and columns were colour-coded based on Random Walk, N0-Q1, N100-Q40 algorithm types. 95
5.11 The swarm foraging performance and corresponding P_s shown in brackets. Foraging times were normalised by the performance of Random Walk algorithm, where robots used pure vision to forage. Raspberry Pi 4 update rate model probabilities were used and columns were colour-coded based on Random Walk, N0-Q1, N100-Q40 algorithm types. 96

5.12 The swarm foraging performance and corresponding P_s shown in brackets. Foraging times were normalised by the performance of Random Walk algorithm, where robots used pure vision to forage. Raspberry Pi 3 vision update rate model probabilities were used and columns were colour-coded based on Random Walk, N0-Q1, N100-Q40 algorithm types. 98
5.13 Monte Carlo simulation for 100,000 time steps showing the mean P_s for various combinations of P_{u2s} and P_{s2s}. 100
5.14 The average proportion of forward movements by a one-dimensional robot as P_s increased. The P_{u2s} and P_{s2s} probabilities were applied at either 1 Hz, 4 Hz or 40 Hz. The robot used N0-Q1 with chemotaxis parameters $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ RepAtt setup to respond to the signal source. 101
5.15 Percentage of forward movement with varying P_{s2s} and P_{u2s} probabilities at 40Hz vision update rate. Colour gradients show the variation in model probabilities. 102
5.16 The transitions of the signal source when it is updated at 1Hz, 4Hz or 40Hz. The data indicates that for combinations of P_{u2s} and P_{s2s} that result in a particular P_s, the variability of the transitions increases with increase in vision update frequency. 103
5.17 Mean normalised time to pick up 90% of 200 targets by a swarm of 36 robots (y-axis) and P_s on the x-axis. Foraging times were normalised using the time taken by Random Walk with $P_{u2s} = P_{s2s} = 1.0$ for the corresponding world and vision update rate setups. 104
5.18 Mean normalised time to pick up 90% of 200 targets by a swarm of 36 robots (y-axis) and P_s on the x-axis. Foraging times were normalised using the time taken by the 40 Hz vision update rates version of the algorithms with perfect vision ($P_{u2s} = P_{s2s} = 1.0$) for the corresponding world and vision update rate setups. 106

6.1 In (a), the nest (N) broadcasts a homing signal that degrades with distance (green gradient), which mobile robots (R) listen to. Robots within the working area ($< d_c$) perform random walk with constant tumble probability P_t, while robots within the chemotaxis region ($\geq d_c$) use chemotactic search to return to the working area. White circles represent robots in working area, while black circles represent robots in chemotaxis region. During nest exploration, the nest moves in vertical (b) and horizontal (c) pattern in the search area. 110

6.2 Effects of nest relative velocity, v_n, probability multiplier, a_m, and divisor, a_d, on the average number of robots within a specific distance, d_r from the nest/guide robot. Chemotaxis activation distance was 10m for all simulations. The background colours range from dark to light, indicating low to high values respectively. 114
6.3 Effects of nest relative velocity, v_n, tumble probability multiplier, a_m, and divisor, a_d, on the average number of robots within a specific distance, d_r ranges from the nest/guide robot. 116
6.4 (a) Omnidirectional speaker setup (b) Switching from directional to omnidirectional speaker resulted in general reduction in the sound intensity sensed by the mobile robot. 123
6.5 Robots used for the experiment and photos of exploration robot following the moving nest 123
6.6 Real robot validation of chemotaxis behaviour to remain close to a nest. Experiments were repeated 5 times, and error bars represent 95% confidence interval. 124

A.1 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 147
A.2 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 148
A.3 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 148
A.4 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 149

A.5 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 149
A.6 Effect of chemotaxis parameters on foraging performance in the One50m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 1000$, $r_m = 1$, $r_d = 1$. 150
A.7 Effect of chemotaxis parameters on foraging performance in the One100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 2$, $r_d = 1$. 151
A.8 Effect of chemotaxis parameters on foraging performance in the Two50m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 4$, $r_d = 50$. 152
A.9 Effect of chemotaxis parameters on foraging performance in the Two100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 2$, $r_d = 10$. 153
A.10 Effect of chemotaxis parameters on foraging performance in the Four50m targets distribution. Best combination for this distribution is $a_m = 4$, $a_d = 50$, $r_m = 6$, $r_d = 1000$. 154
A.11 Effect of chemotaxis parameters on foraging performance of swarm in the Four100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 1000$, $r_m = 1$, $r_d = 10$. 155
A.12 Effect of chemotaxis parameters on foraging performance of swarm in the Half50m targets distribution. Best combination for this distribution is $a_m = 4$, $a_d = 100$, $r_m = 4$, $r_d = 10$. 156
A.13 Effect of chemotaxis parameters on foraging performance of swarm in the Half100m targets distribution. Best combination for this distribution is $a_m = 8$, $a_d = 50$, $r_m = 4$, $r_d = 10$. 157
A.14 Effect of chemotaxis parameters on foraging performance of swarm in the Uniform50m targets distribution. Best combination for this distribution is $a_m = 1$, $a_d = 1$, $r_m = 1$, $r_d = 1000$. 158
A.15 Effect of chemotaxis parameters on foraging performance of swarm in the Uniform100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 4$, $r_d = 100$. 159

A.16 Effect of chemotaxis parameters on foraging performance of swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 1$, $r_d = 100$. 160
A.17 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 161
A.18 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 161
A.19 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 162
A.20 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 162
A.21 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 163

B.1 Sorted foraging times normalized using the mean time for Random Walk for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. The robots foraged using noisy communication without using any noise filtering system (i.e. N100-Q1) 165
B.2 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution and using N100-Q1 version of RepAtt. Best combination from the scoring system is $a_m = 1$, $a_d = 1000$, $r_m = 1$, $r_d = 1000$. 166

B.3 Effect of the chemotaxis parameters on the foraging performance of a swarm in the Uniform100m targets distribution and using N100-Q1 version of RepAtt. Best combination from the scoring system is $a_m = 1$, $a_d = 1000$, $r_m = 1$, $r_d = 1000$. 167
B.4 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m and Uniform100m targets distributions and using N100-Q1 version of RepAtt. Best combination from the scoring system is $a_m = 1$, $a_d = 1000$, $r_m = 1$, $r_d = 1000$. 168
B.5 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 169
B.6 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 169
B.7 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 170
B.8 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 170
B.9 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. 171
B.10 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 1000$, $r_m = 1$, $r_d = 1$. 172

B.11 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 2$, $r_d = 1$. 173
B.12 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 4$, $r_d = 50$. 174
B.13 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 2$, $r_d = 10$. 175
B.14 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 4$, $a_d = 50$, $r_m = 6$, $r_d = 1000$. 176
B.15 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 1000$, $r_m = 1$, $r_d = 10$. 177
B.16 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 4$, $a_d = 100$, $r_m = 4$, $r_d = 10$. 178
B.17 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 8$, $a_d = 50$, $r_m = 4$, $r_d = 10$. 179
B.18 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 1$, $a_d = 1$, $r_m = 1$, $r_d = 1000$. 180
B.19 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 4$, $r_d = 100$. 181
B.20 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 1$, $r_d = 100$. 182
B.21 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 183

B.22 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 183
B.23 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 184
B.24 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 184
B.25 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used. 185

C.1 One cluster environments 188
C.2 Half cluster environments 189
C.3 Two clusters environments 190
C.4 Four clusters environments 191
C.5 Uniform environments 192
C.6 One cluster environments 193
C.7 Half cluster environments 194
C.8 Two clusters environments 195
C.9 Four clusters environments 196
C.10 Uniform environments 197

List of Tables

4.1 The foraging performance of RepAtt when using chemotaxis parameters $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ for both noiseless and noisy communication. Each value represents mean and 95% confidence interval for 30 repetitions of each simulation. 65

5.1 Inference time (in seconds) for Raspberry Pi 4 (Pi4) and Erle-Brain 3 (Pi3) computers. The values represent the mean inference time and standard deviation for a selected video in the test dataset. 86

5.2 Analysis of the MobileNet-SSD and Tiny-YOLOv3 performances and metrics computation by analysing all frames of the test dataset where the 679 filtered litter objects visible (that is, analysis of the data at the full frame rate of 50fps). 90

5.3 Modelling data from testing MobileNet-SSD on resampled data based on frame rates observed on the Raspberry Pi 4 (Pi4) and Erlebrain (Pi3) platforms. 92

5.4 Comparison of the model probabilities of Tables 5.2 and 5.3 with Monte Carlo simulations using the corresponding P_{u2s} and P_{s2s} probabilities. 93

6.1 Chemotaxis activation distance d_c versus robots distances from nest d_r. Each value represents the mean number of robots and 95% confidence interval for 30 repetitions of each simulation. Maximum simulation time was $t_{max} = 1500$ seconds, swarm size was 10, $a_m = 10$ and $a_d = 1000$. 112

6.2 Scatter plot of a sample simulation showing a swarm of 10 robots searching for 100 targets uniformly distributed within a 14 m radius search area. . . . 118

6.3 Comparison of number of targets found within a circular search area of radius 14 m and d_c of 10, 12 and 14 metres in unbounded world. Chemotaxis parameters were $a_m = 10$ and $a_d = 1000$. Each value represents the mean of 30 independent simulations and 95% confidence interval. 119

6.4 Scatter plot of a sample simulation showing swarm searching for 100 targets uniformly distributed within a 100 m $\times$ 100 m search area. 120

6.5 Targets found as exploration time, t, progresses. $d_c = 12$ m, $a_m = 6$ and $a_d = 1000$ for chemotaxis based approach. Each value represents the mean of 30 independent simulations and 95% confidence interval. 121

C.1 Analysis of the MobileNet-SSD and tiny-YOLOv3 performances and metrics computation by analysing all frames of the test dataset and extracting data based on the false positive detections on the test dataset. These were not used in the vision model presented in the thesis in order to simplify the model and focus on scenarios where robots vision system fails to detect targets in its immediate surrounding. 187

Chapter 1: Introduction

1.1 Background

A team of two or more autonomous robots working together to accomplish a common task is called a Multi-Robot System (MRS) [2]. An MRS provides a unique advantage over single robots in tasks that would be impractical or inefficient for a single agent acting alone. Task deemed suitable for MRS include [3] tasks that: are inherently distributed, have high complexity when using a single robot, can benefit from multi-robot parallelism to significantly reduce completion time, and are easier to solve using multiple simple robots than with a single powerful/complex robot. A major characteristic of MRS is that individual robots in the system are far less capable on their own. However, the system is most effective when these robots cooperate with each other during task execution [2]. Types of MRS include:

1. Reconfigurable (or modular) robots, where agents can physically attach to each other in different configurations for purposes of navigation or manipulation [4].

2. Networked robots, where communication network infrastructure is used by robots, sensors, embedded computers and human users to coordinate and cooperate to accomplish a specified task [3].

3. Cooperative manipulators, where two or more robotic arms manipulate a common object [5].

4. Multiple mobile robot systems, which is made up of robots that move within an environment, such as ground, aerial or underwater vehicles [3].

Multiple mobile robot systems has grown within the past two decades to cover a large body of research. In general, techniques for implementing multiple mobile robot systems fall into either intentionally cooperative or collective swarm systems [3]. In intentionally cooperative systems, robots have access to global knowledge of state, actions or capabilities of all robots in the environment and use these information to select actions that accomplish the team's goal.

Collective swarm systems, on the other hand, emphasise robot autonomy through decentralised control and emergent behaviours. The system uses robots that have limited abilities and are only able to sense their immediate environment and communicate with nearby swarm members [6]. This approach to multiple mobile robot systems is the focus of this thesis, and will thus be explained more extensively.

Collective swarm systems or swarm robotics are inspired from biological swarm colonies. They use simple rules to achieve emergent collective behaviours without any central leader [7]. Several such systems exist in nature, examples of which include ant colonies, termite colonies, bee hives and, to a lesser extent, fish schools, bird flocks, locust swarms and herds of mammals like wildebeests. These natural swarms make use of local interaction between members and their environments to, among other goals, forage for food, build nests, clean their nests, navigate their environments and escape from predators [7–9]. The successes of natural swarms in performing tasks that are too complex for a single agent (and inefficient, or sometimes impossible, for multiple agents acting on their own) have motivated research in the development of robot systems that exhibit swarm behaviours. Some benefits for adopting swarm-based strategies for problem solving in robotics include:

1. *Robustness* to failure of a swarm member or disturbance in the environment [10]. This is because swarm systems incorporate redundancy, simplicity of agent interactions (or algorithms), decentralisation of swarm coordination and multiplicity of environment sensing [11].

2. *Distributed workload* among members that make up the swarm. This makes it possible to perform tasks that cover large spatial areas and provide wide situational awareness for the swarm system [10].

3. *Flexibility* in problem solving because the swarm offers solutions to tasks by utilising coordination strategies that can adapt to changes in the environment [11]. For example, during foraging, bees are able to switch to richer sources of nectar when the quality of their current foraging site degrades [8]. The autonomy of swarm members and simple interaction algorithms also help swarms adapt to different environment constraints or setups.

4. *Scalability* of swarm coordination scheme which is not significantly impacted by changes in swarm population [11]. Individual swarm members can join or quit the task without disrupting the swarm objectives [12].

5. *Parallel task execution* by a large number of swarm members performing their respective tasks simultaneously without dependence on a leader or global state of swarm members and information about progress on task execution [10, 12].

Extensive research in swarm robotics largely motivated by the features outlined above have been conducted. A significant amount of work in this research area involves observing tasks accomplished by biological swarms and adapting them to applications that solve real-world problems using non-biological swarm agents. The range of collective tasks implemented in swarm robotics include [13]: flocking, self-organised aggregation, foraging, object clustering and sorting, collective navigation, path formation, self-deployment, collaborative manipulation, task allocation, odour source localisation, object assembly, self-assembly and morphogenesis, coordinated motion, group size estimation, distributed rendezvous, collective decision-making, and human-swarm interaction. Of the aforementioned swarm tasks, foraging is the most used test bed application for swarm robotics systems [14].

Foraging is a term generally used to describe a process where individuals making up the swarm explore an environment in search of one or more objects and transport them to designated nest location [14]. This is best exemplified by the food search activities studied in some ant and bee colonies. Some ants are able to search and collect food located long distances from their nest by leaving pheromone trails on the path to found food sources [9]. Similarly, honeybees forage by utilising unique dance patterns to communicate the quality, direction and distances of food that are as far as 10 km from their nest to other members of the swarm [8]. In swarm robotics research, foraging has been used to investigate interference amongst swarm members, and as a test bed for collective exploration, transport and decision-making [14]. Foraging can be considered an abstraction for many real-world applications such as picking litter, demining, planetary exploration and search and rescue operations. In both natural and robotic swarms, each foraging agent needs the ability to navigate its environment, detect, manipulate and transport targets to a designated site. In addition, agents also use local communication and sensing to improve the foraging strategy of the swarm.

1.2 Motivation

The Self-Repairing Cities Project [15], where robot technologies are developed to identify, diagnose and repair infrastructure in the society with the aim of eliminating disruption in services in UK cities, serves as the main motivation for this research. The aim of this research is to examine the application of swarm robotics technology for cleaning up litter within open urban spaces, such as parks and squares (Fig. 1.1). Swarm robots tasked with collecting litter in an environment need to efficiently explore the search area and exploit litter clusters.These robots are faced with numerous challenges including communication, computer vision, object manipulation, terrain navigation, task allocation and energy management. For example, the communication channel among the robots could be noisy, which means robots at the receiving end of communicated information need a reliable noise-filtering mechanism. Imperfections in robot vision system means they could fail to detect litter objects or wrongly classify

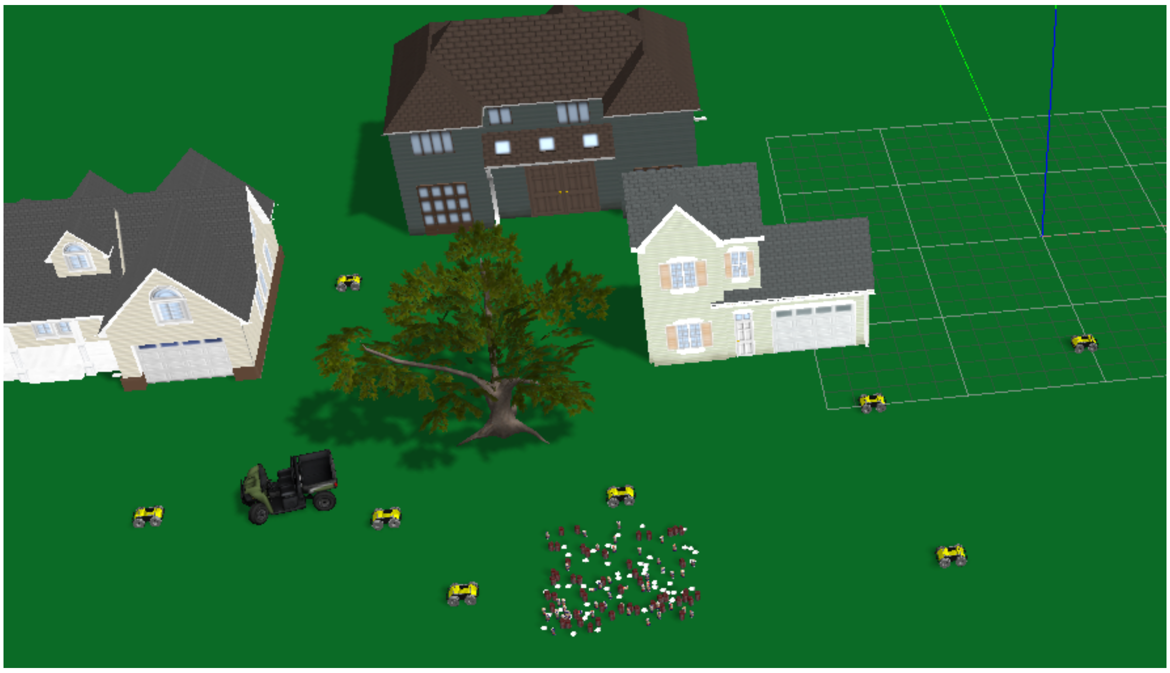


Fig. 1.1 A test model representation of a group of swarm robots tasked with collecting litter in a residential area.

non-litter objects as litter, thus, causing them to misrepresent their local environment. This thesis focuses on the vision and communication subset of these challenges to inform design choices in the development of a robust and scalable swarm foraging algorithm. In order to conform with the existing swarm robotics framework, this thesis emphasises the following key areas:

- bio-inspired robot search algorithms;
- minimal robot capabilities in sensing, communication, computation and complexity;
- autonomy of individual robot agent;
- emergent collective behaviour through decentralised control.

These choices were used within a design framework that models robot communication and vision systems from real-world experiment data to develop a novel swarm foraging algorithm. This hardware-based modelling approach helped maintain focus on the real world deployment of the technology, and consequently aiding in the elimination of unrealistic algorithmic design choices. Furthermore, this design method greatly assisted in the realisation of a flexible, scalable and robust swarm algorithm capable of handling practical imperfections in visual sensing and communication.

1.3 Aim and Objectives

This research aims to develop a simple, scalable swarm foraging algorithm that is robust to realistic noise levels in robot communication and vision. This will be achieved through the realisation of the following objectives:

1. The development of a simple swarm foraging algorithm using inspiration from biological systems.

2. The use of experimental data to develop a realistic communication model to aid collective behaviour of the robot swarm that use the biologically inspired foraging algorithm. The model should also quantify the noise level in the communication channel.

3. The implementation of a vision system for the robots that reflects the performance and properties of state-of-the-art algorithms in computer vision for object detection.

4. Conducting an in depth analysis of the impact of uncertainties in robot communication and vision on the foraging performance of the swarm.

1.4 Contributions Made

A considerable part of this research comprises of extensive simulation tests that account for hardware constraints and other related limitations. The following highlighted contributions have been made to the body of knowledge in swarm robotics and to help bridge the existing gaps in the field:

1. Development of a swarm foraging algorithm adopted from the chemotactic navigation behaviour of biological organisms such as *Caenorhabditis elegans* and *Escherichia coli*;

2. Extensive study using simulation techniques to investigate the impact of noisy communication and imperfections in robot vision systems on the collective behaviour of swarm foraging robots.

3. Development of a probabilistic vision model from the performance characteristics of deep neural network based object detection algorithms. The model also takes into consideration the constrained computation hardware limitations of swarm robot platforms.

4. Implementation of a chemotaxis-based virtual fence for keeping swarm of exploration robots from drifting away from a desired work area near a stationary or mobile nest in unbounded environment;

1.5 Thesis Outline

Chapter 2 reviews literature and concepts that form the foundation for algorithms and models developed in the thesis. These include the biological inspiration for swarm robots foraging and the state-of-the-art in foraging algorithms and robot vision systems.

The Repulsion-Attraction algorithm (RepAtt) for a swarm of foraging robots is presented in Chapter 3. Also included is the development of swarm communication model and experiments to analyse RepAtt's foraging performance, robustness to changes in swarm search space and scalability with increasing swarm size.

Chapter 4 focuses on the modelling of noise in the communication system used by the swarm of foraging robots. It also develops a simple filtering system that mitigates the negative impact of the noise and improve the swarm's foraging performance, scalability and robustness.

In Chapter 5, a probabilistic vision model and its effects on swarm foraging is studied in detail. The vision model is based on the performance of deep neural networks when trained to detect litter in a robot's visual scene. The chapter extensively studies the impact of robot detection accuracy and vision processing rate on the foraging performance of the swarm.

Chapter 6 details the use of virtual fence to restrict swarm robots from drifting away from a work area in unbounded environments. The virtual fence takes inspiration from the chemotaxis search behaviour of *Caenorhabditis elegans* and is effective for both stationary and mobile nest robot when tested on an environment exploration task. In addition, the virtual fence is more practical than a physical wall that contains robots within the work area.

Chapter 7 summarises the contributions of the thesis, draws conclusions and provides recommendations for future research in the area.

Chapter 2: Literature Review

2.1 Introduction

Swarm foraging is an important benchmark for many multi-robot tasks. This chapter discusses the biological inspiration of swarm recruitment strategies in Section 2.2. It serves as the foundation for most swarm foraging algorithms discussed in Section 2.4. The different technologies used by researchers to implement the recruitment of robots on hardware platforms is discussed in Section 2.5. Random search strategies, which form the basis for many swarm algorithms are discussed in Section 2.3. A critical feature required by foraging robots is the ability to detect targets of interest in their surroundings. Thus, it is important to discuss the state-of-the-art in object detection in swarm robotics and computer vision research (Section 2.6). A survey of previous and existing work on the utilisation of robots for handling litter in an environment is discussed in Section 2.7. The various metrics used in measuring the performance of swarm foraging algorithms are discussed in Section 2.8. These topics form the body of research that are relevant for the development of the swarm foraging algorithm in this thesis. Section 2.10 summarises the chapter.

2.2 Recruitment Strategies in Swarms

Recruitment is a means of getting one or more swarm members to join in exploiting a resource. There are various means of recruitment found in biological swarms with the strategies found in ants and bees receiving the most attention.

2.2.1 Recruitment in ants

Depending on the recruitment strategy and ant species, ants recruit other nest mates using mechanisms such as chemicals, sight and/or antennation. The collective strategies used by ants when foraging food vary from solitary foraging to mass recruitment [9].

In solitary or individual foraging, ants leave nest individually and forage food to the nest without any cooperation or communication with other foraging ants. This strategy is useful if food is distributed, unpredictable and can be carried back by a single forager. In such situations, the cost of recruiting other foragers outweighs the benefit to be gained from collecting the resource alone. It is a strategy found in many species of ants in the genera *Harpegnathos*, *Pachycondyla* and *Cataglyphis* [9]. Solitary foraging is considered to be the simplest strategy possible for a swarm [16].

A second strategy is tandem running, where a successful forager (scout) leads a nest mate (follower) to the discovered food source. The scout recruits the nest mate using antenna contact or using odour signals. The two ants then keep a physical contact with each other as the scout leads the way to the food site [17]. In turn, the follower also recruits a nest mate during its next foraging trip. This way, information about the food source is cascaded to more members of the swarm [18]. This recruitment strategy is slow, but useful for recruiting ants to hard-to-find sites. Tandem running has been observed in species of genera *Temnothorax* and *Pachycondyla* [9]. Related to tandem running is group recruitment, where the successful forager recruits two or more nest mates to the resource location. It often involves both motor and chemical signals from the recruiting forager [17]. This has been observed in ants of genera *Camponontus* [9].

Pheromone trail is a popular recruitment strategy studied in literature and is considered to be the most complex [17]. In this strategy, scouts randomly explore the environment in search of food. Successful scouts then lay chemical (also known as pheromone) trails that connects the food source to the nest. Nest mates are attracted to these trails and also lay pheromones on them, thus reinforcing the path connecting the food source to the nest. It is a means of mass recruitment of nest mates to rich, but short-lived, food resource. Depending on the volatility of the pheromone, a certain minimum number of ants are necessary to maintain the trail, because the trails evaporate and diffuse over time. This makes it dependent on the amount of positive reinforcements the trail gets from other ants, to make it last longer [9]. Pharaoh's ants (*Monomorium pharaounis*) have been observed to recruit using pheromone trails.

Food resource sites are, sometimes, stable such that the swarm could forage continuously from the same site for a prolonged period. Examples include tree leaves and extrafloral nectaries. In such cases, the foraging activity can lead to *stable trunk trails* that connect nest to the food location [9]. This trunk can also branch into small trails to cover the foraging area. Foraging ants follow these trails to collect food and return to the nest. This has been observed in ants of genera *Atta, Forelius* and *Pogonomyrmex*.

Army ant 'raids' is a strategy which involves large numbers of ants, often up to 200,000, moving in a unified foraging front that sweeps a tract of forest [19]. They also form a trail network that could exceed 100 m behind the swarm which consolidates into a single column of traffic for returning food to the nest and allows foragers to return to the raid. The strategy gives army ants strength in numbers to overpower their prey and forage them to the swarm's nest.

Team transport involves two or more ants cooperatively transporting items from resource location back to the nest [20]. This is useful for foraging large or cumbersome items which cannot be effectively transported by a single ant. A related strategy known as ‘bucket brigading’ involves foragers that only transport items for a short distance to a ‘cache station’, where it is picked up by another ant that transport it to the next station closer to the nest [9]. This way, the foraged items continuously change stations until they are deposited in the nest. While team transport requires multiple ants collaborating simultaneously, the bucket brigading involves sequential transfer of items from one ant to another. Bucket-brigading has been observed in ants of the genera *Acromyrmex* and *Atta*.

2.2.2 Recruitment in bees

The most popular recruitment strategy in bees is the dance communication language of honeybees, *Apis mellifera*. Bees are well known for their roles in pollination and production of honey and beeswax [21]. Their amazing ability to forage nectar, pollen and water from sites located hundreds of metres from their nest has intrigued researchers for centuries [22]. In the following paragraphs, the recruitment strategies of honeybees, stingless bees and bumblebees are discussed.

Foraging honeybees are categorized into: scout bees which leave the nest to search for the food resource in the environment; and reticent bees (40-90% of forager population) which wait in the beehive to be recruited to the found food source(s) [23]. The dance language paradigm states that successful scouts communicate their findings to reticent bees at the nest using a waggle dance routine. The duration of the dance encodes distance information of the food source, while the alignment of the dancing bee’s body relative to the sun’s current azimuth contains goal direction information. By observing the dance pattern of the scout bee, reticent bees get these distance and direction information of the scout’s finding, which they use to navigate to the food source [8]. In addition, the bees are able to optimise their foraging rewards by adapting to change in quality of food sources, such that they always forage from high-quality sources of nectar [24]. The bees navigate between food sources and their hives by keeping a map-like spatial memory of their environment using sun compass, landmarks and other sensory information [25].

The second paradigm for honeybee recruitment success during foraging hypothesises that reticent bees get recruited through scent. The proponents of this paradigm argue that bees are primarily recruited by odour of the successful scout and the dance is simply to gain attention of reticent bees. Recruited bees then follow the odour trail to locate the food source [22]. The bees are able to follow the trail because they have a well refined olfactory system that can recognise a vast array of different compounds at parts per trillion, which is the reason for research into methods and technologies of using honeybees as living biosensors for chemical signals [22]. Though the two paradigms to recruitment in honeybees differ in many ways, they share some fundamental similarities. Scout bees leave the hive in search of resource, successful scouts return to the nest and recruit reticent bees to the resource site. Once recruited, bees continue to forage from the same site until the resource is exhausted, weather conditions become unfavourable, there is abundance of resource in the hive or they get recruited to a more profitable resource site [23].

Stingless bees (*Hymenoptera, Apidae, Meliponini*) are the largest group of social bees and have more than 400 species. Only a small fraction of stingless bees species have been studied in detail [26]. These bees live in populous permanent colonies and are able to coordinate worker's actions and respond to the spatio-temporal changes of food availability in their environment without any centralised control function [26]. An individual stingless bee makes foraging decisions based on a variety of information sources which can be intrinsic to the bee (spontaneous preference and memory) or extrinsic (colony state, nest mates, conspecific non-nest mates and heterospecific bees in the field) [26]. Just as in honeybees, naive stingless bees are primarily recruited at the nest. However, unlike honeybees, stingless bees have diverse systems of recruitment. For example, in several stingless bee genera, recruits directly follow foragers the entire distance to food sources (also known as piloting) or follow scent trails laid on surfaces by foragers [27]. Stingless bees in the nest also use information such as the availability of pollen in the nest, sounds, movement patterns, trophallaxis (direct transfer of food between bees) and odour of forager bees to intensify or increase their chances of leaving the hive in search of food source [26].

In bumblebees (*Bombus terrestris*), recruits are not given information of direction and/or location of food sources. Bees that succeed in foraging make frenzied movement in the nest to distribute alert pheromones that increases bee activity in the nest. This process informs bees at the nest about the general availability and the scent of rewarding food sources [28]. Reticent bees use this pheromone information and nutritional status (amount of food in the nest) to make individual decisions to start foraging activity [29]. The response to pheromone is stronger for low nectar reserves in the nest, while if there are high volumes of nectar in the nest, only a few bees get recruited. Foraging patterns for bumblebees in the field has been described using Lévy flight, which involves a combination of foraging from nearby and distant flowers for nectar [30].

2.3 Biological Search Strategies

The survival of many organisms, from microbes to large animals, depends on their ability to detect, integrate and process varieties of internal and external cues when navigating complex environments for purposes such as locating food, avoiding predators or attracting mates

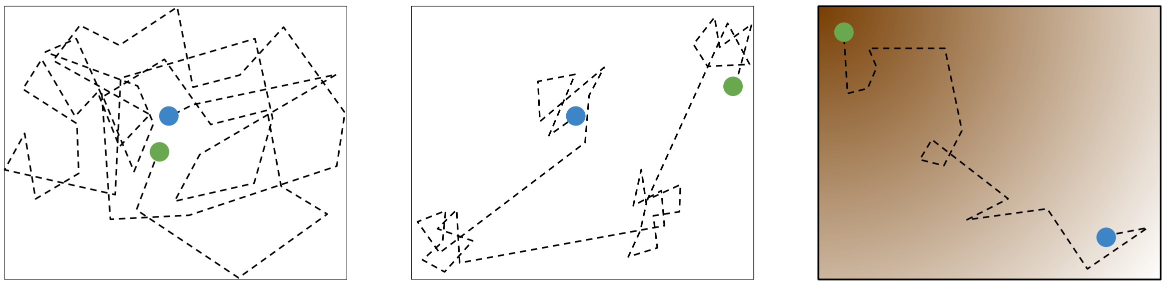

(a) Unbiased random walk (b) Lévy walk (c) Biased random walk

Fig. 2.1 Three types of random walk. Blue and green circles represent starting and stopping locations respectively, while the dotted lines represent the path of the agent. In (c), the brown gradient represents change in concentration of an attractant signal the agent measures to inform its biased search toward the signal source.

[31]. One of the most popular navigation behaviours is the random walk — a Brownian-like motion in which the organism follows a fairly straight path and randomises its movement direction at random intervals. This motion could be pure random walk, where the straight motions and random angles are sampled from a uniform distribution (it results in movement pattern similar to Fig. 2.1a). A second class of random walk, known as Lévy walk (or flight), observed in biological systems is such that the agent makes localised random motions with occasional long straight motions [32, 33] as illustrated in Fig. 2.1b. The agent's movement path-length can be described using a Lévy distribution. A third class is the biased random, where the agent's random walk is biased based on some signal it senses in its environment (shown in Fig. 2.1c). An example of biased random walk is the chemotaxis search observed in organisms such as *Escherichia coli* bacteria and *Caenorhabditis elegans* nematode where the agent's movement is in response to the temporal gradient of the concentration of attractant or repellent chemicals they sense as they explore the environment [34, 35].

In unbiased random walk and Lévy walk, the organism search behaviour is controlled by some internal cue (or state). Thus, within a swarm foraging context, each agent performs independent exploration of the search space to locate targets and when found, they do not recruit others to the found resource. It has been shown that Lévy walk significantly outperforms unbiased random walk for foraging tasks, especially when targets are sparsely and randomly distributed within the search space [36, 37]. This is because it reduces the probability that an agent will revisit areas it has already explored.

Chemotaxis-based search in *E. coli* and *C. elegans* have attracted a considerable amount of interest from researchers. Some work have focused on understanding biological and chemical parameters for chemotaxis [34, 35, 38, 39], control systems and mathematical modelling of the organisms' chemotactic movement [31, 40–42] or using it as inspiration for robot navigation behaviour [33, 43, 44]. When using chemotaxis, the organism uses its chemoreceptors to continuously sense its environment during exploration. The motion can broadly be classed into two: swim mode – characterised by nearly straight forward movement; and tumble mode – used by the organisms to change their direction [43]. The organism's decision to modulate its swim is informed by the temporal gradient of signal concentration it senses. When it senses positive concentration gradient of an attractant chemical (or a negative gradient in a repellent chemical), the organism will tend to make longer swims. However, when it senses negative temporal gradient of the attractant chemical, the organism makes short swims, thus causing it to make frequent tumbles of up to 180°. This change in direction increases the chances that the organism will detect a positive attractant gradient and swim toward it [38, 45–47].

The chemotaxis search described in the preceding paragraph specifically describes organisms' modulation of its random search based on the change in chemical gradient. Broadly speaking, many biological organisms can modify their movement or exploration behaviour in response to change of some parameter they can sense in their environment. For example, when organisms modulate their movement in response to light, it is termed phototaxis [44, 48]; responding in similar manner to temperature gradient is thermotaxis [49, 50]; and when responding to sound, it is termed phonotaxis [51, 52]. These navigation behaviours are generally more complex than those observed in *C. elegans* and *E. coli* chemotactic behaviour. For example, in phonotaxis observed in crickets, female crickets track the specific calling song to localise the male crickets instead of randomly exploring the environment based on intensity of the sound signal.

2.4 Swarm Robots Foraging Algorithms

Most modelling studies of foraging algorithms assume correlated random walk rather than a systematic search strategy [9]. This random walk pattern of the foraging agent may be modified to become straighter in response to congestion in its local environments to minimise inter-robot interference and explore new regions in the environment. It can also be more tortuous (larger turn angles), keeping the forager within a region's vicinity thus causing a region to be thoroughly explored. Optimal tortuosity of the search path is dependent on factors such as target distribution and number of cooperative searchers [9]. In addition, while searching, foragers need to track their nest location to enable them return there. A standard model used for tracking the nest is the ant *Cataglyphis bicolor*. These ants use path integration algorithm (continuous update of distance and direction of nest from distance and direction ant has walked) to keep track of their homing vector [9]. Swarm robotics foraging research has led to the development of a number of algorithms, which in many cases build upon the correlated random walk. Some specific goals focused on in swarm robots foraging literature include minimising foraging time, maintaining swarm energy level and minimising inter-robot interference to eliminate overcrowding within the environment. In subsequent paragraphs, some swarm robots foraging algorithms are discussed briefly.

The multi-robot foraging algorithm without communication in [53] is one of the earliest works in the area. The robots used were controlled using move-to-goal, avoid-static-obstacle and noise motor schemas. Each of these schemas were represented as vectors (i.e. they have magnitude and direction), which were dependent on whether the robot was in foraging, acquire or deliver state. A robot's motion was defined as the resultant of the motor schemas. Robots in the foraging state used random walk to search for attractors (because the noise motor schema had the highest magnitude in this state and randomly generated paths the robot took). When the robot locates an attractor, it switches to move-to-goal state to pick it up and transport it to a central nest. In subsequent improvements of the algorithm, robots recruited other swarm members to the attractor's site by broadcasting its location on a shared memory accessible to all swarm members [54, 55].

In [56], two swarm foraging algorithms were developed to investigate the role of recruitment in swarm of foraging robots to determine conditions that favoured collective foraging and those that did not. This was necessary because there are associated costs in algorithm and hardware design when collective strategies are employed for foraging. The two strategies were termed individualist and collective foraging. In the individualist approach, robots used random walk to search for targets and transport them to the nest (they located the nest by following a home beacon). Upon returning a target to the nest, a robot assessed the energy efficiency of the returned target, which it then used to decide whether to return to previous foraging location or resume a random walk search for new targets. For the collective foraging approach, robots communicated the energy efficiency and locations of targets they find with nearby robots. Robots got recruited to target locations if the energy efficiency of targets communicated to them were better than their target goal or when they have been unsuccessful in locating any target. The results indicated that collective foraging is advantageous when it solves the problem of congestion and when targets are hard to find (because they are clustered within small areas). The individualist foraging approach was more cost effective where resources were easy to locate, information were unreliable, resources were abundant or additional investment cost for implementing collective behaviour outweighs benefits to be gained from improved foraging performance.

In later work in [57], a bee-inspired algorithm, where robots only communicate resource location at the nest was developed. The nest area is divided into unloading region to drop off foraged targets and dance floor for recruiting observer robots. Robots started out as observers that performed random walk within the dance floor. With a fixed probability, observers transitioned to scout robots, which left the nest and used Lévy walk to search for foraging material. An unsuccessful scout returned to the dance floor as an observer, while successful foragers estimated the energy efficiency of targets they found, dropped them off at the unloading region and went to the dance floor to recruit observer robots within communication range. The experiments investigated the effect of information flow to measure how swarms self-organised collectively or adapt individual behaviours in order to cope with changing foraging conditions. Simulation results indicated that swarms were able to switch to more profitable foraging sites when robots used short range communication and were able to measure the rate of change of resource quality of the sites they foraged from. Other strategies such as long range communication and relying on foraging experience of other robots were only good for specific foraging scenarios, thus were not suitable for cases where swarms needed to easily adapt to wide variety of foraging conditions.

An artificial recurrent neural network robot controller whose weights were updated online by robots during foraging was proposed in [58]. The swarm task was for two robots to collectively pick up targets in their environment. Once picked up, the target reappeared in a random location and each robot involved in the pick up got assigned a score. The neural network controller defined the genome of each robot behaviour and the genome's fitness encoded the number of targets the robot had picked up using the genome. Robots executed this genome to search for targets for a limited time while sharing the fitness and genome data with other robots it encounters. At the expiration of the time assigned for executing the genome, each robot assessed its genome and those it had collected. The robot then selected the genome with the best fitness to mutate and use for the next time-limited exploration. The approach helped all robots in the swarm to continuously update their strategy to optimize their foraging efficiency.

In [59], each robot in the swarm was represented as a particle in a Particle Swarm Optimization (PSO) algorithm. The dimensions of the each particle was modelled as the 8 sensors with which the robot measured the potential in its immediate environment. The potential field computation was dependent on attraction and repulsion weights. The attraction weight is a linear equation dependent on the proportion of the targets found by the swarm, while the repulsion weight was computed based on the percentage of the search space that had been explored by the swarm. This meant that the PSO algorithm required global update of total targets in the environment, number of targets left and how much of the environment had been explored by the swarm. In addition, robots in the swarm also had access to the global best fitness, which it used as part of the equation to compute its movement direction, based on the PSO algorithmic steps.

Inspired by desert seed-harvester ants, Hecker and Moses [60] developed the Central Place Foraging Algorithm (CPFA) for swarm of robots searching for and retrieving items in their environment. These ants searched for food items independently using memory of their foraging experience and occasionally laid pheromones to recruit other ants when they discovered a large deposit of food in a location. In CPFA, robots leaving the nest randomly selected a region to forage from within the search space, navigated to that location and performed a localised search for targets there (known as informed search). The robot gradually widened its search area to explore a wider space if it had been unsuccessful in finding targets near the chosen site (known as uninformed search). While searching, the robot's decision to return to the nest was controlled by a time dependent probability of giving up the search. When the robot located targets during search, it picked it up, noted the location it found the target and estimated the density of targets within that region. The robot then returned the found target to the nest and probabilistically decided whether to lay pheromone waypoint at the nest to point to its last foraging location or refrain from doing so. The robot also decided whether to return to previous foraging location, random location or make use of pheromone waypoint laid by other robots at the nest. In swarm design approach, the nest served as a location to deposit targets and get information about pheromone waypoints laid by robots in the swarm, thus robots recruitment is facilitated by the nest. To mitigate the effects of diminishing returns encountered in tasks where robots were to locate and forage all resources in the environment, in [61] the CPFA was extended to help the swarm estimate the number of target clusters in the environment and allocate search areas to different clusters based on the estimated size of each cluster.

The Multi-Place Foraging Algorithm (MPFA) was proposed in [62] as an extension of CPFA. In MPFA, foraging robots selected from multiple depots to drop foraged materials, based on which was closest to them. One motivation of MPFA was that it helped the swarm cover a wide area and minimised the time spent by robots returning foraged items to a central location. For depositing targets at a central nest, MPFA was extended in [63] by making the multiple depots transport the items deposited in them to the central nest. Additionally, the depots autonomously determined areas to position themselves within the search space in order to minimise the journey performed by exploring swarm robots. They showed through extensive simulations that the multi-depots approach outperformed classic CPFA when the search area was large. It also reduced the congestion at the nest experienced when robots used only CPFA for foraging.

In [64, 65], the swarm of foraging robots adapt their search strategy based on the distance of targets resource from the nest. Initially, robots left the nest in search of targets to forage using random walk. During this search, some robots decide to form beacon networks that will provide gradient information toward the nest for robots involved in the search for the target. If the target is found, information about its location is relayed as a gradient through the beacon network for other robots who use the gradient information to locate the target deposit. If no target was found and all robots have transitioned to a beacon network, the swarm transitions to the sweeping algorithm. The sweeping algorithm increases the swarm's reach in the search space by forming a one-dimensional chain of robots that form a gradient from the nest to the end of the chain. This chain then makes a clockwise sweep using the nest as the centre. If the swarm locates a target deposit, foraging robots follow the gradient to pick up targets and return them to the nest. On the other hand, if the sweep was unsuccessful in locating targets, the swarm disbands the chain and all agents perform independent random walk search for targets and also use random search to locate the nest once they find a target.

The Cooperative-Color Marking Foraging Agents (C-CMFA) algorithm was proposed in [66]. In the model, robots used stigmergy to forage within a grid environment, where they can either move in up, left, right or down direction. When robots leave the nest, they explore the environment in square spiral format while leaving a trail on each grid cell they visit. This trail is an integer hop count from the nest to the cell, which resulted in an artificial potential field. Upon encountering a target, the robot picks it up and follows the negative gradient of the potential field to the nest, while laying pheromone trails on this path to create an optimal route from nest to target location. Robots cooperate with other swarm members by avoiding cells that have been visited by other robots and following pheromone trails they encounter.

In [67], the Distributed Deterministic Spiral Search Algorithm (DDSA) for foraging swarm robots was proposed. DDSA operates by making each robot in the swarm follow a square spiral path starting from the nest such that the space between each spiral is no more than the sensing distance of individual robots. Such search pattern eliminated resampling of areas already covered by other robots in the swarm and guaranteed that the whole environment would be covered in a deterministic fashion. When a robot locates a target, it returned the target to the nest and resumed exploration from where it left off. DDSA performed well for swarms of 6 to 15 robots, but continuously got worse when tested on larger swarm sizes. In addition, the algorithm was well suited for environments that have targets located near the nest and gave poorer performance when targets were far off.

In [68], robots made foraging decisions by adaptively responding to the amount of food available in the environment. The robots used a sigmoid function and response threshold to decide between three states: Resting (remaining at the nest), Searching and Interacting (leave the nest to forage targets and communicate with other swarm members), and Returning (return to the nest to deposit targets or recharge battery). In the proposed method, a robot at the nest decides to go foraging when the stimulus was high. The robot then performed random search of the arena and interacted with its neighbours by exchanging state information with them. Through direct communication, the robot learned of the foraging success of its neighbours, which helped it estimate its chances of successfully finding targets to forage. Once the robot found a target or ran out of battery, it returned to the nest. The robot's subsequent decision to leave or remain at the nest is dependent on its foraging experience and that of other robots it had interacted with during its last foraging expedition. The result of this adaptive role selection by the robots in the swarm resulted in adaptively resizing the number of foraging robots in response to availability of targets in the environment so as to optimise the foraging efficiency of the swarm.

The response threshold-based approach was also used in [69], where the swarm's goal was to maintain a desired food level at the nest. The robots did not communicate with each other, but were able to sense the level of food at the nest, and individually decided to continue waiting at the nest or leave the nest in search of food, based on the stimulus level. The stimulus in the proposed work was the difference between desired food level and current food level at the nest. Initially, all robots wait at the nest and routinely checked the nest's food level. If the stimulus was high (critically low food level in comparison to desired level), the robots were more likely to leave the nest in search for food. Upon returning to the nest, robots default to the waiting state and routinely checked the food level. When level of food at the nest was close to or more than desired level, the robots had little propensity to go out foraging. In the tests (using simulated and e-puck robots), food items were placed randomly in the environment, and level of food at the nest decreased over time to simulate food consumption.

Bucket brigading is also a common swarm foraging approach used in literature, which involve robots transporting targets through a chain of handovers between multiple intermediate robots from the resource location to the nest. In [70], robots started at arbitrary locations in the search space and explored the environment to locate an Ants Colony Optimization (ACO) algorithm generated trajectory that connected resources to the nest. Once on the ACO trajectory, the robot autonomously balanced its work time with the robots on its left and right in the chain. They do this by local communication of work time between each robot and their intermediate neighbours, and use the information to compute appropriate time/distance to travel when loaded. In this way, over time, the robots in the chain reach a consensus work time where no swarm member was overloaded.

In [71], a cost function was proposed which robots used to determine the path length they should travel when they transported the foraged objects toward the nest. In this task partitioning approach, robots did not communicate directly with other swarm members, but searched for targets using random walk. The first time a robot found a target, it transported it directly to the nest, and estimated the cost of this transport and determined a path length it would use on its next foraging trip. During subsequent trips, the robot transported targets for only short distances before depositing them for other robots to pick up and transport to the next location based on their individual path length (or to the nest if the robot was near the nest). Robots used odometry information to return to their last foraging location to perform localised search for more targets. They search wider areas if no more targets were found in their last foraging location.

Random walk search was used in [72] (robots move forward and stochastically change their direction) to locate targets to forage and return to the central nest. To specifically solve the problem of congestion at the nest location and conserve energy, role division and searching space division was introduced. For role division, the swarm was divided into *collecting* and *conveying* robots. The collecting robots randomly searched for uniformly distributed targets in the environments and deposited them at intermediate temporary stores, while the conveying robots picked up targets from temporary stores and conveyed them to the central nest. The addition of temporary stores (near the nest) in the search space divided it into two regions such that areas between nest and temporary storage were explored by conveying robots, while regions beyond the store were explored by collecting robots. In addition to reducing congestion around the nest region, the approach also improved energy efficiency of the swarm because collecting robots did not have to make the long journey to the nest to deposit targets before resuming their search, and conveying robots only needed to search for targets within a limited area. Additionally, robots adaptively switched between roles based on time thresholds for each role. This helped the swarm adaptively balance between collecting and conveying robots.

Congestion of robots on paths that lead to the nest is one of the issues reported in swarm foraging literature. When robots converge to a path that connects nest to resources, the inter-robot interference increases due to the congestion. In [73] and [74], the Robot Probabilistic Cellular Automata Ant Memory (RPCAAM) was proposed, where robots used pheromone-based communication to locate and recruit swarm members to a targets source. When in the homing state, a successful robot chose one out of multiple nest locations to return the target to by using a probabilistic decision function that considered the robot's distance from the nest nearest to it. The function assigned high probability to the robot's neighbourhood cell that had the shortest distance to the nest, and lower probabilities to other neighbourhood cells. Thus, using this probabilistic decision process, the robot sometimes followed a suboptimal path and reduced congestion along the shortest path to the nest.

2.5 Recruitment in Swarm Robots

Recruitment is the means by which swarms of robots cooperate with other swarm members. It plays the major role in improving swarm efficiency, scalability and robustness when accomplishing tasks such as foraging, aggregation, collective transport, swarm navigation among other swarm tasks. Many of the strategies used in swarm robotics are based on several biological phenomena. In swarm robotics, recruitment is mediated by some form of communication. Much research has gone into various technologies that can aid decentralised recruitment of robots using both stigmergy and direct means of recruiting swarm members.

2.5.1 Stigmergy-based communication

Stigmergy-based communication refers to the use of markings in the environment (for example leaving pheromone trails) as a medium of communication. Many ant-inspired approaches model the pheromone-based recruitment. In [75], an arena covered with phosphorescent paint that glowed in the dark for several minutes after exposure to ultraviolet light source was used. Robots were equipped with a downward facing ultraviolet LED, which they used to stimulate the phosphorescence on the floor beneath them as they traversed the arena between the nest and target source during foraging. Robots were able to get recruited to the foraging path using light sensors. In [76], a 6 m $\times$ 9 m parquet floor and approximately 1,500 RFID tags placed underneath the floor were used to store and update pheromone information. The Augmented Reality for Kilobots (ARK) system was used with a swarm of 100 - 200 Kilobots to demonstrate pheromone recruitment on robot platforms in [77]. The ARK system allowed custom extensions to Kilobots with sets of virtual sensors and actuators which are not available on the minimalist robot platform. The setup consisted of an overhead camera array to track the Kilobots, an IR-OHC for communication and a base control station to simulate the virtual environment [78]. The ARK communicates with robots in real time to update their behaviours. This system acts as a central server that managed pheromone information for the robots - Kilobots communicated with the ARK system to deposit and get pheromone information.

A different approach, which used a set of mobile deployable and movable sensor nodes was proposed in [79]. Robots communicated with the nodes using UDP multicast over a 802.15.4 communication standard. These nodes stored pheromone values which acted as waypoints to establish gradients robots can follow to connect nest to resource locations. Additionally, they can store indicators which robots read to determine the states of the nodes (inactive, faulty or deployed). Foraging robots were also able to move the nodes in the environment to optimise the foraging paths followed by recruited robots. Na et al. [80] used a horizontally placed 42" flat LCD screen to display pheromone information which robots can read using light sensors. The setup also included an overhead USB camera connected to a computer that tracked the robots and managed the pheromone system. Similarly, [81] used a 65" high resolution LCD to display pheromone information processed using a base computer station. Robots also used colour sensors to read pheromone information. In a different approach in [82], robots communicated pheromone deposit intention to a tracking device by using LED light. The tracker processes the information and displays it on the arena via a projector device. The pheromone communication experiment was performed on a swarm of 5 Alice robot platforms. In a clustering behaviour inspired by how bees cluster around regions of favourable temperatures, [83] implemented a decentralised clustering algorithm where Jasmine robots used photosensors to sense light intensities and cluster around brightest spots projected from a light source.

Some researchers have resorted to using chemical signals for pheromone laying and equipping robots with sensors to detect the presence of such chemicals. In an early work by [84], trails of camphor were laid on the floor and a robot that executed a trail-following algorithm observed in *Lasius fuliginosus* by using silicone OV-17 for sensing the presence of camphor on the robot's path. A more recent work by [85] used ethanol (C_2H_5OH) as the pheromone signal which robots lay and sense while performing a cooperative transport task. The properties of ethanol that motivate its use include volatility at normal temperature and its diffusion, which correspond to the actual pheromone properties used by ants. Their experiments involved up to 10 real robots to demonstrate the effectiveness of ethanol as pheromone substance for recruitment and indirect communication among robots.

To avoid difficulty in the realisation of pheromone-like substances for hardware robot platforms, alternate approaches have resorted to dividing the swarm into mobile and beacon robots. The beacons hold pheromone information utilised by the mobile robots when executing swarm tasks. For example, in [64], robots decide to become beacons when they sense less than 3 beacons in their neighbourhood. This approach forms a beacon network that holds gradient information in the form of hop counts between resource and nest. Mobile robots can then follow the gradients when performing foraging tasks. In a similar approach in [86], a swarm of ground robots (footbots) and aerial robots (eyebots) were used. The footbots navigate from nest to resource site by following gradient information communicated to them by pre-deployed eyebots that attach themselves to the ceiling. In [87], a swarm of networked robots communicate messages, known as virtual ants, among themselves to optimise the path that connects resource location to the nest. The virtual ants deposit pheromone information on the robots. This information is used by the networked robots to decide a shortest path connecting resource to the nest. Robots in the network that do not form part of this path, stop being beacons and become foraging robots that follow the shortest path during foraging. Experimental validation of their approach was conducted on a swarm of 20 e-puck robots that use infrared sensors to communicate the virtual ants.

2.5.2 Direct communication

With inspiration from the trophallactic behaviour (mouth-to-mouth feedings) observed in social insects such as bees, [88] developed a recruitment strategy that is based on direct signalling between robots without the need for central unit for communication. In their approach, robots maintain two dynamically updated internal states that represent their gradient values which point to resource and nest sites. Robots used infrared to communicate internal states with nearby neighbours, which is exploited by the recruits to choose its direction of motion. In [89], robots used LED lights to communicate with other swarm members that search for suitable locations to extend a chain from the nest. The searching robots used their camera to process the LED colour displayed by the last robot in the chain in order to choose its appropriate colour upon joining the chain. Robots transporting foraged items to the nest used the LED light patterns to determine nest's direction. In a swarm of cleaning miniature robots, photodiodes were used by the robots to sense LED-based communication signals from other robots in [90]. Within the context of *social odometry* (swarm of robots sharing position estimates with neighbours in order to optimise navigation between two endpoints), in [91], robots estimated the distance and orientation of their neighbours in order to optimise their paths between multiple goal locations. The platform used was the marXbot swarm robots [92], whose range and bearing sensor gives distance and angle between nearby robots and also allows neighbouring robots to send messages to one another. In [93], e-puck robots used peer-to-peer local communication during foraging to collectively select the shortest path between resource site and central nest. The robots locally exchange target and nest location estimates with their neighbours whenever they are within communication range (e-puck robots use a range and bearing board, which is similar to those of marXbots, to estimate neighbour's pose and exchange short messages).

2.5.3 Impact of noise on recruitment

Imperfection in recruitment signal can impact the success of cooperation among swarms of robots. This noise could be imperfection in location of targets communicated to other swarm members and/or uncertainty in the recruitment information sent to neighbouring robots among others. Although hardware implementation of swarm robot algorithms can reflect the impact noise has on swarm foraging, little research has been conducted on the quantification of the effect of noisy recruitment. For example, in [60], where robots recruited swarm members by informing them of the resource location, positional errors were used to simulate noisy recruitment. The robots used odometry measurements for localising themselves with respect to resource locations. Experiments using iAnt robot platforms estimated this error to be a function of distance between the robot and resource location. In the response threshold foraging model in [69], noise in sensing the level of food in the nest was used to investigate the impact of noisy sensing. The inclusion of noise significantly impacted the survival rate of the swarm. In [87], communication error was simulated as corruption or loss of virtual ant passed between robots in the swarm to help them select the shortest path to target deposit. As the percentage error increased, the swarm's ability to select the shortest path gradually degraded until they were unable to distinguish between near and distant resource deposits.

2.6 Detecting Target Objects

Object detection is an important and challenging aspect of computer vision that deals with analysing a visual scene to localise the presence of a certain class of objects (such as humans, cars, animals, litter or buildings) within the scene [94]. Computer vision has broad applications in satellite imaging, cancer research, surveillance, self-driving cars and robotics among others. Given an image frame, the objective of object detection is to use computational models and techniques to answer the question of what objects are in the scene and where they are located within the given image [94].

For real world deployment of swarm robot platforms, it is important to use realistic targets for robots to detect and study how the detection inaccuracies of the robots impact the swarm algorithm. Not all swarm robotics applications require robots to have cameras for vision. Many tasks such as aggregation, flocking and swarm navigation can be implemented without need for a vision processing system. However, vision is important for swarm tasks such as foraging, collective transport and object clustering. It is common in swarm robotics research to abstract robotic vision and give robots 100% accuracy in their vision system. This is because researchers have generally focused on developing algorithms that are effective in accomplishing swarm tasks without considering how imperfections in robot vision systems could impact their algorithm. In addition, when working with hardware platforms, the target models used are usually simplified to beacons, coloured cubes, encoded materials and similar technologies to make it easy for robots to detect them. The impact of imperfections on the vision system of robots that make up the swarm can go a long way in revealing the benefit of swarm algorithms when they are deployed in real world environments. In Section

2.6.1, target detection approaches in swarm robotics are reviewed; Section 2.6.2 reviews the state-of-the-art in object detection – deep neural network object detection – used extensively in computer vision research; and Section 2.6.3 details the steps in building an object detection model for robot platforms.

2.6.1 Individual target detection

In many simulations, robots detect objects based on their relative distance from the objects. These simulated robots generally have omnidirectional detection capability and they usually have 100% object detection accuracy within their target sensing range [57, 65, 66, 72, 88].

When implementing detection on hardware, the targets are usually of a single type. In some cases that involve clustering, objects of different colours are used [13]. In general the objects to be detected are simplified. The iAnt robot platform in [60] used a downward-facing camera to detect modelled targets as QR matrix barcode tags in the environment and used OpenCV library to process the frame images. When iAnt robots detected QR tags by aligning the robot with the tag, they detected the tags 55% of the time. However, if the detection was done by searching for neighbouring resources by simply rotating 360° about its axis and analysing the frames as it rotates, the robot was successful 43% of the time. Colour detection cameras were used to detect blue cylindrical food tokens in [69, 75] when using a swarm of e-puck robots. When using Kilobots in [78], the Augmented Reality for Kilobots (ARK) system was used to include virtual camera sensors to the robot. The Kilobots used their virtual camera to detect targets and 'pick' them up when they reach the target's location. Barcode beacons were used in [79] to represent targets which a robot scans with its camera to update a successful target pick up.

2.6.2 Deep neural network for object detection

The revolution caused by the deep neural networks has resulted in significant transformation within computer vision research (and in particular object detection) leading to systems that are able to detect thousands of different object classes in images. Deep neural network based object detection is at the moment the de facto means of object detection in the computer vision community. Although deep neural networks based detection methods were developed within the past six years, research in object detection predates this period. Within the past twenty years there has been a number of algorithms developed based on handcrafting of features of objects to be detected. One of such algorithms is the Viola Jones Detector [95] used for detecting human faces in images. Their approach simplified the computations of sliding windows of all possible scales and locations of the human face in the image by incorporating integral image, feature selection and detection cascades techniques to speed up the image processing task. The Histogram of Oriented Gradients (HOG) feature descriptor, motivated by the task of pedestrian detection, computed descriptors on the dense grid of uniformly spaced cells and used local contrast normalisation to improve accuracy [96]. Another popular algorithm predating the use of deep neural networks is the Deformable Part-based Model (DPM), which won the VOC-07, VOC-08 and V0C-09 detection challenges, used the principle of ‘divide and conquer’ to learn proper ways of decomposing objects to separate features [97]. Thus, the detection of the objects within an image was based on ensemble of detections on different parts of the objects [94].

These object detection algorithms represent the major algorithms used for object detection prior to deep neural networks based approaches started to dominate the field. They made use of techniques such as image segmentation [98], template matching [99], knowledge-based and object-based image analysis [100] for the purpose of detecting specific objects in images. By 2010, the performance of hand-crafted features for detecting objects in images plateaued, where further improvements did not result in any significant impact on performance of the detectors [94]. In addition, with increasing number of object categories, their detection accuracy decreases, while the inference time increases [101]. This reduces the wide adoption of these approaches to tasks that involve many object classes. For example, autonomous vehicles are tasked with detecting a wide variety of objects such as other road users, pedestrians and traffic signs at a very high level of accuracy. The arrival of deep neural networks based object detection brought about a significant improvement in the detection accuracies and number of object categories that could be detected within a single algorithm – a single detector can be used to detect hundreds of different object types within an image at a high degree of accuracy [102].

These deep convolutional networks gained popularity for their superior performance when learning high-level feature representations of data. This performance played a role in the drive to use it for object detection in images and resulted in the first deep neural network based object detector known as Regions with Convolutional Neural Networks features (RCNN) [103]. Since then, the field of computer vision has continued to be dominated by deep neural network based approaches grouped into ‘two-stage detection’ and ‘one-stage detection’.

RCNN is a two-stage detection approach initiated with the extraction of candidate object bounding boxes developed using selective search. These proposed object bounds are then rescaled to a fixed size image and fed into a CNN model trained on the ImageNet dataset to extract object features. Finally, linear support vector machine classifiers are used to predict object’s presence within each region [94]. Although RCNN improved mean Average Precision (mAP) from 33.7% to 58.5% when compared to DPM on VOC-07 dataset, RCNN was very slow. This led to subsequent improvements of accuracy and speed to develop Fast RCNN [104] and Faster RCNN [105] in subsequent iterations of the approach. Spatial Pyramid Pooling Networks (SPPNet) [106] and Feature Pyramid Networks (FPN) [107] are two more examples of two-stage detectors.

The one-stage detection propose boxes from input images directly without need for region proposal step used in two-stage detectors, thus making them more time efficient [108]. Although, two-stage detectors have high object recognition and localisation accuracies, they are less popular than one-stage detectors in real-time object detection tasks because they are slower. You Only Look Once (YOLO) is a popular one-stage object detector that divides the image into regions and predicts object bounding boxes and probabilities simultaneously using a single neural network [109]. YOLO significantly improved inference time of object detections, but suffered a drop in localisation accuracy, which was the focus of future revisions of the algorithm in [102] and [110]. Single Shot MultiBox Detector (SSD) introduced multi-reference and multi-resolution detection techniques to improve the accuracy of one-stage detectors when detecting small objects [111]. Two-stage detectors were consistently more accurate than one-stage detectors because of extreme foreground-background class imbalance encountered during the training of dense detectors according to [112]. This led to the development of RetinaNet, which used a new loss function termed 'focal loss' during training to help one-stage detectors attain similar performance to two-stage detectors without compromising their inference time [94].

One key constraint to adopting deep neural network approaches to object detection, especially on swarm robots platforms, is the computational requirements of these networks. Swarm robots are generally developed with minimal processing capabilities. This helps to minimise cost of developing swarms since there could be hundreds or even thousands of them deployed for solving swarm tasks. As the field of object detection using deep neural networks evolved, there has been some modifications to detectors in order to reduce their computational requirements and make them attain more frame rates on constrained computational devices such as embedded systems and mobile phones. Examples include the light-weight versions of SSD and YOLO known as MobileNet-SSD [113] and tiny-YOLO [114], respectively. These small sized networks can be run on devices such as Raspberry Pi at multiple frames per second. These networks generally have reduced performance when compared to their larger sized counterparts. However, their detection accuracies are reasonably good and because of multiplicity of sensing from multiple robots in the swarm, targets missed by one robot could be detected by other member(s) of the swarm.

2.6.3 Deploying deep neural network based object detectors on robots

The processes undertaken to develop and deploy deep neural network object detectors on robots are generally same for most robot types and application areas. For localised processing of visual information to detect targets, a robot needs a camera to sense its environment and enough computation power to process the information using a deep neural network algorithm. This is an essential requirement for both single or multi-robot system tasks. The only difference being that the model is deployed on one robot processor for single robot tasks, while for multi-robot systems, the detection model is deployed on each robot that make up the system. The steps for deploying deep neural network object detectors on robots are as follows:

1. ***Source for dataset***: Deep neural networks need a large number of representative dataset in order for them to generalise properly. Datasets can be sourced from online databases such as ImageNet [1], PASCAL-VOC [115], MS-COCO [116], which contain millions of images that cover thousands of object classes [117]. Search engines such as Google, Flickr and Bing have also made it easier to find relevant datasets for use. In addition, a number of researchers have made their datasets freely available online for the community of other researchers to use for benchmarking their work. Alternatively, individuals can generate custom datasets by taking photos (or crowdsourcing for photos) of the objects they intend to train their network on.

2. ***Annotate dataset***: Annotation is the process of specifying bounding box (or sometimes pixel masks) of the region occupied by objects of interest. Some datasets such as the PASCAL-VOC and MS-COCO are already annotated, so the researcher only needs to download the images and their corresponding annotations. However, some images such as those custom generated or found on ImageNet, will need to be annotated by the researcher using tools such as crowdsourcing on Amazon Mechanical Turk [118]. Alternatively, open source software such as CVAT [119] and Openlabelling [120] can be used to manually annotate the dataset, especially when they are not too much for a single (or few) individual(s) to do.

3. ***Training and testing***: After annotations, the next step involves dividing the dataset into training and testing data. Training data is used for updating the network's weights, while testing data is used to compute the performance (compute a loss function) of the network at the current iteration. Training process is usually performed on GPU enabled platforms, which significantly speeds up the process in comparison to it being done on a CPU [121]. During the training process, the accuracy of the detector improves over

time and it can be terminated based on some predetermined criteria. For example, the network can be trained until its loss function converges or until a maximum number of iterations is reached [111].

4. ***Deployment***: The trained network weights are saved and deployed on the computer that controls the robot [122]. Within the robot's control algorithm, the object detector network takes an image (from the robot's camera) as input, processes it and produces an output as reference as to where objects of interest occur within the input image (or information regarding absence of the target objects). Based on the results from the detector, the robot can then respond accordingly. Depending on the computation power of the robot's computer, there could be a mismatch between the rate at which the network receives input and outputs a result, and such issues need to be taken into consideration during the design. For example, the input to the network could be at 50 Hz, and because of low processing power of the robot, its corresponding output could be at 3 Hz.

2.7 Robots for Picking Litter

The job of foraging litter, within a park for example, is an interesting swarm task that can be extended to other foraging applications such as search and rescue in disaster scenarios, harvesting resources in an environment and demining an area. These tasks have a number of overlapping problems for the robots:

1. The number of targets to forage and their distribution within the search space are unknown.
2. The search area could cover a wide space which potentially makes it difficult to survey from a single location.
3. The terrain of the search space can vary from one region to another, and it could contain structures such as trees and other mobile and stationary obstacles.
4. The area could potentially be unfenced.
5. Central communication infrastructure could be unavailable.

While this list is by no means exhaustive, the point is that swarm robots are likely to face similar challenges when solving any of the tasks in a foraging scenario. So in this research, foraging litter has been chosen as a good real world application for swarm robots. Other inspiration for choosing litter include: alignment with the smart cities project where robots are deployed to manage city infrastructure, and foraging litter (in a park at least) is far easier to setup for the swarm without compromising the challenge level of the task at hand for the robot swarm.

Some research has been done in using robots to clean an environment. One of the earliest research that used multi-robot team for collecting litter was reported in [123], which won the Office Cleanup event at the 1994 AAAI Robot Competition and Exhibition. In their system, robots searched independently for litter within the search area and identified litter and other features using a camera. In the multi-robot system, robots avoided each other by analysing their visual feed and avoiding regions that have other robots within view. The approach developed separate reactive behaviours that accomplish specific tasks (such as get litter, find litter, move the litter etc) that were triggered in appropriate sequence by a behaviour manager. DAVID was developed in [124] for cleaning up offices in addition to collecting and delivering mail and stationary. The system used a single robot to forage tennis balls within a partially mapped office environment. In [125], the design and experimental results of using DustCart for door-to-door refuse collection within an urban environment was presented. DustCart was able to safely navigate round Peccioli, a small town of Tuscany in Italy, using beacons in its environment to localise its position and interact with users to collect their refuse and deposit the waste at a designated area. The robot was managed by a centralised Ambient Intelligence system (AmI) through a wireless network. In [126], the DustCart and DustClean (designed for cleaning pedestrian areas) robots were equipped with Air Monitoring Module (AMM) for collecting environmental data used for evaluating air quality while performing their tasks. A number of studies have focused on the design of robot platforms for cleaning an environment [127, 128]; improving the obstacle avoidance behaviour of robotic vacuum cleaners [129] by using infrared line projections instead of relying on vision; use of fuzzy inference system on user input to self-reconfigurable floor cleaning robot to balance area coverage and energy consumption [130]; and the challenge of knowledge representation for an autonomous robotic platform for picking up litter [131]. In a recent work, a single robot programmed with deep neural network based object detection was employed to explore an environment to locate and pick up litter in an outdoor park [132]. Their results indicate that the robot was able to pick up all litter quicker when it used a methodical approach to search for litter instead of random exploration of the environment. Within a swarm robotics context, urban waste management system where robots used a stigmergy-based approach to collect litter from bins and deposit them in multiple deposit (or nest) locations was developed in [133]. GAMA, the realistic agent-based simulation tool model of the Kendall area in Cambridge, MA, USA was used to test the feasibility of the swarm foraging system. Refuse bins were equipped with RFID tags for handling pheromone information and placed at intersections within the city. Robots explore the environment by navigating between refuse bins to empty them, deposit and read pheromone information from RFID beacons, transport picked up litter bins at the nearest deposit location and proceed to the nearest recharge stations when running low on battery power. Their results show that the swarm system was efficient in managing the waste in the environment, with the use of multiple deposit sites producing better results than when robots transported the refuse bins to a central deposit location.

2.8 Swarm Foraging Metrics

Swarm foraging algorithms performance are usually ranked based on the designer's desired criteria. The algorithm's performance is also compared to what is attainable by other competing algorithms. The following subsection outlines some metrics measured in swarm foraging literature and algorithm baselines that will be used to analyse the algorithms proposed in this thesis.

2.8.1 Metrics measurement

Measuring the performance of swarm foraging algorithm is important for determining how accurate its design objectives are realised. Several studies in swarm foraging focus on estimating the amount of targets foraged (either the time taken to forage certain percentage of the targets or fraction of the total targets that were foraged within a specific time period) [57, 63–66]. For foraging algorithms with primary emphasis on task allocation, the energy efficiency of the swarm [68, 72] and minimisation of inter-robot interference [70] are common metrics used to measure the algorithm's performance. New metric such as the information cost reward framework have also been proposed to analyse the underlying mechanisms of recruitment that are suitable for swarm tasks [134].

In addition to foraging performance, swarm specific features such as robustness and scalability are also important indicators of an algorithm's performance. Robustness is usually measured based on analysing the swarm performance under different target distribution [56], noisy or imperfect recruitment information [67] and varying the swarm size [56, 66, 67]. Scalability of the swarm can be measured under the context of whether computational requirements of the system increases with swarm size or how a change in swarm size affects foraging efficiency. For example, in [60], scalability was defined as the relative improvement of foraging efficiency of the swarm in comparison to when a robot acts alone, where efficiency is a measure of the contribution of one robot to the total number of resources collected by the swarm.

2.8.2 Baseline algorithms

It is common practice to compare swarm foraging algorithms with a scenario where robots in the swarm act independently without communicating or recruiting other swarm members [56, 65, 67]. Another approach, used in [63], was to compare multiple variants or extensions of a base algorithm in order to analyse the effects of iterative improvements to the algorithm. In [61], the approach used was to measure the proposed algorithm's foraging performance with two extremes: a scenario where robots foraged using random walk without recruiting other swarm members; and a setup where location of targets to forage were known *a priori* by the swarm. This is the approach adopted in this thesis since it is able to clearly demonstrate the improvement gain achieved by the swarm, and makes it easier to estimate the closeness of the swarm algorithm to a centrally coordinated swarm system which would produce optimal results.

The centralised swarm coordination system used in this thesis is the Adaptive Large Neighbourhood Search (ALNS) heuristic proposed in [135]. ALNS is a very effective route optimisation algorithm used to solve the problem of handling multiple transportation requests using a limited number of vehicles. Each request has a pick up and drop off location, and each vehicle is limited by the size, weight or quantity of goods it can carry. The aim of the algorithm is to construct routes for the vehicles such that all pick up and drop off locations (known as nodes) are arranged along the same route. The resulting route followed by the vehicle is such that pick up tasks are performed before corresponding deliveries. The algorithm starts by constructing routes that connect vehicles to all the pick up and drop off tasks. The ALNS algorithm controls which heuristics are used to remove and insert nodes that make up a vehicle's route. The steps that make up the ALNS algorithm include:

1. *Request removal:* ALNS implements three removal heuristics, which are Shaw, worst and random removal. The Shaw heuristic selects routes to remove based on their similarity; the worst heuristic focuses on removal of the worst routes; and the random heuristic randomly selects a specified number of routes to remove.

2. *Inserting requests:* routes are updated based on either a basic greedy heuristic or a regret heuristic. For the basic greedy heuristic, the nodes that result in least cost are inserted first, while the regret heuristic performs a look ahead step to implement the route insertion. The regret heuristic overcomes the problem of leaving 'high cost'

insertion requests to the end by looking ahead for a specified number of iterations to decide whether an insertion request is a good choice in the long run.

3. *Choosing a removal and an insertion heuristic:* ALNS alternates between the different removal and insertion heuristics to result in a more robust overall heuristic. To select a heuristic, ALNS assigns weights to the different heuristics and uses a roulette wheel selection principle.

4. *Adaptive weight adjustment:* the weight of each heuristic is automatically adjusted using statistics from earlier iterations. This is achieved by keeping track of the performance score for each heuristic such that a high score corresponds to a successful heuristic. This is then used as the basis for weight adjustment.

5. *Acceptance and stopping criteria:* ALNS then uses a simulated annealing algorithm as the acceptance criteria for route removal and insertion requests. The algorithm stops when a specified number of iterations of the steps have passed.

Over time, the ALNS algorithm optimises the routes followed by the vehicles for the pick up and drop off tasks. When applied to foraging, ALNS uses the location of targets as pick up locations and the nest as the drop off location for each target. The algorithm then iteratively optimises the allocation of targets each robot in the swarm picks up when it leaves the nest with a goal of optimising the time spent by the swarm to complete the foraging task.

2.9 Gaps in Body of Knowledge

A crucial point that has been highlighted in this chapter is that, even though robotic swarms are inspired from nature, they do not focus on developing a perfect representation of the biological behaviour [64]. Therefore, the swarm robots foraging algorithms generally have significant differences with their biological counterparts. In addition, swarm robots foraging algorithms generally focus on a small aspect of the foraging goal. In some algorithms, the aim was to minimise interference in the swarm through path decongestion [73, 74] or task allocation [70–72]. Some other algorithms placed emphasis on using response threshold model to achieve a specific swarm goal such as maintaining an energy (food) level for the swarm [69] or responding to the amount of food in the environment [68]. These algorithms were inadequate for studying the social interactions of robots in the swarm and the resulting impact it has on the swarm's ability to locate and retrieve targets within different target distribution scenarios and world sizes. There are many algorithms that considered swarm social interaction, where robots communicated with other swarm members through the nest

[61, 62], shared memory [54, 55] or robot-to-robot communication [56–59], to improve the collective behaviour of swarms during foraging. Some of these algorithms have also been implemented in hardware [60, 123] as a validation technique. The promising results thus far together with further improvements is a strong indication that deployment of swarms for real world problems is a realistic achievement in the near future. However, the handling of multiple simultaneously communicated signals is yet to be accounted for. For example, in [57] and [60], robots were constrained to selecting one out of multiple foraging sites, and their choices were not based on the integration of the information about the multiple sites. The stigmergy based algorithms in [64] and [66] do not scale well with increase in search space size as pointed out in Section 2.5.1.

The review also revealed that minimal work has been done on studying the effects of imperfect sensing and communication on the collective behaviour of swarm robots. The implementation of the swarm algorithms on robot hardware platforms served as a good indicator of how these sensing and communication uncertainties impact the deployment of algorithms on real world environments. However, the state-of-the-art in hardware realisation of swarm foraging algorithms simplify the communication and vision system used by the robots. So they do little to reflect the challenges the swarm will face when deployed in real world environments.

From the body of literature reviewed in this chapter and to the best of my knowledge, at the time of writing, the state-of-the-art in swarm robots foraging has not addressed the following issues:

1. A simplified swarm coordination system that takes advantage of real world physics to handle communication with multiple neighbours simultaneously.

2. An extensive analysis of the impact of realistic noise level on the performance of swarm foraging robots and the implementation of a simplified filtering system to mitigate the effect of the noise. This also includes the impact of noise on swarm scalability and robustness.

3. Scalability of swarm foraging algorithm with increase in the size of the swarm's search space, including the common real world scenario where swarms are deployed in an unbounded search space.

4. The development of robot's vision system using the state-of-the-art of computer vision algorithms that also take into consideration a robot's vision processing power.

5. An extensive analysis of the impact of imperfect vision system or vision processing ability on the performance of a swarm of foraging robots.

2.10 Summary

This chapter has discussed foraging strategies found in biological and robotic swarms. The hardware technologies used to study recruitment on real robot platforms, and the existing research in the swarm robotics field that account for imperfections in communication and vision systems of robots to study their impact on swarm foraging algorithms are also described. The state-of-the-art in computer vision (which had been virtually ignored in the swarm robots foraging research community) was also reviewed, owing to the fact that vision plays a vital role in helping swarm robots handle real world visual scenes, such as the tasks and challenges robots are expected to face when foraging litter in an environment. Techniques used by researchers to assess the performance of swarm foraging algorithms formed the conclusive remarks detailed in this chapter.

To address the gaps in the body of knowledge, this thesis proposes a new foraging algorithm, termed Repulsion-Attraction (RepAtt) algorithm. RepAtt is introduced n Chapter 3 and describes, in depth, the social interaction that improves the foraging efficiency of a swarm of collective robots. RepAtt significantly simplifies the search algorithms used by the swarm and hardware features necessary to realise the algorithm on real robots. By selecting this simplified solution, RepAtt is capable of solving the problem of concurrent communication with multiple robots in the swarm, efficient scalability with increasing search space, adequate noise management in the communication channel and effective performance in imperfect robot vision systems. In addition, this thesis presents a novel and extensive study of the impact of noisy inter-robot communication and deep neural network based noisy vision model on the collective behaviour of a swarm of foraging robots.

Chapter 3: The RepAtt Swarm Foraging Algorithm

3.1 Introduction

The Repulsion-Attraction (RepAtt) algorithm proposed in this thesis was inspired by the chemotactic search of *C. elegans* nematode. The search behaviour was modified to fit a foraging scenario that involved both attractive and repulsive chemotaxis. One key modification is that the robots serve as the source of the two chemotactic (attraction and repulsion) signals that degrade with distance. Each robot in the swarm is also capable of sensing the intensity of the attractive and repulsive signals which it then uses to modulate its search behaviour. This chapter describes the algorithm development and presents the result analysis of its effectiveness for swarm foraging.

The features of the robot used in the simulation study are described in Section 3.2. The model for attraction and repulsion communication signal the robots use for RepAtt is introduced in Section 3.3, while the concept of the RepAtt algorithm is discussed in Section 3.4. Chemotaxis search behaviour of the robots is based on tumble probability multipliers and divisors. The effects of these parameters on the robot's movement are discussed in Section 3.5. Gazebo simulation platform setup used in all experiments are detailed in Section 3.6. A parameters search approach discussed in Section 3.7 was used to determine the chemotaxis parameters that offered the best foraging performance. The optimised chemotaxis parameters were then used to conduct foraging time analysis in Section 3.8, robustness studies in Section 3.9 and the analysis of RepAtt's scalability in Section 3.10.

3.2 The Robot Model

A robot model with minimal features was used for this study. The robot is equipped with:

- mechanisms for vision based detection of targets to forage;
- means of radially broadcasting and sensing attraction and repulsion signals;
- sensing obstacles in its path using front, left and right contact/bump sensors;
- wheel-based locomotion.

The robot's vision is controlled by detection distance (C_d) and field of view (C_{FoV}) which represent the linear and angular detection range of the robot, respectively. As $C_d \longrightarrow \infty$, the robot gains global target detection distance, and as $C_{FoV} \longrightarrow 2\pi$ the robot gains omnidirectional view of its environment. To detect obstacles in its path, the robot uses three contact sensors to sense the presence of obstacles on its left, right and front followed by a response based on the obstacle avoidance behaviour, which is described in Section 3.4. For locomotion, the robot uses two driving (left and right revolute) and two stability (front and back caster) wheels. In all simulations, the robot's movement was characterised by straight movements and rotating on the spot to change its orientation.

3.3 The Communication Model

To perform chemotaxis, agents need to sense the intensity of a signal that changes with distance. The nature of this signal change can be linear, logarithmic or exponential. In chemotaxis, the agent is concerned with any detected change in the intensity of the signal between two consecutive measurements as it explores its environment. This dependence on change in the signal's intensity, by extension, means that the most important feature of a signal that can be used for chemotaxis is how consistent the signal's change with distance is. Examples of scenarios that exhibit such properties include decrease in chemical concentration with increasing distance from source location, reduction in light intensity with distance, decrease in sound intensity with increasing distance and several electromagnetic signals such as Wi-Fi, Bluetooth and radio frequency.

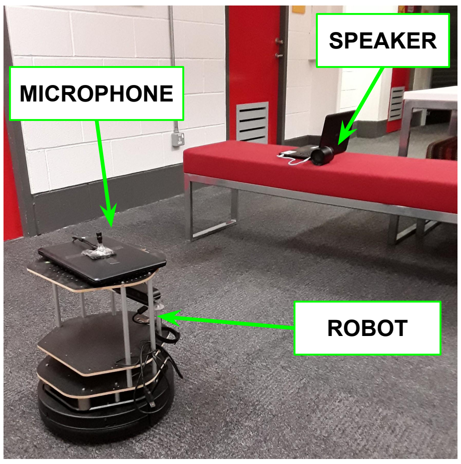
For the purpose of using hardware to inform the model, sound-based communication whose degradation was modelled as a negative exponential function of distance was used to develop the swarm communication mechanism. Sound was chosen because:

- it can be interfaced with the robots using readily accessible hardware (microphone and speaker);
- it does not require any 'coupling' of source and receiver devices (unlike technologies such as Bluetooth and Wi-Fi);
- it relies on environmental physics to handle the sensing of the resultant sound intensity from multiple signal sources;
- working with sound was easier for observing the experiments (for example, when using light, the experiments would have to be conducted in a dark environment, which made visual observations difficult).

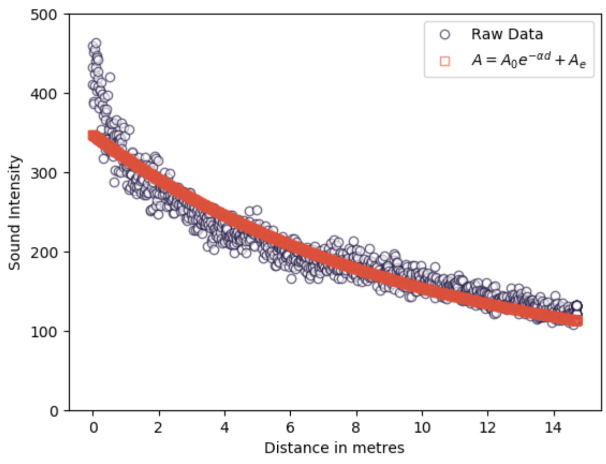
Equation 3.1, adapted from [136] (where a sound intensity bias parameter A_e has been added), was used to represent sound degradation as a function of distance. In Equation 3.1,

$A(d)$ is the sound intensity d metres away from the sound source, while α and A_0 represent the attenuation factor and sound intensity of the speaker respectively. Next, the steps taken to estimate the parameters of the sound model equation used in the simulation studies are described.

$$A(d) = A_0 e^{-\alpha d} + A_e \tag{3.1}$$



(a) Experiment setup



(b) Sound degradation

Fig. 3.1 An experimental test model of white noise sound signal intensity degradation with distance.

The experimental setup shown in Fig. 3.1a was used to collect modelling data. It involved using a Turtlebot2 robot equipped with an omnidirectional microphone to listen to white noise sound signal from a directional speaker. In this experiment, the Turtlebot2 was programmed to move away from the speaker at a velocity of 0.1 metre per second for 15 metres. It logged the sound data at 40 Hz and used odometry to compute the distance travelled. This process was repeated 5 times. The next step computes the unknown parameters of Equation 3.1. This was achieved by evaluating the least square error fit between the collected data and Equation 3.1 using MATLAB's non-linear curve-fitting function. The parameter estimates at the end of the optimisation process were $A_0 = 299.1793$, $\alpha = 0.1039$ and $A_e = 48.1824$. Fig. 3.1b shows the plot of 1000 data points sampled from the experimental data and their corresponding values when their distances were used to compute signal intensity using Equation 3.1.

To account for the total signal intensity in a situation where multiple sound sources are broadcasting sound, the linear sum of the intensities shown in Equation 3.1 was used. This is a reasonable assumption since our experiments show that multiple (white noise) sound sources always add up to give a higher amplitude. A sample waveform produced from an experiment involving two sound sources is shown Fig. 3.2. One source broadcast sound continuously for 130 seconds, while the second only broadcast sound for 20 seconds at intervals of 10 seconds. The peaks of the waveform occurred when the two sound sources were broadcasting at the same time, while the troughs occurred when only one sound source was active. The low intensity values at the beginning and end of the experiment represent the ambient sound level at the time of the experiment.

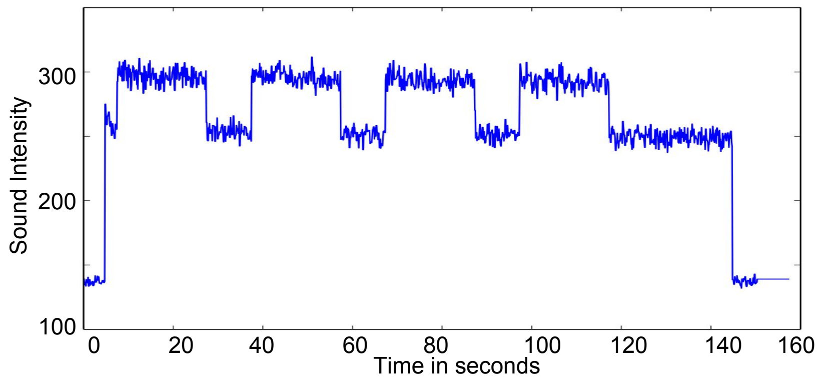


Fig. 3.2 Sound intensity output from two white noise sources.

The communication model described in the preceding paragraphs can be employed for swarm application where multiple robots broadcast distance-degrading signals radially to communicate useful information to neighbouring swarm members. Consider a scenario where robot i, located d_{ij} metres away from signal source j that broadcasts signal of type k, the subscripts and superscripts can be used to identify the specific signals. Equation 3.1 can be rewritten as Equation 3.2 when multiple robots are involved in the communication, where $\alpha = 0.1039$, $A_0 = 299.1793$ and $A_e = 48.1824$ are model constants. Thus the model assumes that signal intensity at the source and ambient sound level the robots sense are the same for all robots in the swarm and for all communication signal types used by the swarm. The total signal of type k at time t from multiple sources $j = 1,2,3,..,n$, where n is total number of signal sources, is given in Equation 3.3. The change in k signal intensity between t and $(t-1)$ which the robot i needs for chemotactic search, ΔI_i^k, is therefore Equation 3.4. For the RepAtt algorithm, only two signal types – repulsion $(k = r)$ and attraction $(k = a)$ – can be broadcast and sensed by the robots.

$$A_{ij}^k = A_0 e^{-\alpha d_{ij}} + A_e \tag{3.2}$$

$$I_i^k(t) = \sum_{j=1}^{n} A_{ij}^k \qquad \text{where } j \neq i \tag{3.3}$$

$$\Delta I_i^k(t) = I_i^k(t) - I_i^k(t-1) \tag{3.4}$$

In practice, when using sound-based communication (or some other media), the attraction and repulsion signals will occupy different frequency bands. For example, repulsion signal could occupy low frequency band, say from 500 Hz to 8,000 Hz while attraction signal could occupy 10,000 Hz to 18,000 Hz. This way, the listening robot can use hardware-based low-pass and high-pass filters to separate the two signal types so as to compute their gradients separately.

3.4 Algorithm Description

The main function of RepAtt in a swarm of foraging robots is to improve their coordination during the search for targets they forage to the nest. These robots can carry only limited quantity of targets and can only detect nearby targets which they search for using random walk algorithm. RepAtt improves robots' random walk search by equipping them with the ability to broadcast repulsion and attraction signals that other swarm members can use to perform chemotaxis-based search. Algorithm 1 is the pseudocode description of RepAtt algorithm executed by individual robots that make up the swarm. A robot's behaviour in RepAtt depends on whether it is in the obstacle avoidance, homing, acquiring or searching state. These states are shown in Algorithm 1 and further explained in the paragraphs below.

In **Obstacle Avoidance State**, robots avoid static (nest or walls) and dynamic (other robots) obstacles when any of the three front contact sensors get activated. When a robot bumps into an obstacle on either the left or right contact sensors, the robot changes its direction by 45 degrees rightwards or leftwards respectively. However, when the front contact sensor gets activated, the robot makes a random turn at an angle greater than 90 degrees. Upon changing its direction, the robot then makes a random linear motion between 0 and 1 metre before making the switch to searching, acquiring or homing state. The state the robot switches to depends on its local environment and the amount of space it has to carry more targets.

A robot enters the **Homing State** when its capacity, *cap*, is full. Within this state, the robot heads to the central nest to deposit all targets it has collected. For simplicity, it is assumed that robots know the direction of the central nest (which in practice can be a homing signal or nest beacon that is visible to all the robots within the foraging search space). In the homing state, a robot ignores attraction and repulsion signals from other robots until it successfully unloads all its foraged targets at the nest.

When a robot detects a target to forage within it sensing range, it enters the **Acquiring State**. In this state, the robot also ignores all communication signals from other swarm members, stops broadcasting repulsion signal, navigates to the nearest target and picks it

Algorithm 1 RepAtt Swarm Foraging Algorithm

```
1: Initialise Parameters: base tumble probability $P_b$, robot capacity $cap$, attraction multiplier $a_m$, attraction divisor $a_d$, repulsion multiplier $r_m$, repulsion divisor $r_d$, tumble mean $\mu$, tumble deviation $\sigma$
2: while true do
3:     if obstacle encountered then
4:         Obstacle Avoidance State
5:         Avoid obstacle on path
6:     else if $cap == 0$ then
7:         Homing State
8:         Go home and drop collected targets
9:     else if $found > 0$ then
10:        Acquiring State
11:        Go and pick up closest target
12:        if $found > cap$ then
13:            Broadcast Attraction $A_i^a$
14:    else
15:        Searching State
16:        $P_t = P_b$, $G_r = 1$, $G_a = 1$
17:        Broadcast Repulsion $A_i^r$

18:        if $\Delta I_i^r > 0$ then
19:            $G_r = r_m$
20:        else if $\Delta I_i^r < 0$ then
21:            $G_r = 1 \div r_d$

22:        if $\Delta I_i^a > 0$ then
23:            $G_a = 1 \div a_d$
24:        else if $\Delta I_i^a < 0$ then
25:            $G_a = a_m$
26:        $P_t = P_b \times G_r \times G_a$

27:        if rand(0,1) $< P_t$ then
28:            Make turn of $\theta \sim N(\mu, \sigma^2)$ radians
29:        else
30:            Make straight motion
```

up. During this process, the robot counts the number of targets within its visual range to determine whether to broadcast attraction signals or refrain from doing so. If the targets it detects exceeds its current carrying capacity, $found > cap$, the robot broadcasts the attraction signal, which other robots within communication range listen to. Otherwise, the robot refrains from broadcasting any signal.

Searching State is when a robot does not sense any target item to forage within its detection range. In this state, the robot listens to signals communicated by other swarm members, broadcasts a repulsion signal and executes random walk algorithm to search for targets. The random walk is controlled by a uniform distribution rand(0,1) and a tumble/turn probability, P_t such that if rand(0,1) $< P_t$, the robot turns a random angle. Otherwise, the robot makes a straight motion. The robot's goal within this state is to improve its chances of finding targets and minimise its visit to regions being explored by other robots. This is achieved by adapting the robot's tumble probability in order to minimise the repulsion (I^r) and maximise the attraction (I^a) signals it senses. Using the change in signal intensities the robot senses (Equation 3.4), the robot increases its turn/tumble probability when moving in the wrong direction (that is, when $\Delta I^r > 0$ or $\Delta I^a < 0$). On the other hand, when the robot is moving in the desired direction (that is, when $\Delta I^r < 0$ or $\Delta I^a > 0$), the robot reduces turn/tumble probability which in turn helps the robot to maintain its current direction (making longer swims). The turn/tumble probability, P_t, a robot uses for random walk in the search state is modulated by five predefined constants: base tumble probability P_b, repulsion multiplier $r_m \geq 1$, repulsion divisor $r_d \geq 1$, attraction multiplier $a_m \geq 1$ and attraction divisor $a_d \geq 1$. These are termed the 'chemotaxis parameters' and play significant roles in how the robots utilise the attraction and repulsion signals they sense. The special case where $r_m = r_d = a_m = a_d = 1$ gives the random walk algorithm with constant turn probability, P_t. In such a scenario, the robot does not use the attraction and repulsion signals it senses to improve its search strategy, thus making communication unnecessary. In other cases where $r_m > 1$, $r_d > 1$, $a_m > 1$ and $a_d > 1$, the robot's turn probability, P_t, will vary based on gradients (ΔI^r and ΔI^a) sensed by the robot. How the values of the chemotaxis parameters impact robot's behaviour will be explained in more detail in Section 3.5

3.5 Chemotaxis Parameters

As a robot explores the environment when in search for targets, it constantly updates its attraction and repulsion gradients (ΔI^a and ΔI^r respectively). The chemotaxis parameters play an important role on how the robot updates its tumble probability, P_t, in order to benefit from the gradient information it computes. The a_m and a_d are chosen based on attraction gradient ΔI^a, while r_m and r_d are chosen based on repulsion gradient ΔI^r. Furthermore, a_m and r_m increase tumble probability (P_t), thereby resulting in the robot making more frequent turns, while a_d and r_d decrease P_t and helps the robot make longer straight motions (longer

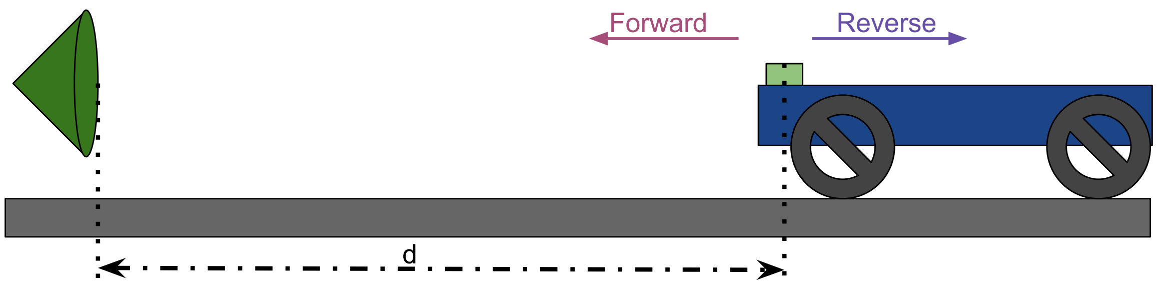


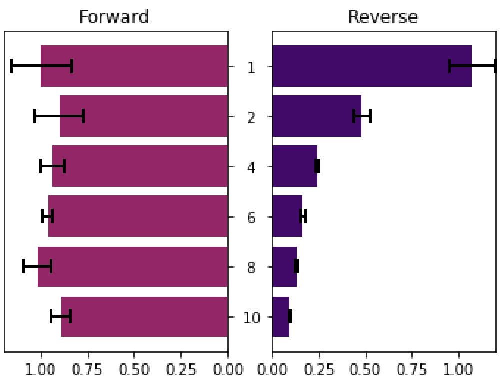
Fig. 3.3 One-dimensional Environment

Algorithm 2 One-dimensional Chemotactic search for an attractive signal source

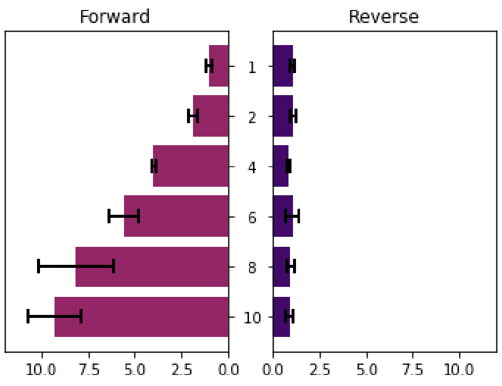
```
1: Initialise Parameters: tumble probability $P_b = 0.0025$ per time step, $a_m$ and $a_d$
2: while true do
3:     $P_t = P_b$, $G_a = 1$
4:     if $\Delta I^a > 0$ then
5:         $G_a = 1 \div a_d$
6:     else if $\Delta I^a < 0$ then
7:         $G_a = a_m$
8:     $P_t = P_b \times G_a$
9:     if rand(0,1) $< P_t$ then
10:        Reverse
11:    else
12:        Forward
```

swims). To understand the effect of these 'parameters', consider a robot in a one-dimensional world shown in Fig. 3.3, the robot can only make forward and reverse movements, d metres from a signal source that can either broadcast attraction or repulsion signal that degrades with distance. The signal peaks at the signal source, and degrades in intensity as $d \to \infty$.

To investigate how chemotaxis parameters individually affects the forward (toward signal) and reverse (away from signal) motion of the robot, a simple Gazebo simulation was conducted. In the experiment, a_m, a_d, r_m and r_d were individually varied from 1 - 10, while other parameters were set to 1. For this one-dimensional chemotaxis experiment, Gazebo time step was set to 25 ms, $P_b = 0.0025$ per time step and simplified version of Algorithm 1 shown in Algorithm 2. The results of the experiment is shown in Fig. 3.4, where each bar represent normalised mean path length for 30 independent repetitions of each parameter combination. Each bar was normalised using the forward motion mean path length for random walk (i.e. $a_m = a_d = r_m = r_d = 1$, which is the top left bar of each plot).



(a) a_m = vertical axis, $a_d = r_m = r_d = 1$



(b) a_d = vertical axis, $a_m = r_m = r_d = 1$

Fig. 3.4 One-dimensional robot simulation using chemotaxis to move toward attractive signal.

When the stationary signal source (Fig. 3.3) broadcasts attraction, forward motion of the robot increases attraction intensity sensed by the robot, thus resulting in $\Delta I^a > 0$, while reverse motion results in $\Delta I^a < 0$ because the robot moves away from the attractive source. In Fig. 3.4a, only the attraction multiplier, a_m, varied from 1 - 10, while $a_d = r_m = r_d = 1$. This had the effect of reducing mean path length for reverse motion of the robot, while forward mean path length remained fairly constant. Reverse motion was approximately equal to forward motion when $a_m = 1$, however, it reduced to about 10% of forward motion when $a_m = 10$. A reverse motion caused P_t to increase by a factor dependent on the value of a_m, which in effect helps the robot minimise its motion in the 'wrong' direction. As $a_m \to \infty$, the mean path length away from an attractive signal approaches 0 m, because the robot will more quickly tumble in order to orient itself toward the attractive signal.

In Fig. 3.4b, a_d was varied from 1 - 10, while $a_m = r_m = r_d = 1$. The effect, as illustrated in Fig. 3.4b, is a gradual increase in the mean path length of the robot as a_d increases, reaching a factor of approximately 10 when $a_d = 10$. Thus the mean path length toward signal source will approach infinity as $a_d \to \infty$ because a_d will increasingly suppress tumbles when robot moves forward, causing it to 'swim' more toward the signal source.

When the stationary signal source in Fig. 3.3 is changed to broadcast repulsion, forward motion increases repulsion intensity sensed by the robot ($\Delta I^r > 0$), while reverse motion decreases repulsion intensity ($\Delta I^r < 0$). The results indicate that when r_m was varied from 1 – 10, its effect mirrors that of a_m in Fig. 3.4a (that is, forward movements were reduced in proportion to a_m). Similarly, varying r_d from 1 – 10 was a mirror of the effect of varying a_d (that is, mirror of Fig. 3.4b, where reverse movements increased proportionately).

From the description above, the chemotaxis parameters can be divided into tumble enhancers (a_m and r_m) and swimming enhancers (a_d and r_d). Increasing tumble enhancers minimises the robot's movement in the wrong direction, while swimming enhancers maximise its movement in the desired direction. The effect of these parameters also depends on whether the signal gradients are positive or negative. Desired chemotactic behaviours can also be achieved by combining the effect of these parameters. For example, high values of a_m and a_d (and $r_m = r_d = 1$) will help in minimise movement away from attractive signal and maximise movement toward the signal, while the robot will remain unaffected by repulsion signals.

3.6 Swarm Foraging Simulation Setup

Algorithm testing was largely done using the Gazebo simulator – one of the most popular physics-based simulation platform for robotics research [137]. Gazebo was set to 25 ms time step to improve its speed for multi-robot simulations. Multiple distributions of 200 targets shown in Fig. 3.5 were used to test the effectiveness of the RepAtt foraging algorithm. For the Half50m and Half100m worlds, 100 targets were placed in a cluster, and the rest of the targets were randomly distributed in the 50 m $\times$ 50 m space around the nest. To ensure reliability of the results, each simulation was repeated 30 times. For each robot, the chosen simulation constants are: velocity of 0.605 m/s, time taken to process target during pick up is 5 seconds, number of targets a robot can carry is $cap = 5$, target detection distance $C_d = 3$ metres, omnidirectional robot field of view $C_{FoV} = 2\pi$ radians, base tumble/turn probability $P_b = 0.0025$ per time step, tumble angle mean $\mu = \pi$ radians and tumble angle deviation $\sigma = \pi/2$ radians. These parameters were used for all simulations unless stated otherwise. The swarm's task is to forage all targets in the environment, with a metric that measures the mean time it took the swarm to pick up 90% of targets used to evaluate the algorithms. Time taken to deposit targets at the central nest was not used to measure foraging performance because robots only returned to the nest when their capacity was full. This meant that with a $cap > 1$, there could be a trap where a robot never returns to the nest because it was unable to locate enough targets to fill its storage capacity. This could potentially be solved by implementing a 'give up' behaviour (similar to the one proposed in [61], where robots returned to nest if they were unsuccessful for a fixed time). The 'give up' behaviour could be useful for real-world foraging tasks such as picking litter, where robots need to: return to nest when the environment is clean, be recruited by successful forager robots, deposit foraged litter whenever they are near the nest but not yet full, and recharge batteries if their energy level reduces to some critical value. However, this was not implemented because it would involve searching for an optimal 'give up' strategy, which is could be task-dependent. Thus, time to pick up targets was used to measure foraging performance since it was an easier alternative for avoiding the 'trap' situation.

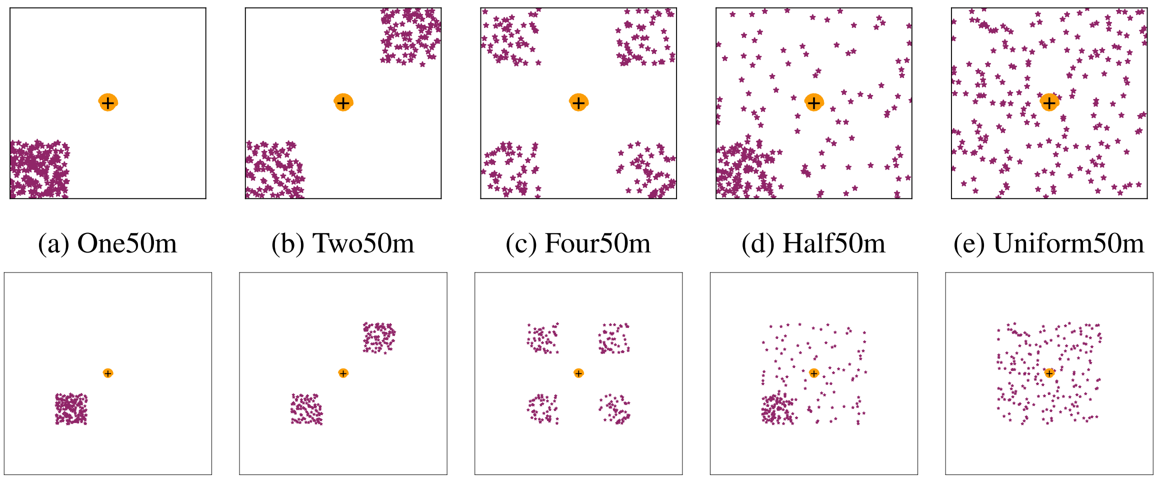
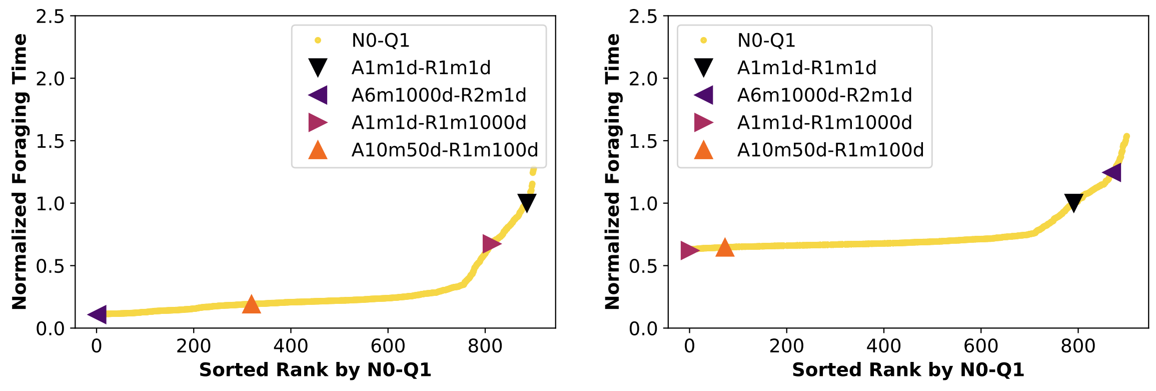


Fig. 3.5 Plot of initial target distributions, nest and robot locations for 10 test environments. (a) - (e) and (f) - (j) are 50m × 50m and 100m × 100m world boundary dimensions respectively. The 200 targets to forage are purple, black '+' is central nest and yellow blob represent the robots.

The RepAtt communication model used in the simulations presented in this chapter is based on Equation 3.2. This model ignored noise in the communication channel and the robots updated the signal they sensed at every time step. This thesis differentiates this perfect communication channel from a noisy version (which will be introduced in Chapter 4) using the notation N0-Q1, indicating 0% noise and communication update at every time step. The choice of this notation will be discussed further in Chapter 4 where noisy communication is added to the simulation. The N0-Q1 notation has been introduced here in order to make it consistent with the subsequent chapters.

3.7 Parameter Optimisation

It is evident from Section 3.5 that chemotaxis parameters are essential to a robot's ability to use foraging information. These parameters can make or break the foraging performance of the swarm when using RepAtt. With this in mind, an extensive parameter search was conducted in Gazebo to investigate the influence of these chemotaxis parameters on the swarm within a foraging scenario to determine which combinations of a_m, a_d, r_m and r_d performed best for swarm of 36 foraging robots under the 10 world setups shown in Fig. 3.5. The multipliers a_m and r_m were selected from 1, 2, 4, 6, 8 and 10, while the divisors were selected from 1, 10, 50, 100 and 1000. This resulted in 900 different combinations of the



(a) One100m (b) Uniform50m

Fig. 3.6 Sorted foraging times normalised using the mean time for A1m1d-R1m1d for the corresponding world setup.

chemotaxis parameters (i.e. $6 \times 6 \times 5 \times 5 = 900$). The notation used to represent the specific combination of these parameters is Aa_mma_dd-Rr_mmr_dd. Each combination was repeated 30 times on the 10 world setups, giving $900 \times 10 \times 30 = 270,000$ total simulations performed to search for the best performing parameters for the RepAtt algorithm.

The goal in each simulation was for the swarm to pick up 90% of 200 targets in the environment. The metric used is the time to complete these pick ups, with the combination having the least time ranked as the best. The performance of each of the 900 combinations was sorted according to average time from 30 simulation repetitions. The scoring system used is based on foraging time. The combination with smallest mean time was given a score of 1, while the longest received a score of 900. The total score for each combination was computed by summing their respective scores across the 10 world setups it was tested in, with the best parameter combination being the one that attained the lowest overall score. This approach ensures that only the parameter combinations that are able to perform well in both clustered and uniformly distributed environments are ranked higher than those that perform best in only specific world setups.

The resulting rankings for One100m and Uniform50m are shown in Fig. 3.6. These were chosen because they represented the world setups that took the longest and shortest times it took the swarm to complete the foraging task (rankings for all simulations are available in Appendix A). From Fig. 3.6, it is evident that the improvement on Random Walk (A1m1d-R1m1d) is much larger in One100m ($1 - 0.11 = 0.89$) in comparison to the improvement observed in Uniform50m ($1 - 0.62 = 0.38$). This indicates that appropriate selection of chemotaxis parameters is crucial in clustered environments, where resources/targets are more challenging for the robots to locate. On the other hand, when resources are abundant, robots have an equal chance of locating targets wherever they are in the environment (for example in Uniform50m). This in effect means that poor choice of chemotaxis parameters can have a higher negative impact on the swarm's foraging performance as evident from Fig. 3.6b, where the worst parameter setting of A10m1000d-R1m1d took 1.54 times more time than Random Walk. The impact of communication when resources are scarce versus when they are abundant has already been discussed by other researchers [57], and Fig. 3.6 is consistent with what has been reported in swarm literature. It is, however, important to note that chemotaxis parameters selection can also result in significant improvement in swarm's performance even when targets/resources are abundant. This is noticeable from Fig. 3.6b that an improvement of up to 38% $(1-0.62=0.38)$ can be gained over pure Random Walk (i.e when robots do not cooperate with other swarm members).

In the One100m world (Fig. 3.6a), A6m1000d-R2m1d were the parameter combinations that gave the best performance. This suggests that parameters that encouraged clustering of robots during chemotaxis (a_m and a_d) helped the swarm to exploit the cluster of targets to minimise foraging time, especially when the effect of repulsion was minimal (that is, r_m and r_d values were low). For the Uniform50m world (Fig. 3.6b), A1m1d-R1m1000d, delivered the best foraging performance. This suggests that combinations that helped robots maximise their path length when moving away from repulsion signals ($r_d >> 1$), consequently aiding dispersal in the process, was good for the swarm in this uniformly distributed environment setup. In addition, Fig. 3.6b suggests that parameters that increased robot tumbles (a_m and r_m) did not help improve swarm foraging performance in Uniform50m world. The combinations that resulted in the lowest score across the 10 world setups was A10m50d-R1m100d. This is interesting because it indicates that the combinations that performed well in all the tested distributions were those that can aid robot attraction to rich resource locations ($a_m > 1$ and $a_d > 1$) and helped swarm dispersal ($r_d > 1$). It is clearly the merger or integration of parameters that offered the best performance in clustered and uniformly distributed world setups. The paragraphs that follow explore in more details the influence of the chemotaxis parameters for the One100m, Uniform50m and also their combined effect across the 10 world setups.

The heat maps in Figs. 3.7, 3.9 and 3.11 present the foraging performance of the 900 different combinations of chemotaxis parameters in One100m, Uniform50m and combined rank for ten world setups respectively. Only the colours, and not the text in the cells, are needed for interpreting the heat maps in the figures. The heat maps for all the worlds are available in Appendix A. They have been grouped according to a_m (each row) and a_d (each column), where each heat map represents r_m (inverted y-axis) and r_d (x-axis) combinations.

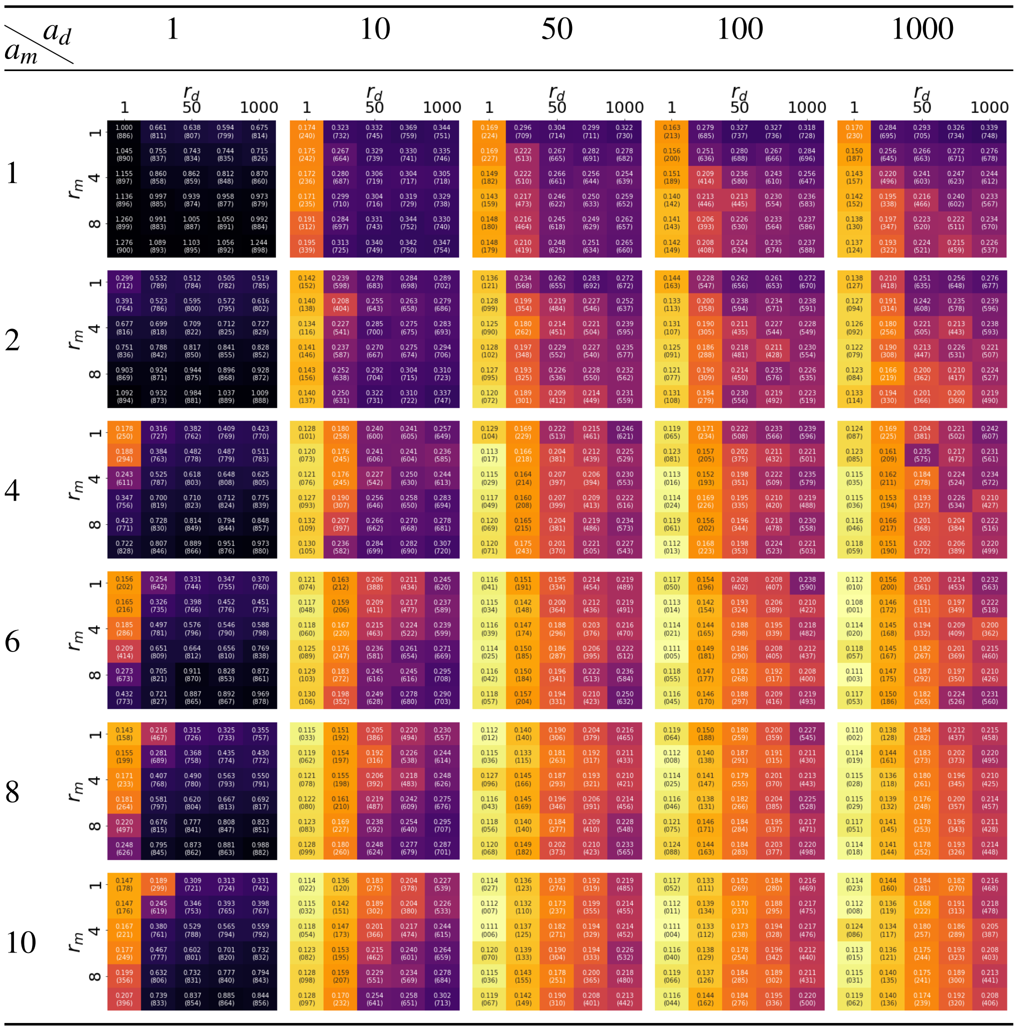
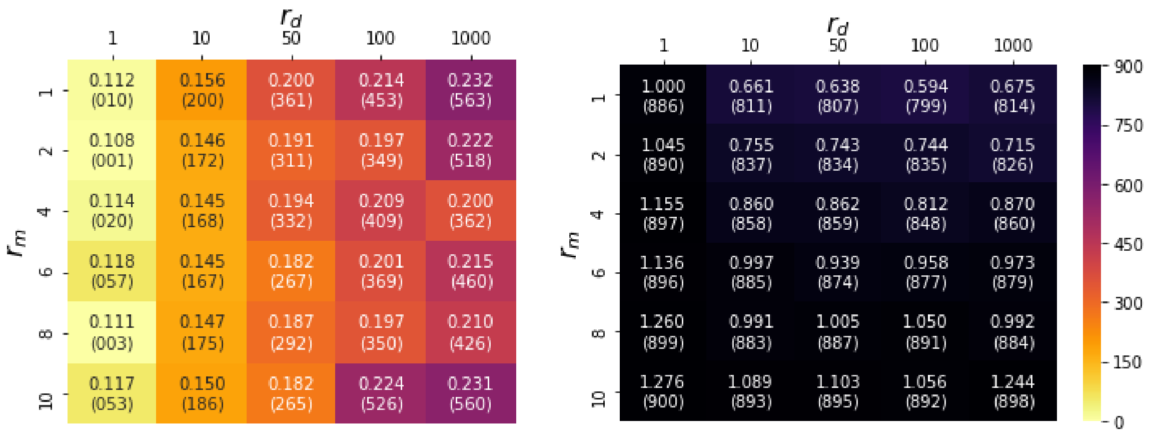


Fig. 3.7 Effect of the chemotaxis parameters on the foraging performance of a swarm of 36 robots foraging 180 out of 200 targets in the One100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 2$, $r_d = 1$.

The colour of each cell indicates its rank, with brighter colours representing highest/best ranks and darker colours indicating lowest/worst combinations.

For One100m world setup in Fig. 3.7, as a_d increases (move from left to right column), more cells become brighter in the heat maps as well as for when a_m increases (move from top to bottom row). This suggests that for clustered environments, foraging performance is approximately directly proportional to a_m and a_d. However, observing the general trend of each heat map of Fig. 3.7, brighter cells occur on the left which become darker when progressing to the right side. This indicates that foraging performance is inversely proportional to r_d. The effect of r_m is inconclusive for large values of a_d, whereas for $a_d = 1$ and

10 it can be estimated that r_m is also inversely proportional to foraging performance. Using the logic explained so far, the expected best parameter combination would be A10m1000d-R1m1d for the One100m world. However, this combination was ranked 23rd with a mean of 0.114 ± 0.005 (0.005 as 95% confidence interval), which placed it in the top 2.5% of the 900 different combinations. The best ranked combination for the One100m world was A6m1000d-R2m1d with a mean of 0.108 ± 0.003, and represents only 5.3% difference in mean time when compared with A10m1000d-R1m1d. The enhanced view of the heat maps that presented the best and worst ranked combinations are shown in Fig. 3.8 for the One100m world setup results from the results of Fig. 3.7.



(a) $a_m = 6$, $a_d = 1000$ (b) $a_m = 1$, $a_d = 1$

Fig. 3.8 Enhanced view of the heat map highlighting the location of the best and worst parameter combinations in the One100m distribution. The annotations show the normalised foraging time with the corresponding ranking in brackets.

The Unform50m world setup is the type of environment that can easily be negatively impacted by communication [57]. Fig. 3.9 shows the heat maps of the 900 different combinations of the chemotaxis parameters. Unlike in the One100m world, Uniform50m is negatively impacted by increasing a_m, and to a lesser extent, increase in a_d. In addition, the data also reveals that increasing r_m also had a similar effect on the foraging performance (within each heat map, moving downwards results in the cells becoming darker). Therefore, for Uniform50m, foraging performance is inversely proportional to r_m. However, increasing the r_d parameter resulted in improved swarm foraging performance (cells generally become brighter when progressing from left to right in each heat map). Based on these observations, the expected best parameter combination for the Uniform50m is A1m1d-R1m1000d, which is indeed the case when considering the top-right heat map of Fig. 3.9. Fig. 3.10 shows the heat maps that contains the best and worst combinations of the chemotaxis parameters.

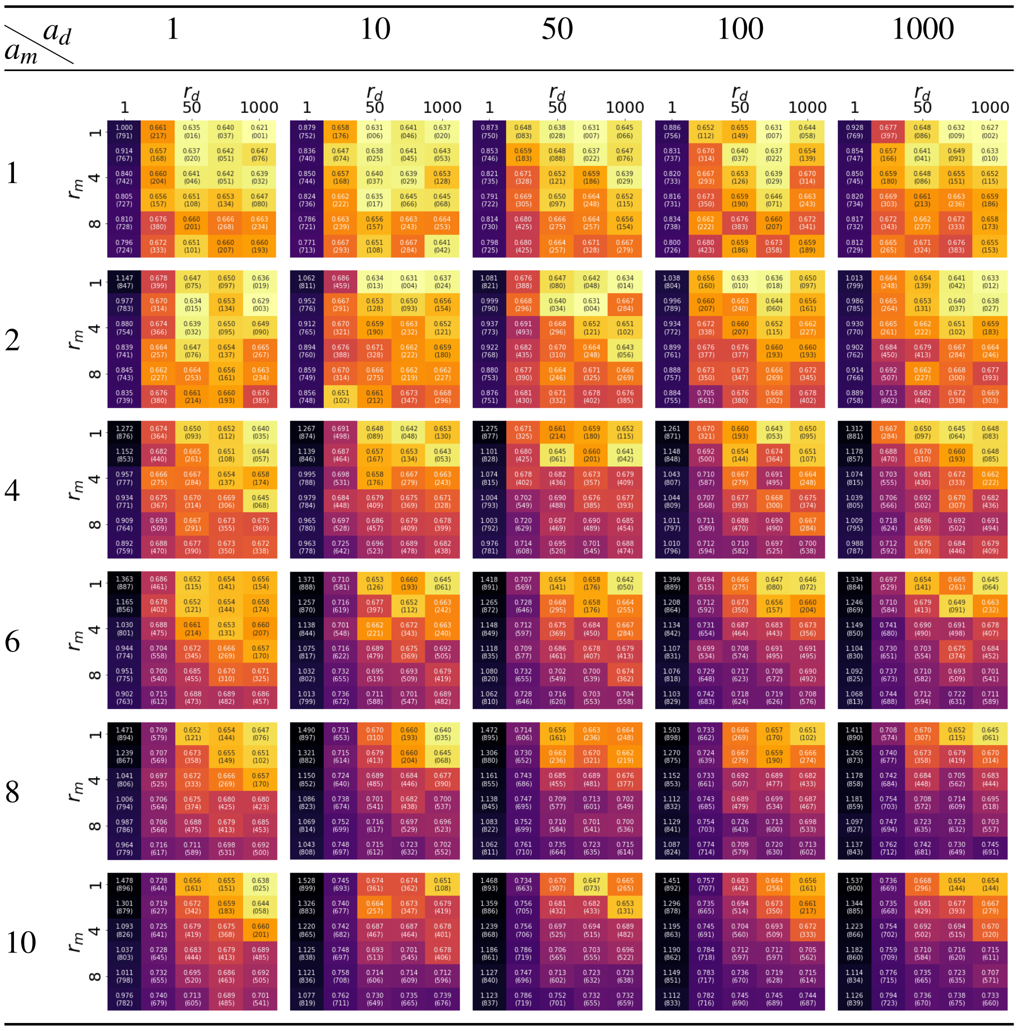
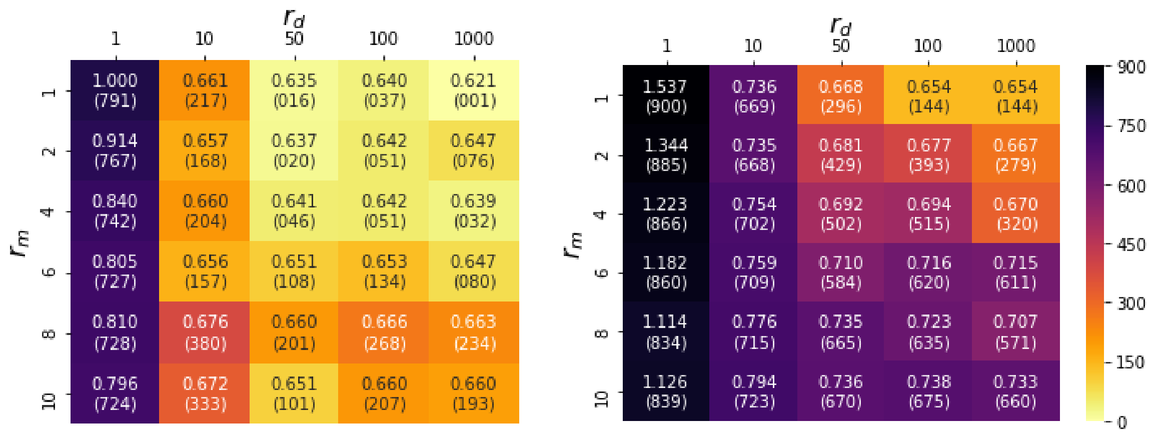


Fig. 3.9 Effect of the chemotaxis parameters on the foraging performance of a swarm of 36 robots foraging 180 out of 200 targets in the Uniform50m targets distribution. Best combination for this distribution is $a_m = 1$, $a_d = 1$, $r_m = 1$, $r_d = 1000$.

The result indicates that the worst combination is A10m1000d-R1m1d, which represents a combination that maximises the use of attraction information while ignoring repulsion signal that helps quick swarm dispersal and prevent robot clustering.

It is evident (and also intuitive) from studying Figs. 3.7 (OneCluster100m) and 3.9 (Uniform50m) that the best parameter combinations when targets/resources are clustered in the search space are generally the worst parameters for when targets/resources are uniformly distributed in the search space. This suggests that knowledge of the resource distribution within the search space is paramount for the setup of swarm parameters. In some applications, for example searching for source of gas leak, it is practical to assume or estimate the resource



(a) $a_m = 1$, $a_d = 1$ (b) $a_m = 10$, $a_d = 1000$

Fig. 3.10 Enhanced view of the heat map where the best and worst parameter combinations occur in the Uniform50m distribution.

distribution and thus the swarm designer can conveniently select parameters that optimise the swarm's search strategy. This in turn raises the question of whether a methodical-centralised search would be better than a decentralised swarm search strategy. In many practical applications for example, picking litter, planetary exploration, search and rescue, resource distribution information is not always available (and usually not easy to approximate), which makes it important to generalise swarm parameters design in order for the swarm to be robust (or easily adaptable) to differing resource distributions. This compromise to make RepAtt algorithm generally applicable for different resource distributions is shown in Fig. 3.11, where the chemotaxis parameters combinations are based on the scoring system described earlier in this section. The heat maps show that a_m and a_d are directly proportional to foraging performance. The effect of r_m suggests an inverse relation to foraging performance, however, this effect is not strong. For the r_d parameter, what is evident in most of the heat maps is that $r_d = 1$ is definitely not good for swarm foraging. As r_d increases, it appears to peak and then start to degrade swarm foraging performance as r_d approaches 1000. The resulting combinations that ranked highest across the 10 world setups is A10m50d-R1m100d, while A1m1d-R1m1d (Random Walk) gave the worst performance. The enhanced view of the heat maps showing the point of these occurrences is presented in Fig. 3.12.

In subsequent simulations in this chapter, $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ will be used as chemotaxis parameters for RepAtt algorithm because the combination resulted in the highest rank out of 900 competing options and 10 world setups. As a side note, it is entirely possible for a different parameter setup to perform best when foraging settings such

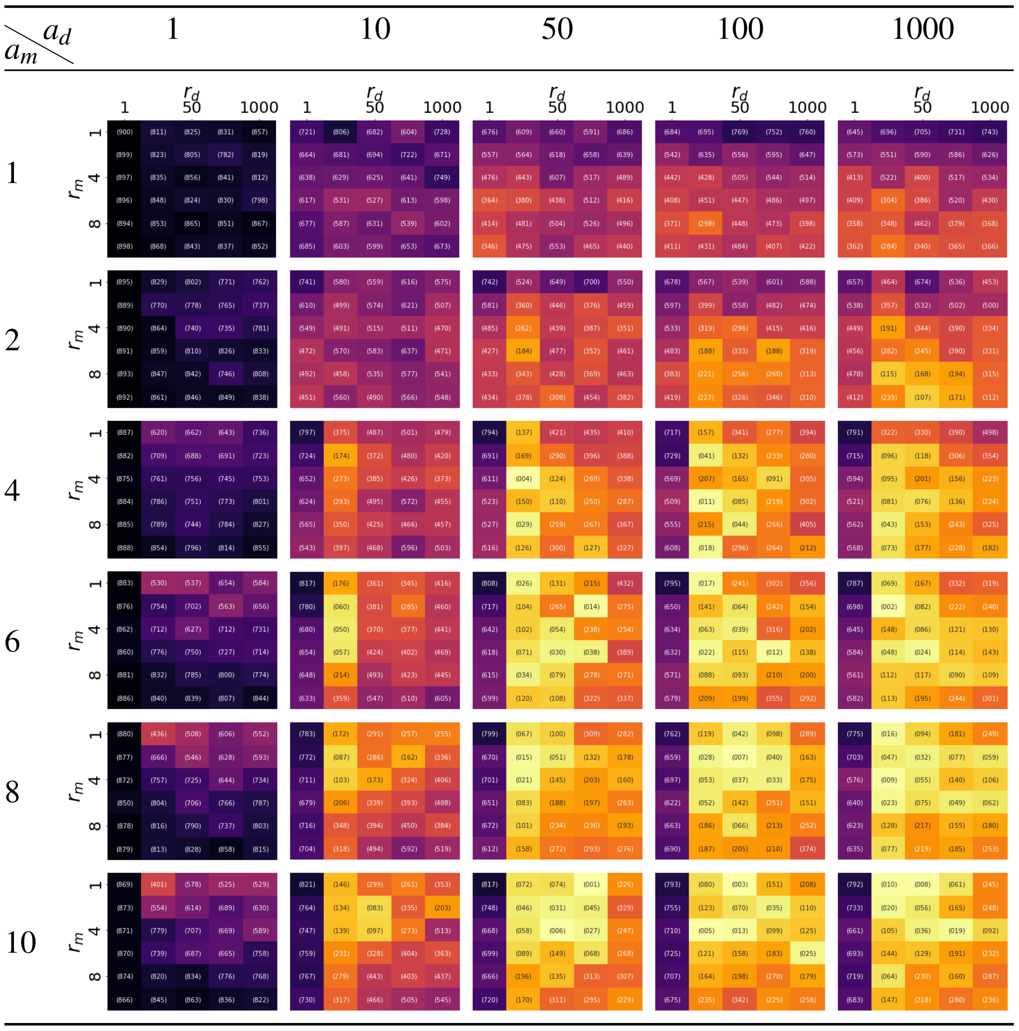
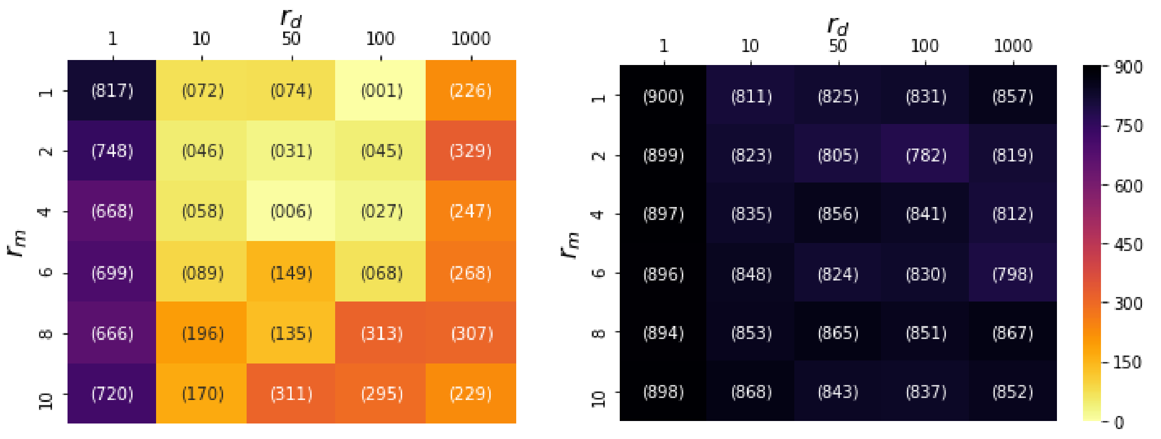


Fig. 3.11 Effect of the chemotaxis parameters on the foraging performance of a swarm of 36 robots foraging 180 out of 200 targets in the ten environment setups. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 1$, $r_d = 100$.

as tumble probability (P_t), robot capacity (cap) and so on change, however, these general concepts will still remain valid:

1. When targets are highly clustered and difficult to locate by a swarm, effective use of attraction signal will significantly improve swarm's foraging performance. Thus, large values of a_m and a_d parameters will play a major role in improving the swarm's performance.

2. In world setups where targets are abundant or uniformly distributed in the search space, each swarm member has a high probability of encountering/locating targets



(a) $a_m = 10$, $a_d = 50$

(b) $a_m = 1$, $a_d = 1$

Fig. 3.12 Enhanced view of the heat map where the best and worst parameter combinations occur when sorted according to rank scores across the 10 environment setups.

irrespective of its position. This is especially true in the early phases of robots foraging task. The communication of repulsion signals is the most useful information for improving swarm foraging in such environments. Large r_d parameter, which helps swarm dispersal, improves how the robots in the swarm use the repulsion signal during the foraging process.

3. To design swarm parameters that can perform well in both clustered and uniform environments, a selection of robot parameters that are good for these two extreme environments is the best approach.

3.8 Swarm Foraging Time Analysis

A subset of the simulations presented in Section 3.7 is to investigate the performance gain RepAtt gives a swarm of foraging robots across the ten different world setups in comparison to uncoordinated algorithm (Random Walk) and the centrally coordinated ALNS algorithm. The optimised route for the ALNS algorithm were supplied by Dr. Philip Kilby of the Australian National University. These routes were then applied to the swarm of 36 robots to guide the path they followed during the execution of the foraging task. The RepAtt algorithm, shown as N0-Q1 in the results in this section, used $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ chemotaxis parameters. Scatter plots showing the foraging progress of a swarm of 36 robots foraging targets in One100m and Uniform50m world setups are shown in Figs. 3.13 and 3.14 respectively.

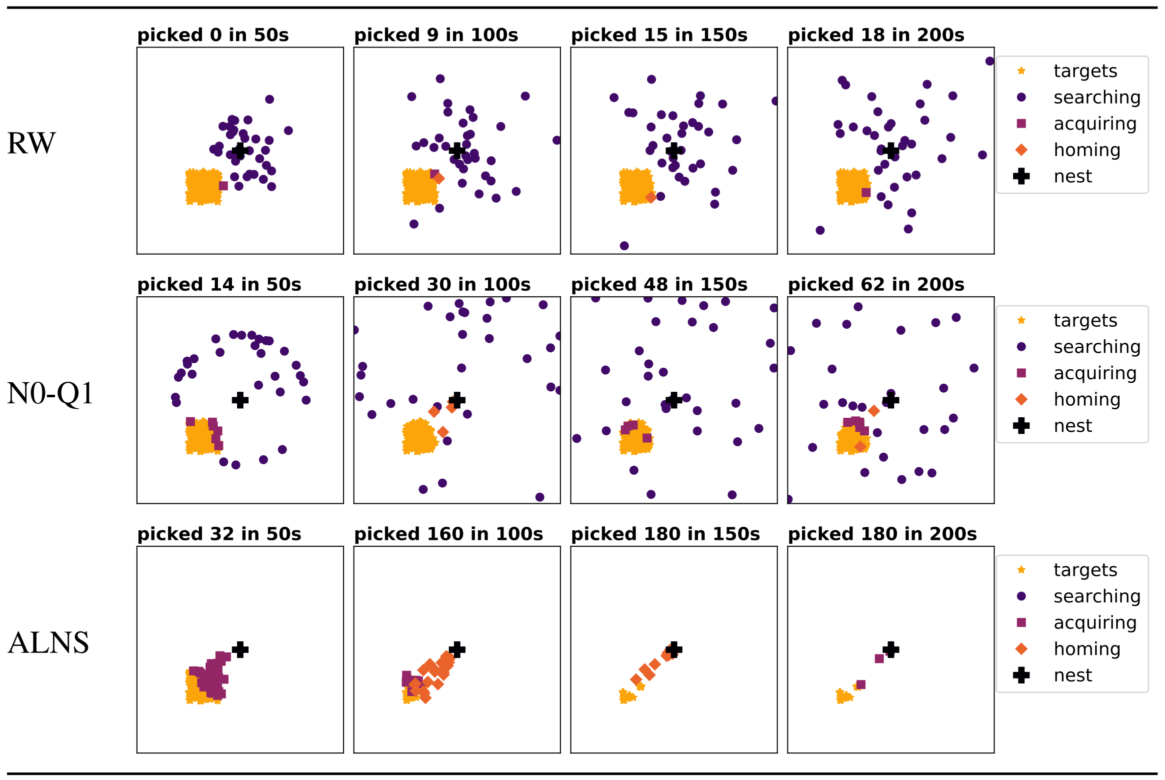


Fig. 3.13 Scatter plot of simulation progress of a swarm of 36 robots foraging 200 targets in the One100m world setup.

In the One100m world simulation (Fig. 3.13), the robots using Random Walk took a long dispersal time in covering the search area. The lack of communication also meant the robots did not recruit swarm members to target clusters they located. When the robots used N0-Q1, repulsion communication helped the swarm to disperse faster and locate the targets cluster. Once a robot locates the targets cluster, it broadcasts an attraction signal to recruit nearby robots to exploit the cluster, thus, increasing the number of robots that are around the targets cluster region. The ALNS algorithm eliminates the targets search phase. The robots executed the predefined paths they were programmed to follow to pick up their individual targets. It was therefore unsurprising that the swarm was able to pick up 180 targets within 200 seconds.

In the Uniform50m world simulation snapshots shown in Fig. 3.14, the robots did not spend much time searching for targets. This also means that, even with Random Walk, robots would have better success rates in locating targets than in the One100m world setup. The swarm using Random Walk was successful in picking up 150 targets within 200 seconds. The robots using N0-Q1 were slightly more successful than Random Walk because the repulsion

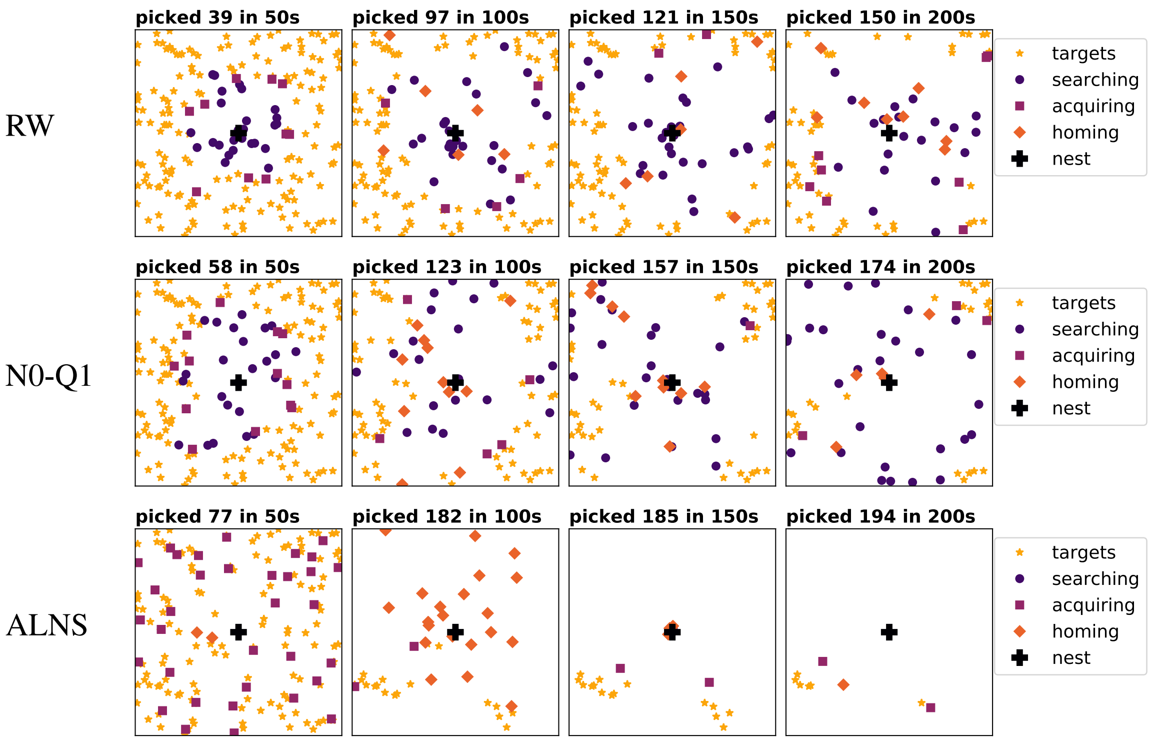
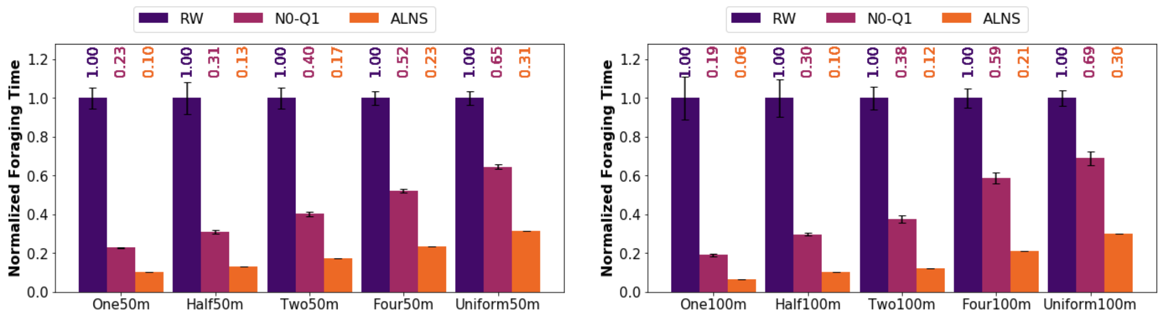


Fig. 3.14 Scatter plot of simulation progress of a swarm of 36 robots foraging 200 targets in the Uniform50m world setup.

signals played an auxiliary role in helping the swarm disperse within the 50 m $\times$ 50 m search space. When using ALNS algorithm, the robots were able to pick up 194 of the targets within 200 seconds.

The bar plot in Fig. 3.15 shows the average time taken to pick up 90% of targets by the swarm of 36 robots in the ten world setups. Each bar represents the mean value for 30 simulation repetitions, while the error bars are 95% confidence intervals. The data shows that, in comparison to Random Walk, RepAtt decreased foraging time by improving swarm coordination in all world setups. In the 50 m $\times$ 50 m world size (Fig. 3.15a), the improvement in foraging time was 77% in the One50m world, making it relatively closer to the ALNS's 90% improvement over Random Walk. Similarly, RepAtt significantly reduced swarm foraging time in the remaining four world setups of Fig. 3.15a with the weakest effect (of 35% improvement) in the Uniform50m targets distribution. In the 100 m $\times$ 100 m world size, the search space was quadrupled to make it more challenging for the swarm. Even in this situation, Fig. 3.15b shows that RepAtt was, again, positively impactful on the swarm's coordination, thus reducing foraging time by 81% (One100m), 70% (Half100m),



(a) 50m × 50m

(b) 100m × 100m

Fig. 3.15 Time taken by swarm of 36 robots to pick up 90% of targets for different world setups, normalised using the time taken by Random Walk.

62% (Two100m), 41% (Four100m) and 31% (Uniform100m) in comparison to Random Walk. This is in comparison with ALNS's values of 94%, 90%, 88%, 79% and 70% in the respective distributions.

The results reinforce the theory that coordination and communication have greater beneficial effect for highly clustered target distributions in situations where it is difficult for robots to locate the target deposits. This is the reason for larger performance gaps between Random Walk and ALNS in the one, two and half clustered target distributions and relatively smaller margins for the less clustered four clusters and uniform distribution worlds. Fig. 3.15 shows that with the inclusion of just communication and excluding complex localisation mechanisms for robots, the swarm's foraging time improves significantly.

3.9 Robustness of RepAtt

One key advantage of autonomy of individual robots that make up the swarm is their robustness to changes in swarm size and world setups. How swarm size affects task execution for RepAtt is covered in Section 3.10, while this section focuses on what effects changing the distribution of targets have on swarm's foraging ability (in terms of time taken to complete the foraging task). Fig. 3.16 is a box plot showing the variability in foraging time for N0-Q1, Random Walk and ALNS algorithms. Random Walk displayed the highest variability in Fig. 3.16, indicating that it is the least robust (or adaptable) to variation in target distributions and world sizes. Thus, the performance of Random Walk is highly dependent on the kind of problem, making it a more specialised solution that is not generally applicable to a wide variety of conditions. ALNS showed the least variability in target distributions, thus making it more generally applicable. However, ALNS requires a priori knowledge of the search space, which impacts its wider applicability. RepAtt displayed good performance across

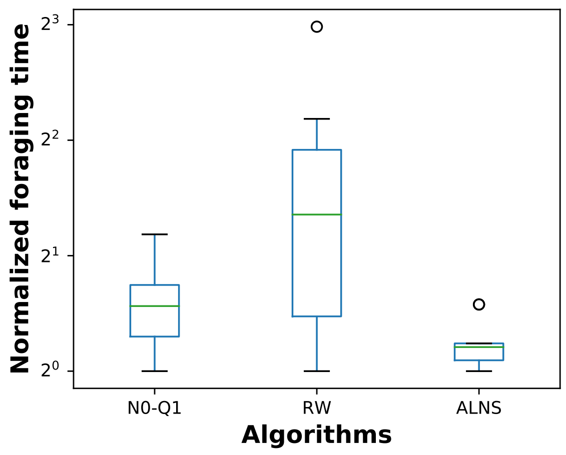
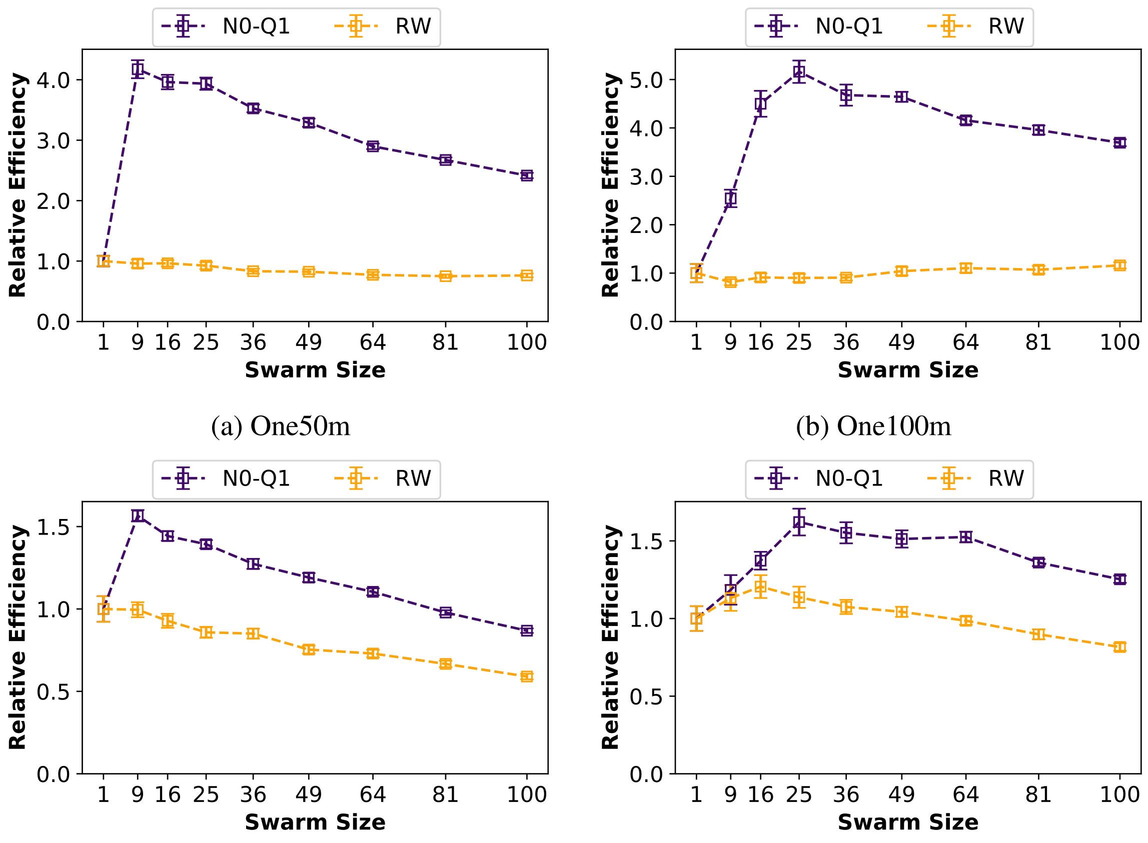


Fig. 3.16 Change in swarm foraging times across the ten target distributions. Foraging time was normalised based on distribution with shortest time, which was for the Uniform50m world, for the corresponding algorithms.

the different target distributions and did not result in any outlier when tested across the ten world setups. Foraging time variation across the ten world setups was 7.91 in Random Walk, 2.27 in RepAtt and 1.49 in ALNS. This implies that RepAtt was able to improve swarm's robustness by 71.25% in comparison to ALNS's 81.12%. This improvement is significant because it is quite close to what is attainable by a centrally coordinated algorithm with perfect knowledge of the environment, while RepAtt is a decentralised system that uses only local information of the environment to improve swarm foraging during task execution.

3.10 Scalability of RepAtt

Major advantages for using swarm robotics include its potential for improving the efficiency in accomplishing tasks that are too complex for a single robot. To test how RepAtt's efficiency scales with increase in swarm size, the efficiency improvement was computed as the number of robots varied from 1 to 100. For this foraging task, efficiency is defined as the time to pick up 1 target per robot. This is shown in Equation 3.5, where n is the swarm size, p is number of targets picked up, t_p is the time to pick up p targets. The relative efficiency, E_r, shown in Equation 3.6 is used to determine whether an algorithm improved the efficiency of completing a task in comparison to when the task was performed by a single robot. Thus, for $n = 1$, relative efficiency is $E_r = 1$, while $E_r > 1$ and $E_r < 1$ represent improvement and degradation in efficiency respectively.



(a) One50m (b) One100m

(c) Uniform50m (d) Uniform100m

Fig. 3.17 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.

$$E_n = \frac{p}{n} \times \frac{1}{t_p} \tag{3.5}$$

$$E_r = \frac{E_n}{E_1} \tag{3.6}$$

The effect of swarm size on relative efficiency for uniform and one cluster worlds is shown in Fig. 3.17, representing four out of the ten different test worlds (complete results are given in Appendix A). Results obtained using ALNS are not included because it would have involved performing the route optimisation for each swarm size and targets distribution. For the one cluster distributions (Fig. 3.17a and 3.17b), it is evident that RepAtt significantly improved foraging efficiency, with peak value of 4.17 (9 robots in One50m) and 5.16 (25 robots in One100m). However, in the same distribution, Random Walk was at best able to maintain relative efficiency at approximately 1. This is because, with Random Walk, the majority of the robots in the swarm never encounter the cluster of targets and spend the whole time searching for targets, whereas in RepAtt, the communication of attraction signals helped in directing more robots to the cluster to exploit the deposit. For the uniform distribution worlds (Fig. 3.17c and 3.17d), relative efficiency progressively degraded below 1 with increasing swarm size for the Random Walk algorithm, while RepAtt was able to maintain good efficiency improvement, especially in the Uniform100m world. The efficiency improvement from RepAtt was relatively lower in uniform distribution in comparison to the one cluster world setups, due to the lesser impact communication has when targets are easily locatable by swarms (as explained in previous sections).

In general, swarm efficiency is expected to drop as swarm size increase beyond some acceptable bound. This is due to factors such as increase in the interference between robots, size of search area and limited resources available for robots to forage. Excessively large swarm size cause robots to spend more time avoiding each other instead of focusing on the foraging task. Figs. 3.17a and 3.17c in comparison to Figs. 3.17b and 3.17d respectively show that increasing swarm size has a more negative impact on the 50 m $\times$ 50 m world size than on the 100 m $\times$ 100 m variant. This shows that increasing the search area reduces robot-to-robot interference, although it increases the number of robots required to achieve peak efficiency with RepAtt algorithm. Also, because the number of targets remained constant (pickup 180 out of 200 targets) in all the simulation results of Fig. 3.17, the number of resources (targets) does not increase to meet the need of the increased swarm population. This has the effect of increasing the number of robots that are unsuccessful in locating any targets, thus reducing swarm efficiency.

3.11 Summary

This chapter has introduced the RepAtt swarm foraging algorithm. A robot using RepAtt communicates with neighbouring robots by broadcasting attraction or repulsion signals depending on the presence or absence of targets within its sensing range. Hardware-based experiments were used to collect data for modelling the communication signal robots used in the simulations. Robots using RepAtt searched for targets using chemotaxis to move toward other companion robots within the swarm that broadcast attraction signals and/or move away from those that broadcast repulsion. The chemotaxis search is defined by a_m, a_d, r_m and r_d parameters, which control the robots' efficient use of attraction and repulsion gradients. Through extensive parameters search, the combination where $a_m = 10$, $a_d = 50$,

$r_m = 1$ and $r_d = 100$ produced best foraging performance for the swarm when considering the ten test environments. The simulations also indicate that RepAtt significantly improved swarm foraging performance in all the world setups, while also being robust to changes in targets distribution and in addition to excellent scalability with increasing swarm size.

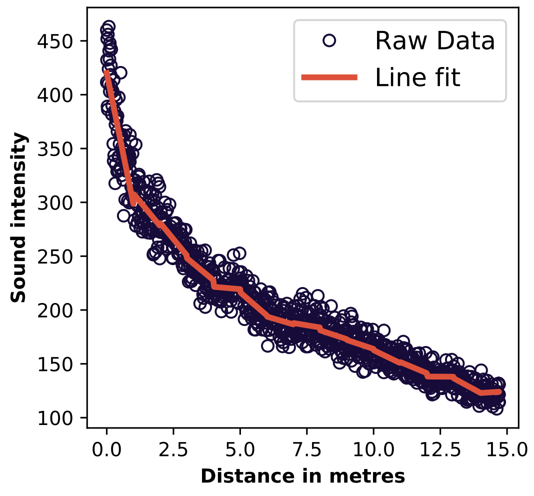
Chapter 4: Noisy Communication

4.1 Introduction

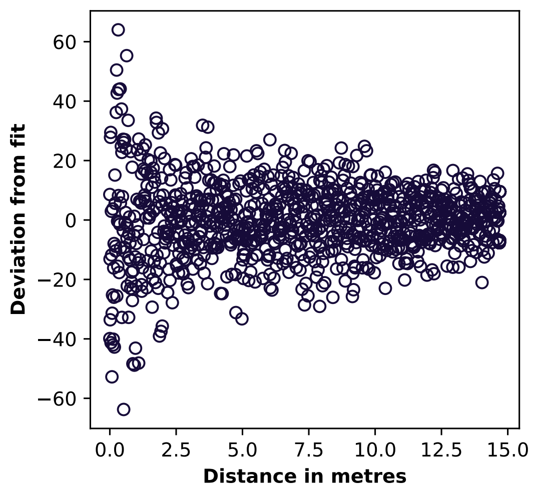
In the sound data experiments described in Chapter 3, specifically the results of Fig. 3.1b, the sound sensed by the robot as it approaches the speaker had significant variations such that the sound level increase was not consistent for data points that were near each other. The fluctuations can also be observed even when the robot was stationary as shown in Fig. 3.2. In this chapter, the variations in the sound model are quantified and modelled as noise in the communication signal that robots sense. The steps taken to mitigate the effects of the noise using a basic average filter (Section 4.2) will also be discussed. A new batch of chemotaxis parameter optimisation experiments were then conducted and presented in Section 4.3 to investigate which setups performed best under noisy communication. The effects of varying the filter queue size on swarm foraging is presented in Section 4.4 and the effects of noise on RepAtt's robustness and scalability are discussed in Sections 4.5 and 4.6 respectively.

4.2 Noise and Filter Modelling

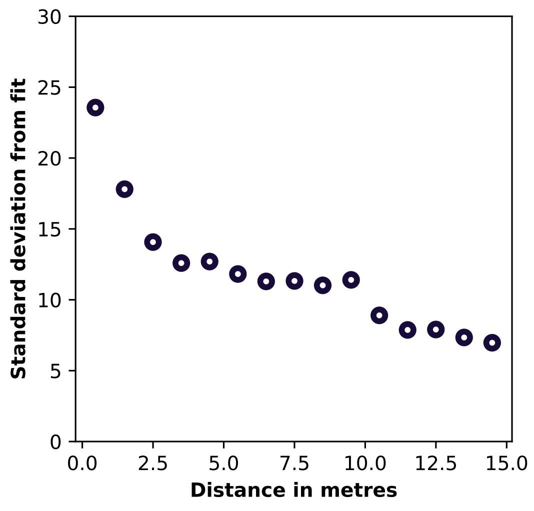
To quantify the noise level in the sound data sensed by the Turtlebot2 robot, the data from all five robot experiments were divided into segments corresponding to one metre of robot movement. A line-fit was then performed on each of these one metre segments as shown in Fig. 4.1a. This was followed by computing the noise in the data as the difference between the line fit segments and the corresponding sound data for each segment (see Fig. 4.1b). The observation from Fig. 4.1b is that as the sound level increases, the variation of the sound data from the line fit also increases. The next step performed was to compute the standard deviation of the noise level (Fig. 4.1c) and compute its ratio to the mean sound intensity for each segment (Fig. 4.1d). It is evident that, although the noise level increased with sound intensity, the ratio of the deviation to the corresponding sound intensity remained fairly constant at about 0.06, thus indicating that the noise level is proportional to the sound intensity. A normal distribution was then used to model the noise component of the signal, such that the mean, $\mu = 0$, and standard deviation, $\sigma = 0.06$. Equation 4.1 represents the noisy version (B_{ij}^{k}) of the swarm communication model, where A_{ij}^{k} is the noiseless model in Equation 3.2 used in Chapter 3 and x is a percentage used to scale the modelled noise. The x term has the effect of simulating the quality of the sensing hardware used by the robot to sense attraction and repulsion signals: $x > 100\%$ means the noise is amplified (thus meaning



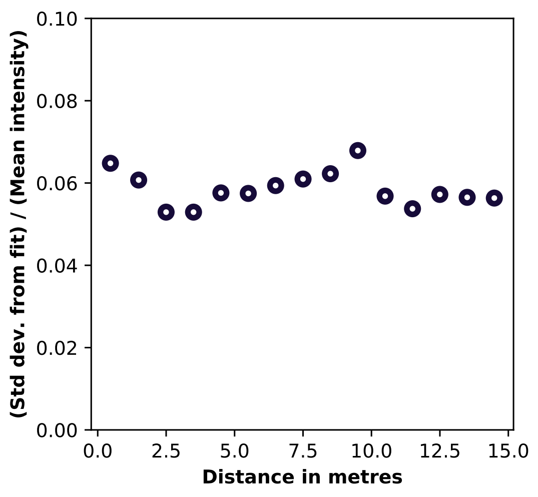
(a)



(b)



(c)



(d)

Fig. 4.1 (a) Line fitted to every one metre segment of the experimental data. (b) Difference between sound measurements and corresponding line in the segment. (c) Standard deviation of the sound variations from the fitted line. (d) Ratio of the deviation of each segment to mean of sound data for the corresponding segment is fairly consistent.

a poor microphone in this sound modelling context), $x < 100\%$ means better hardware than the one used for the sound experiments, and $x = 100\%$ is the current noise model with $\mu = 0$ and $\sigma = 0.06$. Equation 4.1 refactors to Equation 4.2, which then gives Equation 4.3 when A_{ij}^{k} from Equation 3.2, μ and σ are substituted. Fig. 4.2 plots the resulting noisy sound model and data obtained using the Turtlebot2 experiments. This figure confirms a strong correlation between the variation in the model and the data obtained from the experiment.

$$B_{ij}^{k} = A_{ij}^{k} - xA_{ij}^{k}N(\mu,\sigma^2) \tag{4.1}$$

$$B_{ij}^{k} = A_{ij}^{k}\left(1 - xN(\mu,\sigma^2)\right) \tag{4.2}$$

$$B_{ij}^{k} = \left(A_0 e^{-\alpha d_{ij}} + A_e\right) \times \left(1 - xN(0, 0.06^2)\right) \tag{4.3}$$

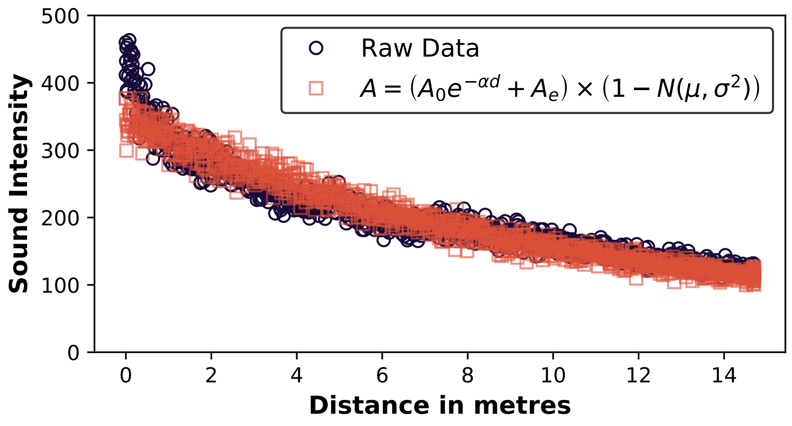
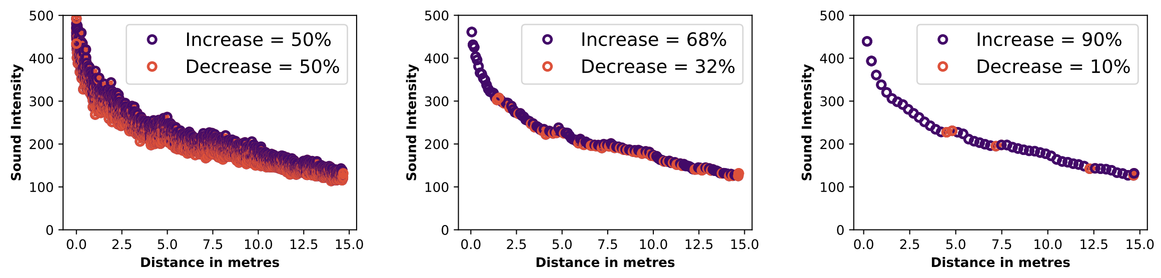


Fig. 4.2 The modelled noise data fits strongly with the level of noise expected from experiment data.

It is intuitive to expect the introduction of noise to have a detrimental impact on the effectiveness of any algorithm. It is, however, beneficial to include noise in order to study such impact and its negative effects to improve the algorithm's robustness so that it is able to withstand real world conditions. For RepAtt, the introduction of communication noise means that robots have lesser chance of accurately measuring temporal gradients. In Fig. 4.3a, using the sound data from the Turtlebot2 experiments, it can be seen that consecutive sound intensities have approximately 50% chance of increasing (purple markers) or decreasing (orange markers). This means that a robot sensing this signal for chemotaxis has a 50% chance of measuring a positive/negative change, which will be demonstrated in Sections 4.3 and 4.4 to have a significantly negative impact on swarm using RepAtt for foraging. A simple noise filtering process was introduced to mitigate the negative impact of this noise. The filter is based on computing the average sound intensity over a limited queue of multiple consecutive instantaneous measurements of sound intensities which the robot maintains. The robot uses this average of intensities as the current signal intensity level and compares this value with the preceding average computation to determine the intensity change of the signal. The notation Nx-Qy is used to represent x% of modelled noise value and y consecutive instantaneous sound measurements in the queue maintained by the robot (that is, queue size) for computing average intensity. Substituting B^k_{ij} into Equation 3.3 and extending it to compute the mean of y consecutive sound measurements gives Equation 4.4. Equation 4.5 is used for computing the change in signal intensity while accounting for the queue size.

$$
\begin{aligned}
I^k_i(t) &= \frac{\sum\limits_{b=t-y+1}^{t}\left(\sum\limits_{j=1}^{n} B^k_{ij}(b)\right)}{y} \qquad \text{where } j \neq i && (4.4)\\
\Delta I^k_i(t) &= I^k_i(t) - I^k_i(t-y) && (4.5)
\end{aligned}
$$



(a) Queue size $y = 1$ (b) Queue size, $y = 40$ (c) Queue size $y = 120$

Fig. 4.3 Queue size variation of the average filter. Increasing size of queue reduces number of negative gradient when traversing from 15m to 1m

The application of the average filter to the Turtlebot2 experiment data improved the fraction of transitions that were positive while moving towards a sound source. Increasing the filter queue size from 1 (in Fig. 4.3a) to 40 (in Fig. 4.3b) increased the positive transitions by 18%. With queue size of 120, a further 22% increment in positive transitions was registered. This shows that the average filter can help improve the percentage of correct gradient measurements by a robot. Fig. 4.4 shows a bar plot of percentage of positive and negative gradients for both the modelled noisy sound and data obtained from the Turtlebot2 experiments. The data indicates that as the queue size increases, the proportion of positive gradients increase at a similar rate, thus suggesting that the model is a good representation of the experimental data. Although increasing the queue size improves the accuracy of the gradient computation, a large queue size reduces the rate at which the robots in the swarm update the repulsion and attraction signals they sense. This reduced rate means robots are not able to react quickly to changes in signal intensities, thus causing them to react based on outdated information. An appropriate queue size is, therefore, important for helping robots respond quicker while being able to compute reliable gradients. Balancing update rate with the computation of reliable gradient will be discussed in detail in Section 4.4.

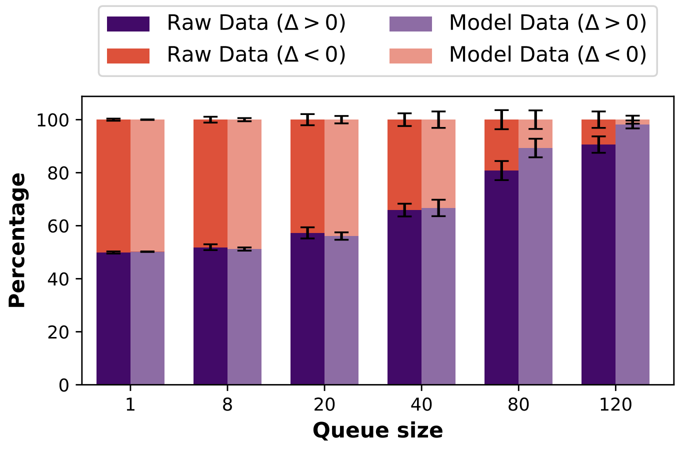


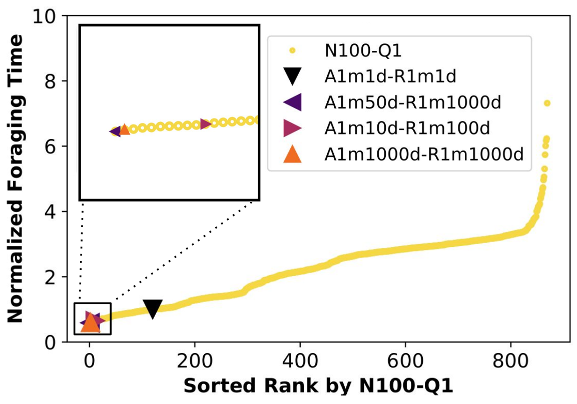
Fig. 4.4 Queue size versus percentage increase/change of the experimental (raw) and modelled data. Error bars represent standard deviation across the five experiments.

4.3 Parameter Optimisation

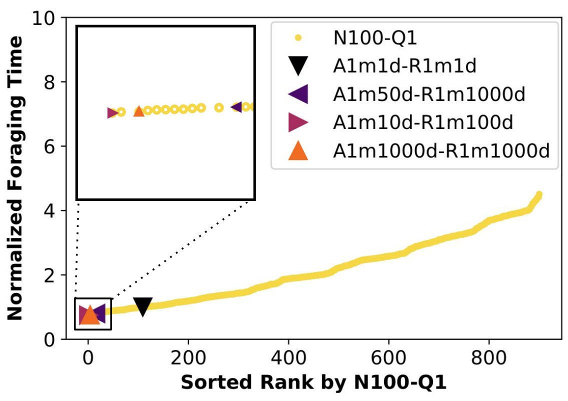
With the introduction of noisy communication, the chemotaxis parameters that offered good performance in Chapter 3 will not necessarily perform well. For example, when the A10m50d-R1m100d combination was used for foraging with noisy communication in One100m and Uniform100m world setups, RepAtt's performance degraded substantially. Table 4.1 shows the extent of degradation when noise was introduced to the RepAtt algorithm, where the times were normalised using the foraging time of Random Walk. This raises the question of whether this setup is best suitable for RepAtt in the noisy communication setup. To further investigate this, two sets of parameter search simulations were performed. In the first simulation test, robots used instantaneous measurements of the noisy communication signal during foraging (N100-Q1), while in the second set, the robots used an average filter with queue size, $y = 40$, (i.e. N100-Q40) to improve their gradient computation.

Table 4.1 The foraging performance of RepAtt when using chemotaxis parameters $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ for both noiseless and noisy communication. Each value represents mean and 95% confidence interval for 30 repetitions of each simulation.

	One100m	Uniform100m
N0-Q1	0.192 ± 0.007	0.690 ± 0.036
N100-Q1	0.946 ± 0.063	1.073 ± 0.029
N100-Q40	0.375 ± 0.021	0.782 ± 0.035



(a) One100m



(b) Uniform100m

Fig. 4.5 Sorted foraging times for 900 combinations of chemotaxis parameters normalised using the mean time for Random Walk for the corresponding world setup. The swarm of 36 robot foraged used noisy communication without any noise filtering system (i.e. N100-Q1)

The experiments for determining the best chemotaxis parameters for the N100-Q1 version of RepAtt were conducted for the One100m and Uniform100m environments. The normalised foraging time for the 900 combinations of the parameters is shown in Fig. 4.5, where the data was sorted from minimum to maximum. The top performing combinations were A1m50d-R1m1000d (for One100m), A1m10d-R1m100d (for Uniform100m) and A1m1000d-R1m1000d (based on the combined score of One100m and Uniform100m). Random walk ranked 109 out of 900, which represents a significant decrease in RepAtt's performance. In the One100m world setup, A1m50d-R1m1000d produced best performance with normalised foraging time of 0.59. Furthermore, many of the worst performing combinations did not forage up to 90% of the targets within the allocated two hours runtime for each simulation on the High Performance Computer by the job scheduler. For the Uniform100m, A1md10-R1m100d with a foraging time of 0.77 performed best while A8m10d-R10m1d was ranked last with a foraging time of 4.5. The data shows that there is a more significant difference between the best and worst ranked parameters in N100-Q1 in comparison to the parameter search for N0-Q1. In both tested environmental setups tested, N100-Q1 performed best when the robots minimised the effects of chemotaxis parameters that increased tumbles when moving in wrong direction (a_m and r_m), and only relied on the effects of a_d and r_d. This occurs because the noise essentially randomised the gradient computation (also obvious from Fig. 4.4 for queue size 1) from one time step to the other. As a result, this created a huge number of undesired tumbles when a_m and r_m are large, which in turn wasted time. By employing the scoring system used for the N0-Q1 RepAtt setup, the 900 combinations were

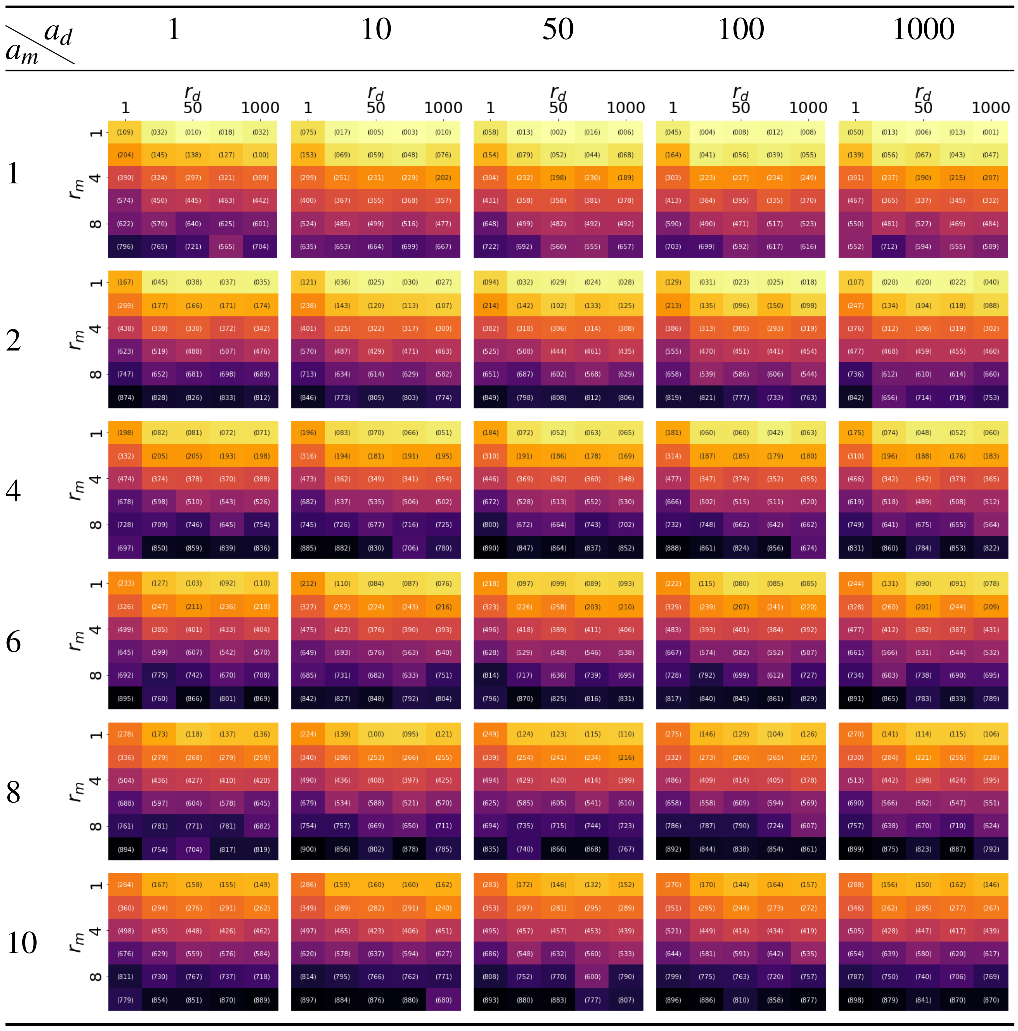
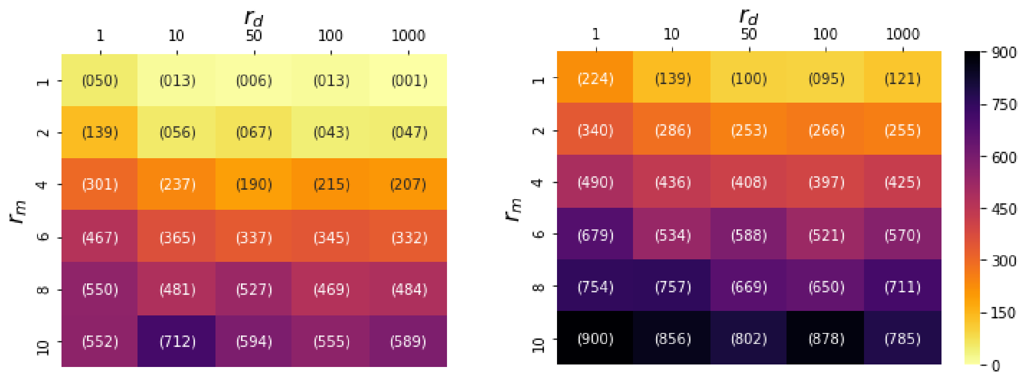


Fig. 4.6 Effect of the chemotaxis parameters on the performance of a swarm of 36 robots foraging 180 out of 200 targets in the One100m and Uniform100m targets distributions and using N100-Q1 version of RepAtt. Best combination from the scoring system is $a_m = 1$, $a_d = 1000$, $r_m = 1$, $r_d = 1000$.

ranked from lowest to highest based on their ranking in individual test environments. The rankings are shown in the heat maps in Fig. 4.6.

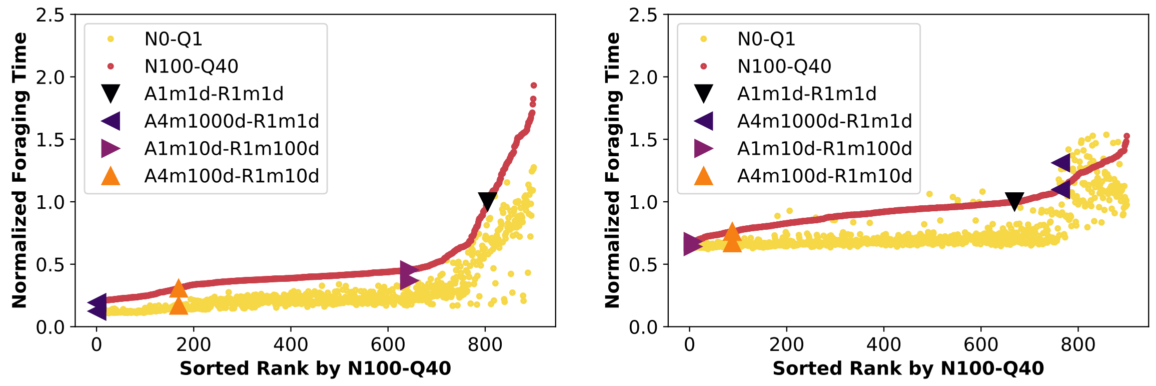
Table 4.6 illustrates how the chemotaxis parameters affect swarm foraging in both One100m and Uniform100m worlds (similar heat maps for each of the two worlds are available in Appendix B). The rows and columns represent a_m and a_d respectively, while within each heat map, r_m varies from 1 to 10 along the rows and r_d varies from 1 to 1000 along the columns. The data are colour graded by rank such that higher ranked cells are brighter, while poorly ranked cells are darker. Only the colours are needed for interpreting



(a) $a_m = 1, a_d = 1000$ (b) $a_m = 8, a_d = 10$

Fig. 4.7 Enhanced view to heat map where the best (a) and worst (b) parameter combinations occur when sorted according to rank scores across the 10 environment setups.

the data. The data indicates that foraging performance is inversely proportional to a_m and r_m (because cells get darker with increasing a_m and r_m). On the other hand, cells in the heat maps get brighter with increasing a_d and r_d. Thus for the N100-Q1, the best parameter combinations are those that minimise a_m and r_m while maximising a_d and r_d. This is also evident from the heat maps where the best (A1m1000d-R1m1000d) and worst (A8m10d-R10md1) combinations for N100-Q1 occurred as shown in Fig. 4.7.



(a) One100m (b) Uniform50m

Fig. 4.8 Sorted foraging times for 900 chemotaxis parameter combinations normalised using the mean time for Random Walk for the corresponding world setup.

The parameter search simulation experiments for determining the best-performing chemotaxis parameters for the N100-Q40 RepAtt setup was conducted for the 10 world setups of Fig. 3.5. The ranking of the foraging times for One100m and Uniform50m are shown in Fig. 4.8, where the foraging times were normalised by that of Random Walk (A1m1d-R1m1d) in the corresponding world (the plots for all the ten test environments are available in Appendix B). The foraging performance for the corresponding combinations of N0-Q1 RepAtt setup have also been included to visualise its correlation with N100-Q40 foraging performance. A4m1000d-R1m1d with a mean foraging time of 0.31 performed best in the One100m environment. This shows that the robots used attraction signals to aid cooperation with other swarms to exploit the clustered deposit of targets, similar to what was observed in Chapter 3 for the N0-Q1 parameter optimisation. However, the attraction multiplier, $a_m = 4$, that exhibited the best performance for N100-Q40 was lower than the $a_m = 6$ in N0-Q1. With the best performance in most clustered environment for N100-Q1 having $a_m = 1$, it suggests that the chemotaxis' reliance on the a_m parameter is affected by the noise level in the signal. In the clustered environment, excessive noise in N100-Q1 meant the optimal solution was one that ignores the effects of a_m by setting it to 1 in order to prevent unnecessary tumbles. Using the average filter to improve the accuracy of the swarm's estimate of the temporal gradient in N100-Q40 meant that the optimal solution was one that increased the effect of a_m by setting it to 4, and when noise was absent the swarm had 100% confidence in the temporal gradients it computes from the attraction signal. Thus for N0-Q1, a higher $a_m \geq 6$ generally performed better than lower a_m for the swarm in the highly clustered One100m environment.

In the Uniform50m world, which is the least clustered of the ten test worlds, the best performance for N100-Q40 was recorded in the A1m100d-R1m50d parameter combination. Only swimming parameters (a_d and r_d) played a role in improving the swarm's performance. Considering that attraction signals are of less significance when resources can easily be located by robots in the swarm (as in Uniform50m environment), the $a_d = 100$ parameter would only have little effect on the swarm's foraging ability. Thus, it is logical to conclude that the most influential factor in improving the swarm's performance was the $r_d = 50$ parameter. This is similar to the results recorded for N0-Q1, where A1m1d-R1m1000d produced best performance for the Uniform50m world setup.

The scoring system used to identify the best parameters for N0-Q1, which involved summing the ranks of each combination across the ten worlds, was applied to the simulation results for N100-Q40. The ranking of the scores is summarised in the heat maps in Fig. 4.9 (the heat maps for the ranking of the individual worlds are available in Appendix B). Moving from left to right in Table 4.9 shows an increase in the number of brightly coloured cells in the heat maps, indicating that increasing r_d improves the swarm's foraging performance. In addition, as a_m increases, the lower parts of the heat maps become darker, suggesting that large values of a_m and r_m together have a negative impact on swarm foraging. Similar to a_d, increasing r_d has a positive impact on swarm foraging performance the cells get brighter when moving from left to right within each heat map). The combined effect of the

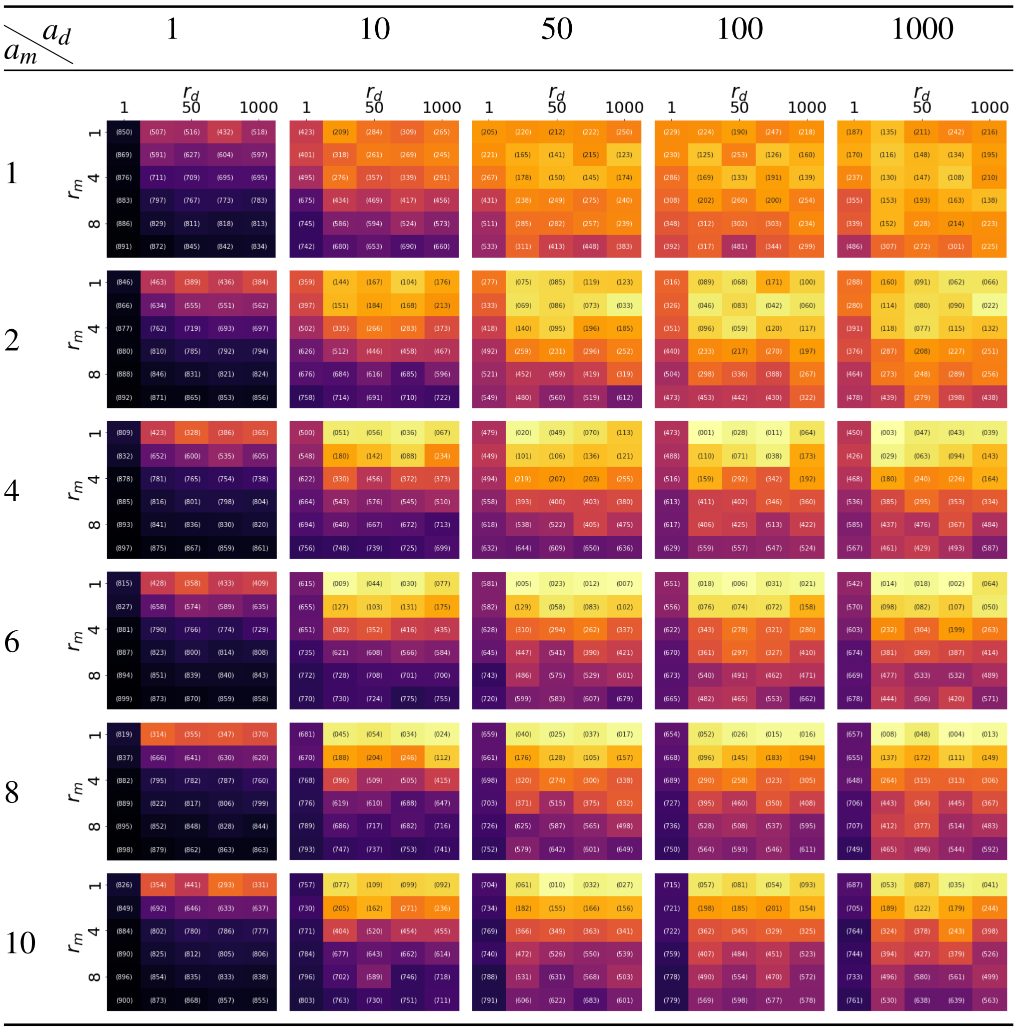
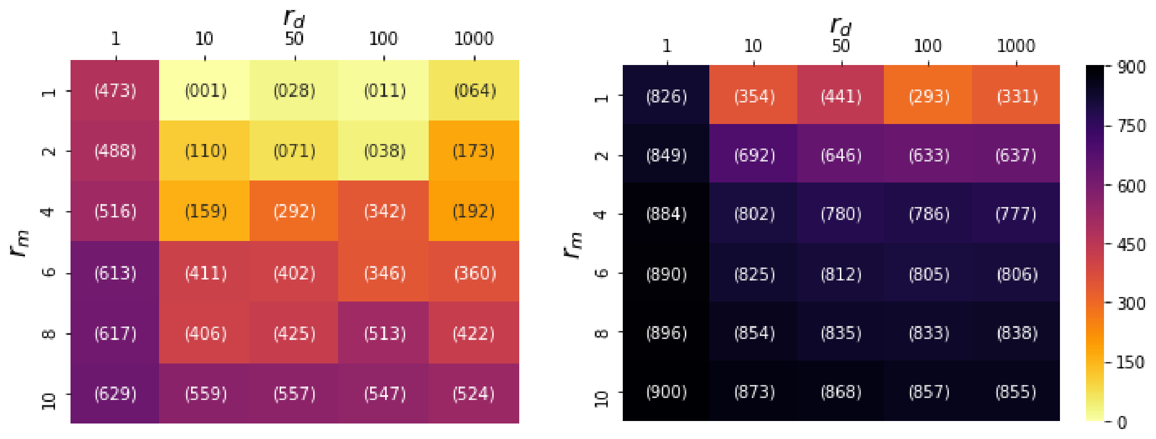


Fig. 4.9 Effect of the chemotaxis parameters on the foraging performance of a swarm of 36 robots foraging 180 out of 200 targets in the ten test worlds. Best combination for this distribution is $a_m = 4$, $a_d = 100$, $r_m = 1$, $r_d = 10$.

chemotaxis parameters across the ten world setups resulted in A4m100d-R1m10d delivering the best foraging performance score for the N100-Q40 RepAtt setup, while the worst ranked combination was A10m1d-R10m1d (where robots relied only on the tumble promoting parameters and ignored tumble suppressing parameters). The heat maps in Fig. 4.10 show the enhanced view for the heat map where the best and worst parameter settings for N100-Q40 occurred. In all subsequent simulations, the best performing parameters for the N100-Q40 RepAtt setup, A4m100d-R1m10d, will be used because it generated superior performance when using realistic communication noise and the simple average noise filtering system.



(a) $a_m = 4$, $a_d = 100$ (b) $a_m = 10$, $a_d = 1$

Fig. 4.10 Heat map where the best and worst parameter combinations occur when sorted according to rank scores across the ten environment setups.

4.4 Foraging with Noisy Communication

In this section, the performance of a foraging swarm using A4m100d-R1m10d RepAtt parameters and noisy communication is studied under varying filter queue sizes in order to identify the most appropriate queue size for the swarm. The swarm size was 36 robots and the queue size was varied from 1 to 120 and the foraging task was conducted on the 10 environment setups of Fig. 3.5.

Scatter plots showing simulation snapshots for N100-Q1, N100-Q40 and N0-Q1 are presented in Figss 4.11 and 4.12 for One100m and Uniform50m environments respectively. In the One100m, where attraction communication was an important factor, robots foraging with N100-Q1 were unable to disperse effectively using repulsion, and their use of attraction signal to cooperate with other robots was less efficient. Thus, after 200 seconds of foraging, they were only able to pick up 49 out of 200 targets. Using the average filter in N100-Q40, robots were able to disperse more effectively and had an improved ability to make use of attractive signals to exploit the targets cluster. Within 100 seconds, the swarm was able to cover the 100 m $\times$ 100 m search space and after 200 seconds, more of the robots were able to locate the clustered targets by using chemotaxis to maximise the attraction they sensed. In the absence of noise (N0-Q1), the robots were most effective in using chemotaxis to locate the attracting robots, thus improving their ability to locate and forage the target cluster. Within 200 seconds, the robots were already able to pick up 50% of the targets and many robots in the swarm were able to come within the region of the targets cluster.

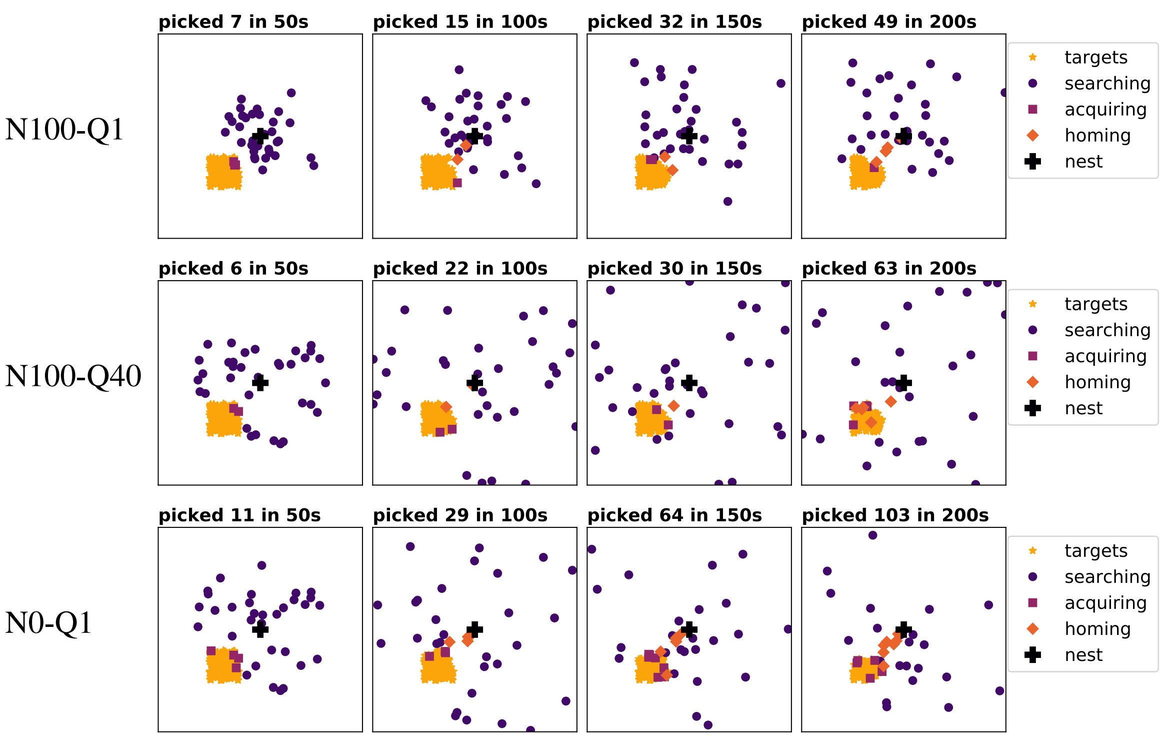


Fig. 4.11 Scatter plots for One100m simulation snapshots. The optimized chemotaxis parameters for N100-Q40 $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ for all the simulations in this table.

In the simulation scatter plots in Fig. 4.12 for the Uniform50m world, the performance of N100-Q1, N100-Q40 and N0-Q1 were indistinguishable within the first 50 seconds. This is because most robots were able to locate targets to forage as they left the nest. However, when targets became scarce in the later part of the simulations, the robots needed to conduct effective dispersal in order to help them explore a wider search space. At this stage, N100-Q1 began to lag in performance behind N100-Q40 and N0-Q1. Thus, as the simulation approaches 200 seconds, N100-Q1 was able to pick up 148 targets, while N100-Q40 and N0-Q1 were able to pick up 172 and 178 targets respectively. This emphasises the need for the swarm to be capable to effective use of communication signals when targets are difficult to locate in the search space.

When robots in the swarm communicate attraction and repulsion signals under noisy conditions, the data so far indicates that it has a negative effect on the swarm foraging performance compared to perfect communication. However, with the inclusion of the simple average filtering system, the robots were able to improve their use of communication signals. Should the average filter queue then be made as long as possible in order to ensure the robots

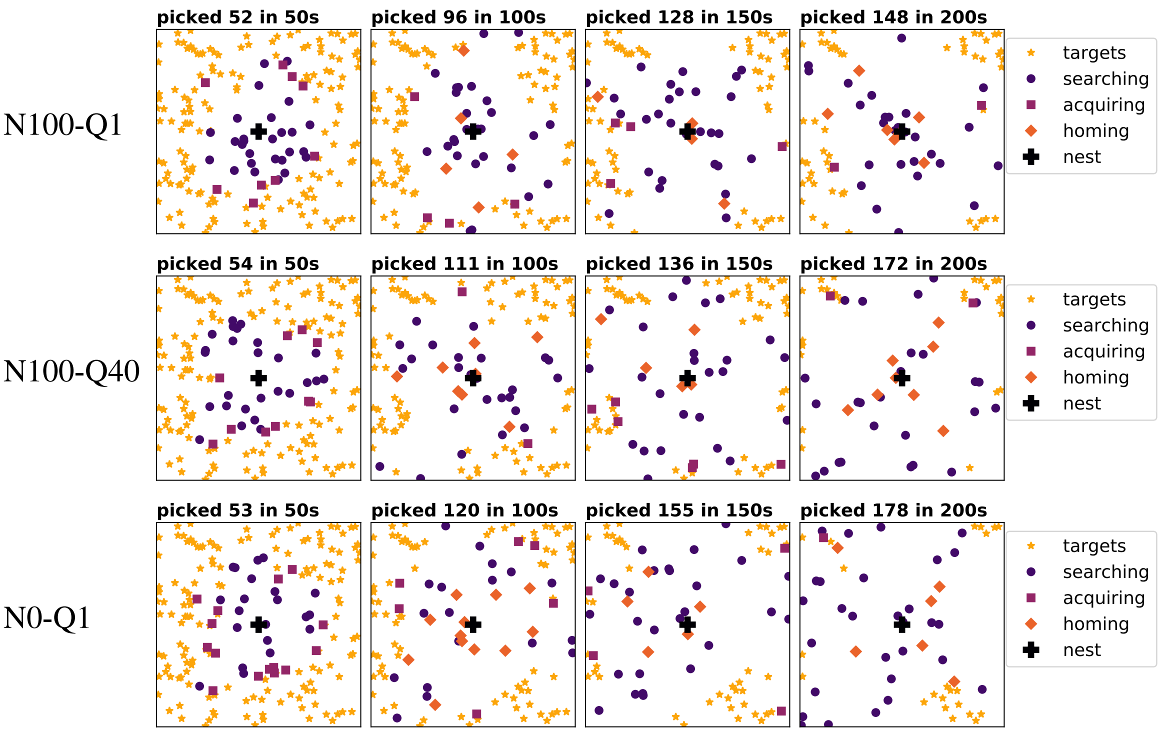
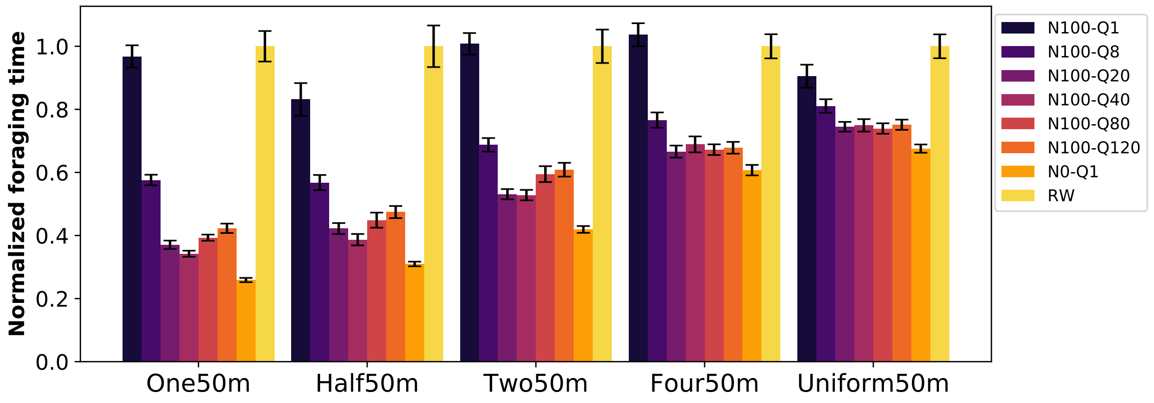


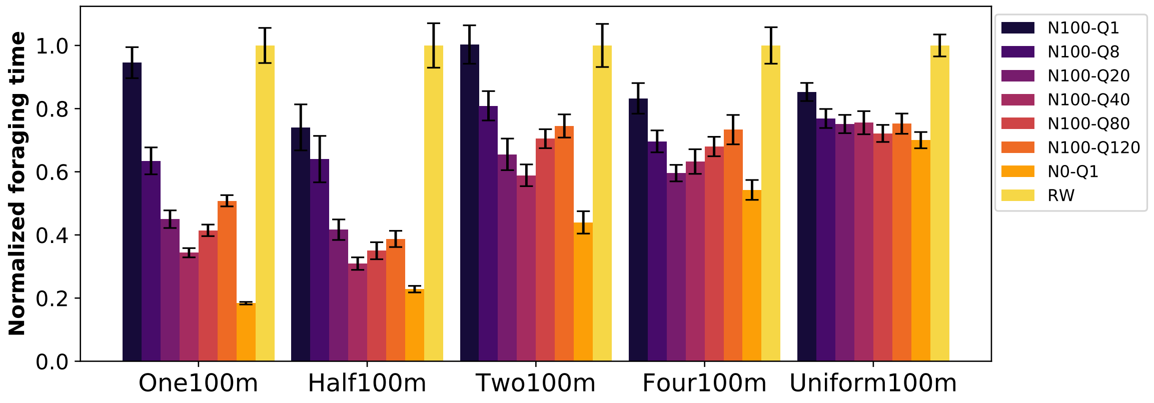
Fig. 4.12 Scatter plots for Uniform50m simulation snapshots. The optimised chemotaxis parameters for N100-Q40 $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ for all the simulations in this figure.

are able to measure temporal gradient with 100% accuracy under noisy conditions? To answer this, Fig. 4.13 shows the simulation results of RepAtt (using parameter combinations A4m100d-R1m10d), where the queue was varied from 1 to 120 and tested on the ten environmental setups.

The data in Fig. 4.13 shows that initial queue size increments have a positive effect on the swarm foraging performance, but performance degrades again once the queue becomes excessively long. The effect of queue size is more pronounced in the clustered environments, where attraction communication plays an important role in improving swarm foraging, and in the 100 m $\times$ 100 m worlds, where the search space was quadrupled (in comparison to the 50 m $\times$ 50 m worlds). In the three most clustered environments of the 50 m $\times$ 50 m world size (One50m, Half50m and Two50m), the foraging performance improved as the filter queue size approached 40, and then started to degrade as the queue size was increased further. In the Four50m and Uniform50m worlds, there was no notable improvement in swarm foraging performance beyond a queue size of 20, but there was no significant decrease in performance beyond this queue size either. For the 100 m $\times$ 100 m world size, the



(a) 50m × 50m



(b) 100m × 100m

Fig. 4.13 Varying queue size and its effect on foraging time. The optimised chemotaxis parameters combination of $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ was used for all simulations. Each bar shows the mean of 30 simulation repetitions and error bars represent 95% confidence interval.

swarm foraging performance in response to increase in queue size was similar to that of 50 m × 50 m for the one, two, half and uniform distributions (the primary difference was a relatively larger response in the 100 m × 100 m clustered environments). However, for the Four100m world, there is a noticeable degradation in foraging performance as the queue size increases beyond 20 (whereas for the Four50m, there was no noticeable effect in the corresponding range). These data (in Fig. 4.13) show that it is important for robots in the swarm to increase queue size to help them compute reliable temporal gradients for both repulsion and attraction signals while simultaneously keeping the queue short so they make timely response to changes in these gradients. The velocity of the robots in the swarm was 0.6 m/s and they sensed sound intensity at a rate of 40 samples per second. Thus, a queue size of 40 means the robots updated their intensity measure once every second (or once every 0.6 m). The results show that using the noisy sound model with a queue size of 40 was most appropriate for the 10 test environments, given the distance the robots would have covered within that time. In subsequent studies, RepAtt with queue size of 40 (N100-Q40) will be used when swarm communication is noisy.

In comparison to N0-Q1, in Fig. 4.13, N100-Q40 had reduced swarm foraging performance. This was most significant in the One100m world where N100-Q40 was 43% less effective than N0-Q1. The smallest degradation in performance was 8%, which was observed in the Uniform100m world. The coordination of the swarm when using N100-Q40 was greatly significant nonetheless, because it improved foraging performance by up to 70% in comparison to Random Walk on the One100m test environment and had a minimum foraging time improvement of 23% in the Uniform50m world setup. This shows that under realistic noise conditions, RepAtt would maintain its effectiveness in environments that rely heavily on communication (One100m) and in those where communication is less relevant (Uniform50m).

4.5 Robustness of RepAtt

In Section 3.9 it was shown that the variation in the foraging time of N0-Q1 was 2.27 across the 10 world setups (A10m50d-R1m100d was used in that simulation), while the centralised ALNS algorithm varied by only 1.49. Fig. 4.14 shows the box plots of how swarm foraging time varied across the 10 world setups with RepAtt parameters of A4m100d-R1m10d, where foraging times were normalised by the shortest time taken for that communication model across all environments (which occurred in the Uniform50m). From Fig. 4.14, the variation in foraging time across the ten world setups for N0-Q1 and N100-Q40 were 1.90 and 2.91 respectively. This represents a 53% reduction in the algorithm's robustness measure (using the foraging time variation as the metric). Even with the decrease, N100-Q40, was still much more robust than Random Walk whose variation was 7.91 (that is, 172% less robust in comparison to N100-Q40). In comparison to the A10m50d-R1m100d robustness analysis conducted in Section 3.9 for N0-Q1, the A4m100d-R1m10d parameters combination was 16% more robust, thus, indicating that the chemotaxis parameters were also influential in RepAtt's robustness to changes in the foraging search space.

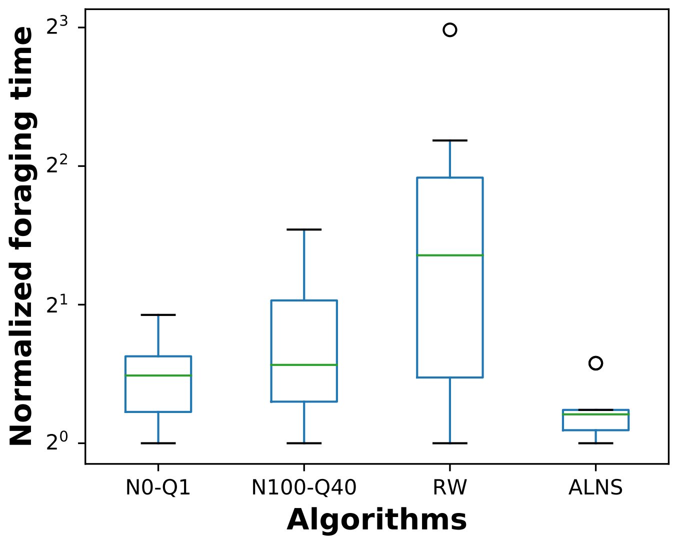


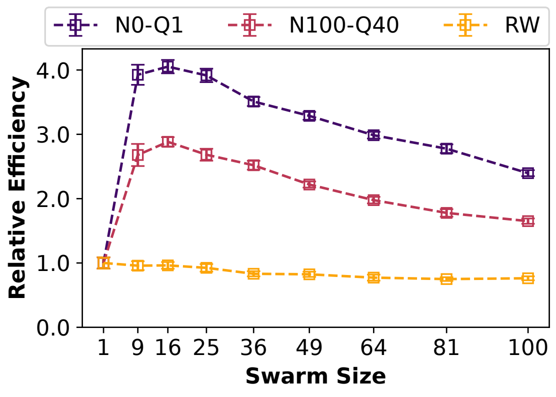
Fig. 4.14 Change in swarm foraging times across the ten target distributions. Each simulation was repeated 30 times, with y-axis showing mean time to pick up 90% of 200 targets (normalised based on distribution with shortest mean time for each algorithm, which is Uniform50m in all cases).

4.6 Scalability of RepAtt

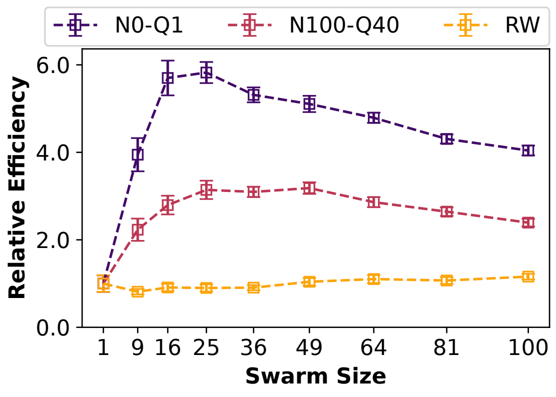
The scalability study conducted for N0-Q1 and Random Walk in Section 3.10 showed that the A10m50d-R1m100d chemotaxis parameters demonstrated impressive scalability with increasing swarm size. Fig. 4.15 shows how N0-Q1, N100-Q40 and Random Walk (RW) scale with increase in swarm size for A4m100d-R1m10d chemotaxis parameters. The plots for the ten test worlds are available in Appendix B. From Fig. 4.15, the largest improvement in the swarm's efficiency was attained in the clustered environments, with N0-Q1 improving efficiency by a factor of up to 5.82, while N100-Q40 attained 3.18 (in One100m environment). As the swarm size increased up to 100 robots, the N100-Q40 further was able to maintain foraging efficiency by a factor of up to 2. In the uniform worlds, the difference in performance improvement between N0-Q1 and N100-Q40 was less evident, with N0-Q1 achieving up to 1.61 with a swarm size of 25, while N100-Q40 attained 1.44 for the corresponding swarm size in the Uniform100m world.

4.7 Summary

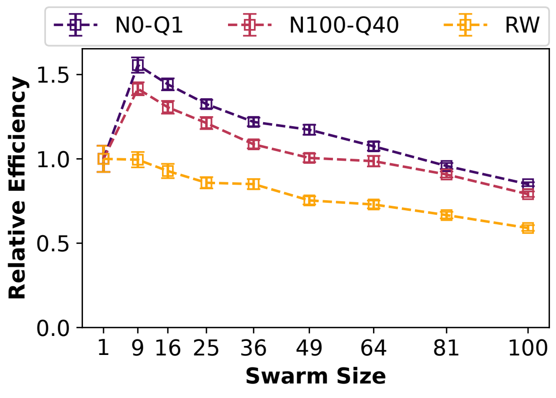
RepAtt's performance relies on the ability of robots to compute accurate gradients to aid their chemotactic search for targets in their environment during foraging. This chapter has



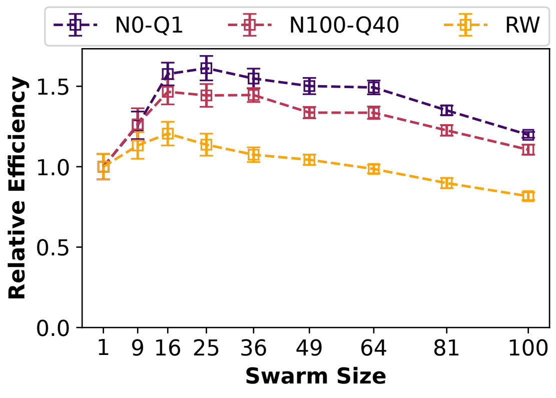
(a) One50m



(b) One100m



(c) Uniform50m



(d) Uniform100m

Fig. 4.15 Relative efficiency computed based on $p = 180$ out of 200 total targets (using Equation 3.6). Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ were used.

investigated how noise in communication of attraction and repulsion signals among robots impact their foraging performance. The noise was modelled from hardware experiments and integrated into the simulations. In addition, a simple noise filtering system based on computing the average across a queue of instantaneous measurements of the attraction and repulsion signals was implemented. This filtering system significantly improved the swarm's ability to mitigate the negative effects of noise in their communication during foraging. Through extensive series of simulations and analysis, RepAtt was confirmed to be capable of improving swarm foraging performance in all the test environments in a robust and scalable manner.

Chapter 5: Uncertainty in Foraging Robot Vision

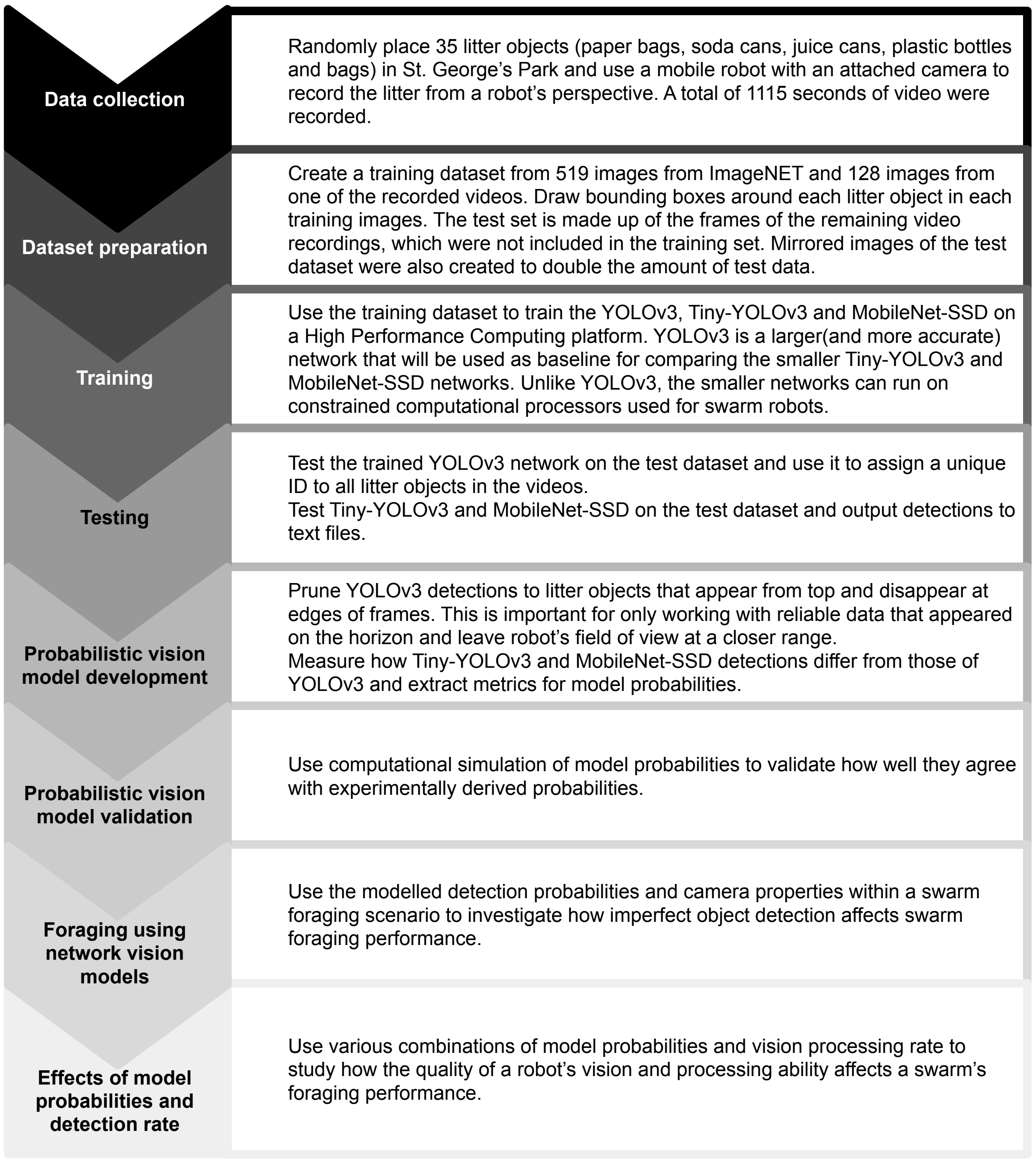
5.1 Introduction

The uncertainty in a robot's visual perception and how it affects performance at a swarm level is one of the least studied area of swarm robotics. It is expected that this uncertainty will have a negative impact on the swarm. However, whether this impact completely 'breaks' the collective behaviour of the swarm is yet to be addressed. This chapter develops a probabilistic robot vision model based on real-world experiments using state-of-the-art machine learning object detection algorithms for the detection of litter in a moderately realistic environment such as a local park. The model was used to extensively study the impact of imperfect vision on a swarm of robots foraging with or without communication. Developing the vision model involved a series of necessary steps summarised in Fig. 5.1.The subsections that follow will describe the modelling steps in detail before the model's application for swarm robots foraging.

The first step was to collect training and testing data (Section 5.2), followed by cleaning and annotation of the dataset (Section 5.3). The steps in training and testing the object detection models are described in Section 5.4. During testing, metric information needed for developing the probabilistic model was collected. This data was then used in Section 5.5 to develop the probabilistic model. Section 5.6 describes the validation process of the imperfect vision model. After validation, the vision model was included in the swarm simulation for RepAtt and Random Walk in Section 5.7. A study of the vision model was conducted in Section 5.8 to simulate variation in object detection hardware (through controlling model probabilities and rate at which robots update their vision system) and apply it for the investigation of its impact on the effectiveness of the swarm algorithms. The data collection, dataset preparation, training and testing steps (Sections 5.2 – 5.4) are generic steps taken to optimise the deep neural networks for object detection. The major contributions of this research for development of the probabilistic vision model for the swarm foraging robots are detailed from Section 5.5 to Section 5.8

5.2 Data Collection

Data collection involved randomly placing 35 litter objects (made up of soda cans, juice boxes, plastic bags, paper bags and plastic containers) in St. George's Park on the University of Leeds campus and using a mobile robot to explore the litter area while recording the scene



Data collection	Randomly place 35 litter objects (paper bags, soda cans, juice cans, plastic bottles and bags) in St. George's Park and use a mobile robot with an attached camera to record the litter from a robot's perspective. A total of 1115 seconds of video were recorded.
Dataset preparation	Create a training dataset from 519 images from ImageNET and 128 images from one of the recorded videos. Draw bounding boxes around each litter object in each training images. The test set is made up of the frames of the remaining video recordings, which were not included in the training set. Mirrored images of the test dataset were also created to double the amount of test data.
Training	Use the training dataset to train the YOLOv3, Tiny-YOLOv3 and MobileNet-SSD on a High Performance Computing platform. YOLOv3 is a larger(and more accurate) network that will be used as baseline for comparing the smaller Tiny-YOLOv3 and MobileNet-SSD networks. Unlike YOLOv3, the smaller networks can run on constrained computational processors used for swarm robots.
Testing	Test the trained YOLOv3 network on the test dataset and use it to assign a unique ID to all litter objects in the videos. Test Tiny-YOLOv3 and MobileNet-SSD on the test dataset and output detections to text files.
Probabilistic vision model development	Prune YOLOv3 detections to litter objects that appear from top and disappear at edges of frames. This is important for only working with reliable data that appeared on the horizon and leave robot's field of view at a closer range. Measure how Tiny-YOLOv3 and MobileNet-SSD detections differ from those of YOLOv3 and extract metrics for model probabilities.
Probabilistic vision model validation	Use computational simulation of model probabilities to validate how well they agree with experimentally derived probabilities.
Foraging using network vision models	Use the modelled detection probabilities and camera properties within a swarm foraging scenario to investigate how imperfect object detection affects swarm foraging performance.
Effects of model probabilities and detection rate	Use various combinations of model probabilities and vision processing rate to study how the quality of a robot's vision and processing ability affects a swarm's foraging performance.

Fig. 5.1 Development workflow for the robot vision model and the investigation of the effects of imperfect vision on a swarm of foraging robots.

with an attached camera. A total of 23 separate videos were recorded. The total recording time across all videos was about 20 minutes. Fig. 5.2 shows a schematic diagram of the mobile robot setup used for the data collection. The camera was elevated at 44 cm from ground level and oriented at 35° facing downwards. The camera model used was a GoPro Hero 5 set to record at a full HD resolution of 1920 × 1080 pixels, wide angle field of view (at 118° and 69.5° horizontal and vertical view angles), and 50 frames per second recording rate. After the recording exercise, all the litter were picked up and binned.

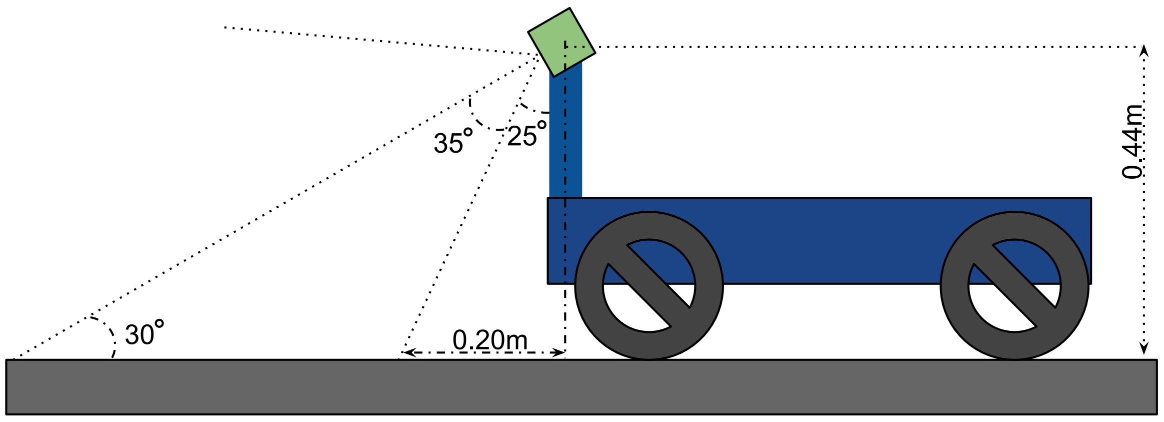


Fig. 5.2 Robot setup for collecting data of litter objects in St. George's Park.

5.3 Dataset Preparation

Deep neural networks based object detection algorithms such as YOLO (You Only Look Once) and SSD (Single Shot Detector) need to be trained with images that contain the objects they need to detect. These images also need to be annotated by specifying bounding boxes around the regions of interest occupied by the object. A training dataset needs to cover a wide range of visualisation angles of the object, different zoom levels, under different light conditions and also partially occluded images of the object(s) to be detected. These variations in the object's view will help the network to better generalise the features of the objects of interest, thus, improving its ability to detect the objects in different contextual scenarios. No pre-annotated images of litter were found on the Internet, so a custom dataset was created for training the networks for litter detection. The training dataset comprised of 519 images downloaded from ImageNET [1] and 128 frames obtained from one of the litter recording videos.

The images were annotated using the Openlabelling annotation tool [120] by manually drawing bounding boxes around each litter object in the images. The result from this process

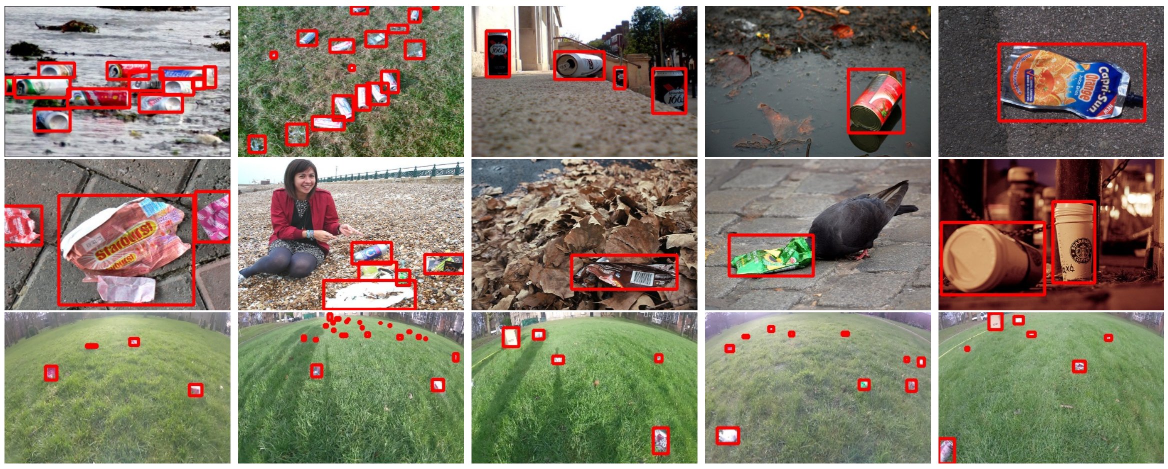


Fig. 5.3 Sample images of images used as training dataset with litter objects annotated using red bounding boxes. The first two rows are images downloaded from ImageNET [1] and the third row are sample images from litter recording in St. George's Park.

is a text file with the same name as the image. Each line in the file is a representation of the top left and bottom right coordinates of the bounding box of a specific litter. Thus, an image containing 10 litters has 10 lines in the corresponding text file. In total, there were 2609 bounding box annotations spread across the 647 images that make up the training set. The annotation process took more than 6 hours. Fig. 5.3 shows the annotations of 15 sample images used for training the object detection networks.

The video recordings used for testing the networks were flipped horizontally in order to increase the number of test data. This resulted in more than 100,000 images (video frames) used to test the networks. It was not feasible to annotate bounding boxes on each frame of the videos within the limited time for the research. So, the approach adopted for measuring the performance of the test network was to use the larger, more accurate YOLOv3 detections as pseudo ground truth data. The less accurate smaller networks (Tiny-YOLOv3 and MobileNet-SSD) performance were then measured based on their agreement with YOLOv3's detections. To make the YOLOv3 detections more reliable, further post-processing involving optical flow tracking were used to interpolate locations of missed detections.

5.4 Training and Testing

The three object detection networks (YOLOv3, Tiny-YOLOv3 and MobileNet-SSD) used in this research to develop the model were trained on the Advanced Research Computing 3

High Performance Computing (ARC3 HPC) for 20,000 iterations. The trained weights were then copied from the ARC3 HPC to the local machine for testing and model development.

The first testing phase was broken down into multiple steps that involved horizon masking, litter ID assignment and detections pruning in order to extract YOLOv3 detections that were reliable to use as baseline (or pseudo ground truth). The second testing phase was used to extract relevant metric information for developing the vision models based on the smaller networks. It involved using two sizes of each network, determining inference time in constrained computational environments, metric data extraction and logging the detections by the networks.

5.4.1 Testing phase 1: preparing YOLOv3 as baseline

The tasks performed in preparing the YOLOv3 baseline data are as follows:

1. Use the YOLOv3 network to detect litter in the test dataset and save the detection data for each frame as a text file (Fig. 5.4a shows detections on a sample frame). Each line in the text file represents the top-left and bottom-right corners of detected litter in the video frame.

2. With the robot stationary on a flat surface in St. George's Park, measure off five metres from centre of bottom edge to the left, middle and right edges of the robot's field of view. Use this measurement to form a circular five metres horizon, then, filter out litter objects that are outside the manually generated horizon. Fig. 5.4b shows the effect of applying the horizon. This step was necessary to eliminate detections that were so distant and essentially 'spots' in the frames.

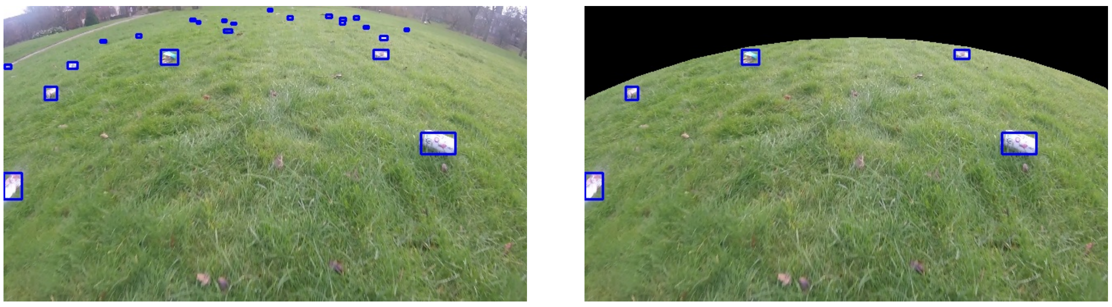

(a) Video frame (b) Horizon applied to frame

Fig. 5.4 Detecting litter using YOLOv3 and applying five-metre horizon to it.

3. All detections that are within the five-metre horizon get assigned unique IDs used for tracking them across multiple frames in a video. These IDs are assigned when the first

litter comes into view. The litter is assigned a new ID whenever it leaves and re-enters the frame. For the first frame, assign unique ID to all detections in a video's first frame and move on to subsequent frames to assign IDs based on the following guide:

(a) Check for litter objects in current frame with the most overlap with detections in the preceding frame by computing the intersection over union (IOU) value for each litter. Detections in current frame with the most overlaps in the preceding frame are assigned the IDs of the litters they overlap with.

(b) For detections that were present in the previous frame and absent in the current frame (because they do not overlap with detections in the present frame), use optical flow to estimate their new positions in the current frame, while retaining their IDs.

(c) Assign new unique IDs to detections that do not overlap with any litter in previous frame and those that had less overlap, in cases where multiple litter in current frame overlap with one litter in the previous frame.

The resulting process assigns unique IDs to all the litters detected in all the test dataset videos. Fig. 5.5 shows a sample frame where L169 litter is a new litter, L165 was tracked using optical flow, while L147 and L159 are examples of detections that overlapped with those of a preceding frame. A total of 13,449 unique litter IDs were assigned across the whole test dataset. Fig. 5.6a shows the pixel locations of all the litter objects across all the test videos within the robots field of view.

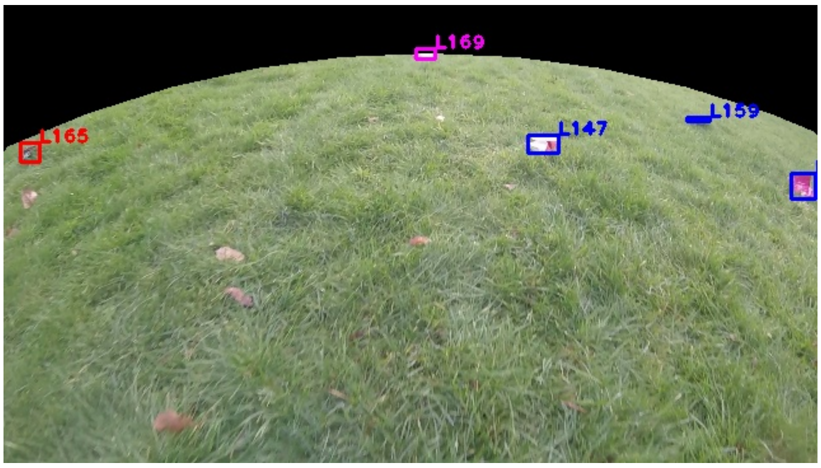


Fig. 5.5 Sample image showing the different tags a litter can have in a frame. Pink box represents the new litter with ID L169, blue boxes are for litters that were tracked using IOU, while red boxed litter is tracked using optical flow.

4. The last step in preparing the baseline data involved pruning the detections with a goal to eliminate unreliable data and use only the most reliable information as metrics for evaluating the performance of the smaller networks and developing the vision model for the swarm. The first pruning step involved extracting litter detections that were tracked using IOU for 70% or more of the frames in which they appeared. This reduced unique litter IDs from 13,449 to 9,899, representing a 26% decrease in the number of unique litter objects (Fig. 5.6b). However, there were many litter objects whose first appearance was not on the five metres horizon (the robot's motion was forward with occasional turns to the right or left, so litter should generally appear from the horizon or the sides of the frame). An additional observation was that many litter objects appear and disappear on the five metres horizon because the park's surface was not entirely smooth, so this was caused by the wobbling of the robot as it traverses the park. Based on these observations, the last pruning step involved extracting only litter objects whose first appearance was within 30 pixels of the horizon and last appearance was within bottom 600 pixels of the frame and within 30 pixels of the bottom, left or right edges of the video frames. This pruning step reduced the unique litter IDs from 9,899 to 679, which represented 5% of the 13,449 original litter IDs count (Fig. 5.6c). Fig. 5.6 shows the bounding box centres for the first, last and all occurrences of the litter objects in frames of the test dataset videos.

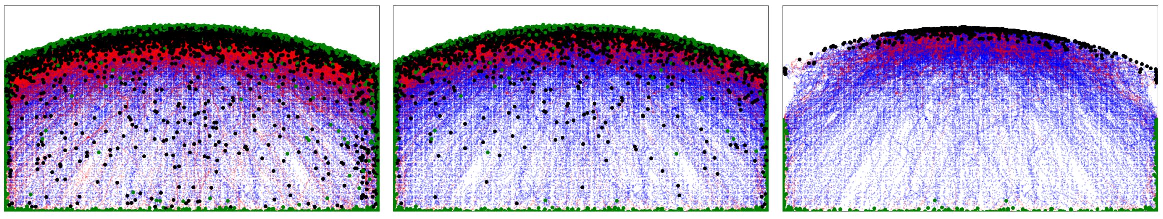

(a) 13,449 unique IDs (b) 9,899 unique IDs (c) 679 unique IDs

Fig. 5.6 The path of each litter detected by YOLOv3 within the videos. Black spots are first appearance of litter, red are positions tracked using optical flow, blue are positions tracked using IOU and green are last location of litter within the frame.

5.4.2 Testing phase 2: extracting metrics data for smaller networks

The detections using the two smaller networks (Tiny-YOLOv3 and MobileNet-SSD) also involved a number of steps, which were used to extract modelling information. These steps are described below:

1. Use two input resolution setups for Tiny-YOLOv3 (128×128 and 224×224) and MobileNet-SSD (124×124 and 220×220). The lower resolutions are less accurate, but have shorter inference times, while the higher resolutions are more accurate at the expense of longer inference times. This is useful for investigating speed versus accuracy objectives on constrained computational environments. The input resolutions for Tiny-YOLOv3 and MobileNet-SSD are different because their deep neural network architectures impose which resolutions align well with the transformations the input image undergoes as it progresses through the network layers. Using input resolutions that do not align well negatively impacted the ability of the network to localise the its detections on the input image. Thus, the input resolutions for Tiny-YOLOv3 and MobileNet-SSD were chosen to be as close to each other as possible.

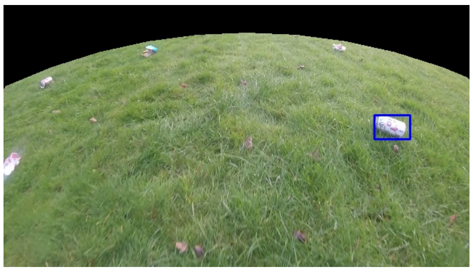
2. The networks were tested on Erle-Brain 3 (robot controller based on Raspberry Pi 3) and a Raspberry Pi 4 computer to measure the inference time (time taken to produce an output for a given an input image), shown in Table 5.1. These platforms were chosen because they were of relatively low cost for the development of swarm of field robots, and also possess the computational ability for performing object detection using lightweight neural networks.

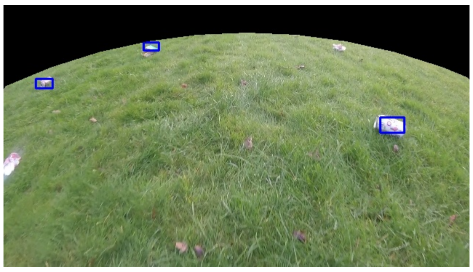
Table 5.1 Inference time (in seconds) for Raspberry Pi 4 (Pi4) and Erle-Brain 3 (Pi3) computers. The values represent the mean inference time and standard deviation for a selected video in the test dataset.

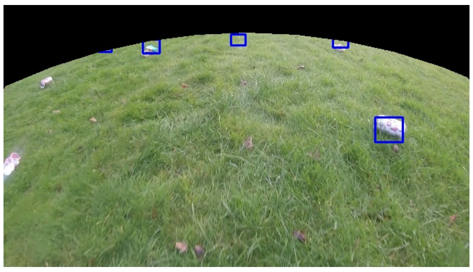
	MobileNet-SSD		Tiny-YOLOv3	
	124×124	220×220	128×128	224×224
Pi4	0.0638 ± 0.0110	0.1562 ± 0.0219	0.0611 ± 0.0064	0.2000 ± 0.0225
Pi3	0.4282 ± 0.0357	1.1913 ± 0.1362	0.4965 ± 0.0488	1.6000 ± 0.0470

3. Confirm that the detection of the network for a specific visual scene is consistent across all test platforms (on Raspberry Pi and ARC3 HPC). This was an important step because if the detections are same, then using the HPC for inference will significantly reduce the time needed to perform detections on all the test dataset.

4. Use each of the two input sizes of the Tiny-YOLOv3 and MobileNet-SSD to detect litter in all frames of the test dataset and store the detections data in a text file (one text file per frame) such that each line in the text file represents the top-left and bottom-right corners of the detections bounding boxes. Fig. 5.7 shows sample detections by the networks.


(a) Tiny-YOLOv3 128 × 128


(b) Tiny-YOLOv3 224 × 224


(c) MobileNet-SSD 124 × 124

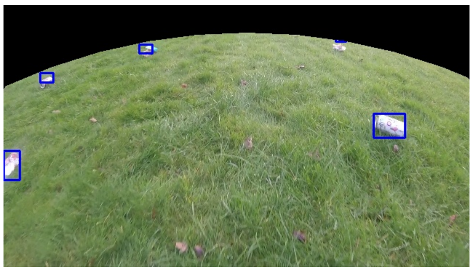

(d) MobileNet-SSD 220 × 220

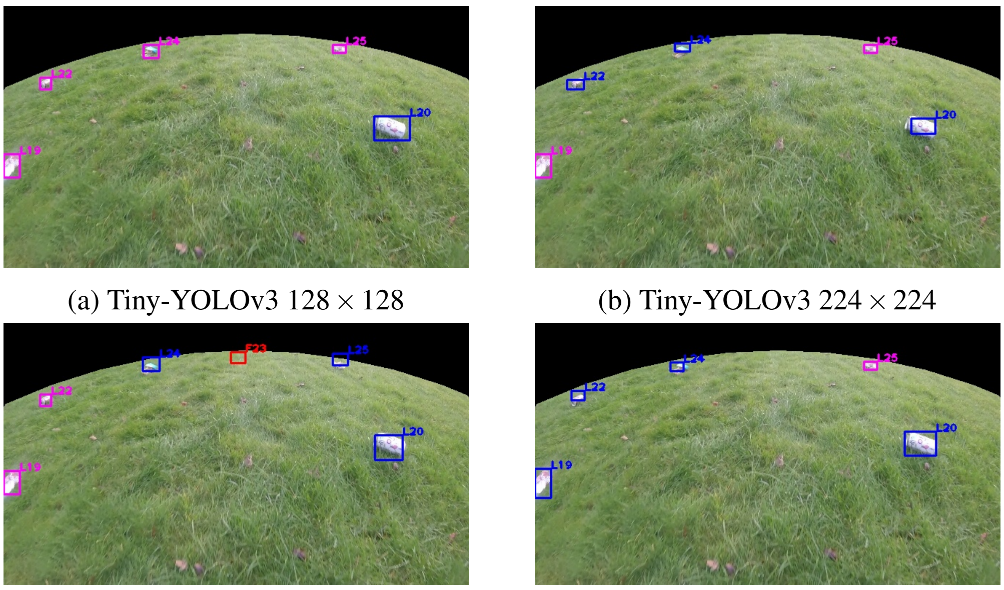
Fig. 5.7 A sample detection using two sizes of Tiny-YOLOv3 ((a) and (b)) and MobileNet-SSD ((c) and (d)). The larger input sizes are more successful in detecting litter, however, they usually have longer inference time because of the increased computational requirement.

5. Working with only detections that fall within a five-metre horizon, compare the network's detection in each frame with that of YOLOv3 using intersection over union measurement (IOU). Use an IOU of at least 0.0001 as a positive detection to effectively account for bounding box positioning mismatch between YOLOv3 detection and those of the smaller networks.

6. Class all IOU above the 0.0001 threshold as True Positive (TP) detection by the smaller network. The detections in YOLOv3 that were missed by the smaller networks are classed as False Negatives (FN) and detections present in the smaller network and absent in the YOLOv3 are classed as False Positive (FP). The classifications of detections in Fig. 5.7 are shown in Fig. 5.8.

7. The detection data are logged in a tabular format and saved in two separate text files:

   (a) TP and FN file, where the litter id (from YOLOv3) data were represented as the rows, and each column represented the frames in the video. For each litter, a value of 1 is assigned to every frame the smaller network detects it (TP) and 0 for frames where detection is unsuccessful (FN). If the litter was not within the field of view of the robot in a particular frame, its cell is left blank for that frame.



(a) Tiny-YOLOv3 128 × 128 (b) Tiny-YOLOv3 224 × 224

(c) MobileNet-SSD 124 × 124 (d) MobileNet-SSD 220 × 220

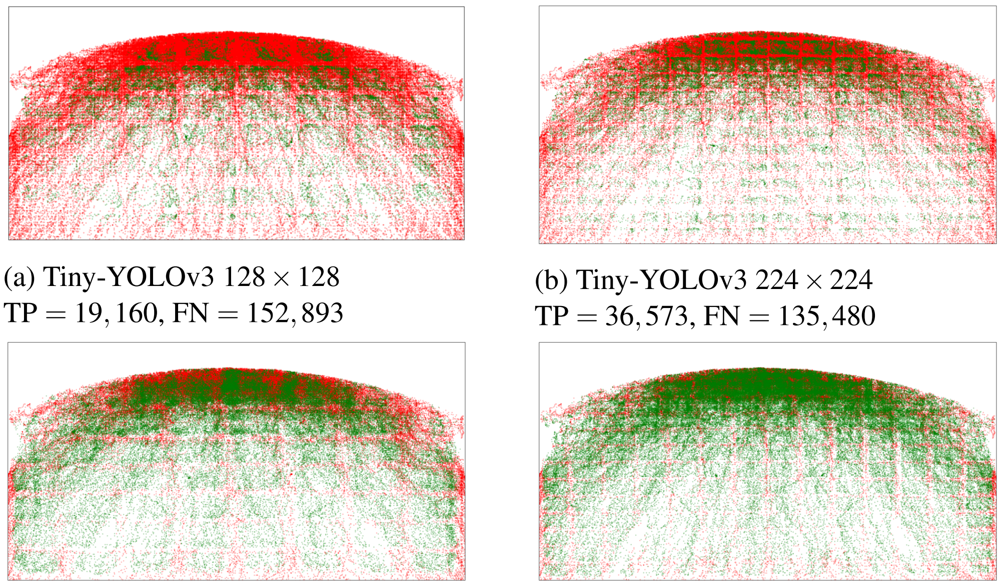
Fig. 5.8 The data from Tiny-YOLOv3 and MobileNet-SSD are compared with corresponding data from YOLOv3. All detections that match (shown in blue) are True Positives; False Negative detections are shown in pink; while the red box represents False Positive detections.

(b) FP file, also uses a similar tabular format to assign 1, 0 or leave blank for appearance of the False Positive data in the frames of the videos.

The resulting detection performance of the networks based on the 679 litters in YOLOv3 is shown in Fig. 5.9. The green dots represent true positive detections, while missed detections are represented as red dots. It is obvious that the MobileNet-SSD variants outperformed the Tiny-YOLOv3 network (because they have more green dots).

5.5 Model Development

The detections data while testing the object detection networks showed that once a litter was detected by the network in the current frame, it was highly likely that it would be detected in the next frame. However, if the network failed to detect the litter in the present frame, it was unlikely for the network to detect that litter in the next frame. This was because the change in visual scene between consecutive frames was generally small, considering that the videos were recorded at 50 frames per second and the robot was driven at a slow pace. Thus, the data showed that detections on consecutive frames were not statistically independent. From observing this detection pattern of the networks, a probabilistic vision model was developed


(a) Tiny-YOLOv3 128×128
TP $= 19,160$, FN $= 152,893$

(b) Tiny-YOLOv3 224×224
TP $= 36,573$, FN $= 135,480$

(c) MobileNet-SSD 124×124
TP $= 68,801$, FN $= 103,252$

(d) MobileNet-SSD 220×220
TP $= 103,887$, FN $= 68,166$

Fig. 5.9 Representation of the matching accuracy of detections in the smaller networks against those of YOLOv3. Red spots are missed detections (FN) while green spots are matching detections (TP). The data was generated from the positions of all 679 unique litter IDs across all video frames used for testing the networks.

to provide a representative approximation of the detection pattern. The model was based on the computation of two transition probabilities:

1. Probability that the object detector will detect the litter in the current frame when the object was unseen in the previous frame. This could be because the object in question had just entered the robot's field of view or it was classified as a false negative in the preceding time step. The probability controls the transition of an object from being unseen to it being seen and is represented as P_{u2s}

2. Probability that a detection in the preceding time step is detected in the current time step. This probability controls how the detection of an object can persist in a seen state across multiple consecutive frames or time steps. This probability is represented as P_{s2s}.

The computation of these probabilities for the Tiny-YOLOv3 (both 128×128 and 224×224 network input resolution) and MobileNet-SSD (both 124×124 and 220×220) were based on analysing the 679 litter objects that were filtered through the pruning process of the baseline YOLOv3 network detections. These steps were followed to arrive at the model:

1. Tally all the seen to seen ($s2s$), seen to unseen ($s2u$), unseen to seen ($u2s$) and unseen to unseen ($u2u$) transitions for all the 679 litter objects. Also, tally the number of frames the litters were detected (s) and undetected (u) by the network. These values can easily be computed from the logged data by analysing the ones (seen) and zeros (unseen) values for each litter across the frames of the test dataset.

2. From the tallied data, compute the transition probabilities for each litter using Equations 5.1 for P_{u2s} and 5.2 for P_{s2s}. Use the u and s data to compute the detection probability of each litter as shown in Equation 5.3.

$$P_{u2s} = \frac{u2s}{u2s + u2u} \tag{5.1}$$

$$P_{s2s} = \frac{s2s}{s2s + s2u} \tag{5.2}$$

$$P_s = \frac{s}{s + u} \tag{5.3}$$

3. Compute the mean and standard deviation of the P_{u2s}, P_{s2s} and P_s from those computed for each litter. The end result of this process is shown in Table 5.2

Table 5.2 Analysis of the MobileNet-SSD and Tiny-YOLOv3 performances and metrics computation by analysing all frames of the test dataset where the 679 filtered litter objects visible (that is, analysis of the data at the full frame rate of 50fps).

	MobileNet-SSD		Tiny-YOLOv3	
	124×124	220×220	128×128	224×224
seen	68801	103887	19160	36573
seen2seen	59380	91994	16622	31840
seen2unseen	9397	11840	2538	4728
unseen	103252	68166	152893	135480
unseen2seen	9394	11807	2535	4699
unseen2unseen	93203	55733	149679	130107
never seen	11	0	238	146
always seen	0	0	0	0
nlitter	679	679	679	679
P_{s2s}	0.7896 ± 0.1824	0.8558 ± 0.1002	0.4358 ± 0.3954	0.5462 ± 0.3667
P_{u2s}	0.1293 ± 0.1031	0.2567 ± 0.1586	0.0224 ± 0.0325	0.0563 ± 0.0725
P_s	0.4078 ± 0.2304	0.6104 ± 0.2156	0.1101 ± 0.1655	0.2102 ± 0.2456

From the probabilities computed in Table 5.2, it can be seen that there is a significant difference between P_{s2s} and P_{u2s} probabilities. All the networks have a low P_{u2s}, which means that the networks have a low chance of detecting previously unseen litter. However, when the litter has been detected by the network, there is a high likelihood for the network to detect it in consecutive frames that follow the detection, which is represented by the relatively high P_{s2s} probability. The MobileNet-SSD networks, with $P_s = 0.4078 \pm 0.2304$ and 0.6104 ± 0.2156 for input sizes 124×124 and 220×220 respectively, performed better than the Tiny-YOLOv3 networks. Therefore, in subsequent analysis, only the MobileNet-SSD networks will be used.

One other key observation is the large standard deviation of the probabilities. This is an indication that the litter objects have varied detectability by the networks. Such detectability can be due to distance, view angle, properties of the litter itself or some other factor. This aspect of variable detectability across multiple litter objects has not been accounted for in this model in order to simplify the model, since variable target types were not used in the simulations.

The detection rates on the Raspberry Pi platforms were not up to the 50 frames per second recording used to collect litter data (inference time shown in Table 5.1 are much longer than 0.02 second). To account for this mismatch in frame rate, the YOLOv3, Tiny-YOLOv3 and MobileNet-SSD detections data were resampled at frame rates measured on the Raspberry Pi platforms. This was done as a post processing step, where only a subset of the detections by each network were used for computing the model probabilities for each deep neural network. The choice of which detections should be utilised in model development or not was based on the frame rates measured on the Raspberry Pi platforms. The resulting data were used to compute new probability values for P_{u2s}, P_{s2s} and P_s for the new frame rates. The probabilities resulting from this resampling step are shown in Table 5.3, for the MobileNet-SSD 124×124 and 220×220, whose names were shortened to mSSD124 and mSSD220 respectively. The registered observation is that as frame rates reduced, the P_{u2s} increased, while P_{s2s} decreased. Using the Raspberry Pi frame rates resulted in an increase in the time between two consecutive frames and, therefore, increasing the dissimilarity between them. Thus, it is expected for P_{u2s} to increase because the relatively substantial change between the frames improved a network's chances of detecting litter objects it failed to detect in preceding frames. Conversely, the increased dissimilarity between two consecutive frames negatively impacted a network's ability to detect a litter it successfully detected in the preceding frame, thus, causing a decrease in the P_{s2s} probability. The probability of detection (P_s), however, remained fairly constant for all frame rates tested. P_s remaining fairly constant is an expected characteristic because at a fundamental level, the detections on each frame is independent of those of other frames. Since P_s is a reflection of how well the network detects litter, resampling the data does not affect its value in any significant way.

The False Positive data was not used for developing the probabilistic vision model. This was done because the False Positive data appeared for only short periods, with more

Table 5.3 Modelling data from testing MobileNet-SSD on resampled data based on frame rates observed on the Raspberry Pi 4 (Pi4) and Erlebrain (Pi3) platforms.

	Pi4		Pi3	
	mSSD124	mSSD220	mSSD124	mSSD220
FPS	14.66	6.49	2.13	0.81
seen	20194	13530	2908	1672
seen2seen	15988	10443	1849	942
seen2unseen	4054	2772	657	231
unseen	30271	8797	4438	1123
unseen2seen	4168	2913	1002	476
unseen2unseen	25576	5520	3159	467
never seen	14	3	54	42
always seen	0	3	12	148
nlitter	679	679	679	679
P_{s2s}	0.7006 ± 0.2254	0.7364 ± 0.1971	0.5823 ± 0.3715	0.5962 ± 0.4440
P_{u2s}	0.2038 ± 0.1686	0.4955 ± 0.2905	0.3685 ± 0.3042	0.7081 ± 0.3450
P_s	0.4081 ± 0.2316	0.6127 ± 0.2185	0.4064 ± 0.2475	0.6079 ± 0.2943

than 90% of this data appearing for less than 10 frames. The mean and standard deviation of detection probability, P_s, computed from the False Positive data were 0.0566 ± 0.0193, 0.0710 ± 0.0185, 0.0794 ± 0.0948 and 0.1202 ± 0.1631 for mSSD124, mSSD220, Tiny-YOLOv3 128×128 and 224×224 respectively. These P_s values are low in comparison to those shown in Table 5.2. Thus, in the interest of simplifying the model, the occurrence of false positives in the vision model was not included. This decision helped to focus on studying the scenario where a robot in the swarm probabilistically fails to detect targets in its local environment.

5.6 Model Validation

A Monte Carlo simulation was performed using the model probabilities in order to validate its correlation with the experimental data. The metric used for this validation is the probability of detection (P_s). In the Monte Carlo simulation, the mean P_{u2s} and P_{s2s} probabilities were applied to 100 virtual litter objects for 1000 time steps. At each time step, if a litter was detected, it was assigned a value of 1, otherwise 0 was assigned. The modelling steps in Section 5.5 were used to compute the P_{u2s}, P_{s2s} and P_s probability values. Since P_{u2s} and P_{s2s} were used to as input to the simulation, it would be expected that the Monte Carlo simulation values would correspond with those of the experimental data. However, the goal is to check if these model probabilities result would approximate the P_s obtained from the experimental data.

Table 5.4 Comparison of the model probabilities of Tables 5.2 and 5.3 with Monte Carlo simulations using the corresponding P_{u2s} and P_{s2s} probabilities.

	50fps		Pi4		Pi3	
	mSSD124	mSSD220	mSSD124	mSSD220	mSSD124	mSSD220
Vision Modelling from Experimental Data						
P_{s2s}	0.7896 ± 0.1824	0.8558 ± 0.1002	0.7006 ± 0.2254	0.7364 ± 0.1971	0.5823 ± 0.3715	0.5962 ± 0.4440
P_{u2s}	0.1293 ± 0.1031	0.2567 ± 0.1586	0.2038 ± 0.1686	0.4955 ± 0.2905	0.3685 ± 0.3042	0.7081 ± 0.3450
P_s	0.4078 ± 0.2304	0.6104 ± 0.2156	0.4081 ± 0.2316	0.6127 ± 0.2185	0.4064 ± 0.2475	0.6079 ± 0.2943
Monte Carlo Simulation of Probabilities						
P_{s2s}	0.7871 ± 0.0195	0.8560 ± 0.0150	0.6995 ± 0.0229	0.7351 ± 0.0181	0.5835 ± 0.0214	0.5952 ± 0.0189
P_{u2s}	0.1301 ± 0.0124	0.2624 ± 0.0242	0.2024 ± 0.0164	0.4952 ± 0.0251	0.3654 ± 0.0188	0.7133 ± 0.0243
P_s	0.3794 ± 0.0301	0.6445 ± 0.0335	0.4022 ± 0.0274	0.6514 ± 0.0202	0.4673 ± 0.0171	0.6380 ± 0.0138

Table 5.4 shows the resulting P_{s2s}, P_{u2s} and P_s from the modelling process described in Section 5.5 in the first three rows of Table 5.4 and their corresponding probabilities from the Monte Carlo experiments in the bottom three rows of Table 5.4. The probability of detection, P_s, resulting from the Monte Carlo simulation shows reasonable agreement with those computed from the vision experimental data. The P_s estimate from the Monte Carlo simulation shows that they were consistent with analysis of the vision experimental data. A maximum deviation of 15% from the vision experimental data was obtained from the Monte Carlo simulation for the Raspberry Pi 4 model probabilities using the MobileNet-SSD 220×220 network. Nonetheless, it is useful to note that P_{u2s} and P_{s2s} underestimated the P_s value in the 50 fps mSSD124 column, closely approximates the P_s in the Pi4, mSSD124 column, and over estimated the probabilities in all other columns. These underestimates and overestimates could be due to some modelling errors cascaded through the steps or variability in the detectability of specific litter objects which caused large standard deviation in the P_s value computed for the experimental data.

5.7 Simulation using Network Models

Gazebo simulations were conducted using P_{s2s} and P_{u2s} model parameters based on the MobileNet-SSD detector to investigate the effect(s), if any, of imperfect vision on foraging performance. The Tiny-YOLOv3 model was not used because it had low values of P_s at similar inference time with MobileNet-SSD. A swarm size of 36 robots were tested on the 10 world setups. The robots' field of view and detection distance were modified to 120° and

5 metres respectively to match the parameters used for the litter detection experiments. Each simulation setup was repeated 30 times and the chemotaxis parameters of $a_m = 4$, $a_d = 100$, $r_m = 1$ and $d_d = 10$ were used for N0-Q1 and N100-Q40 versions of RepAtt. Four sets of simulations were conducted in this section:

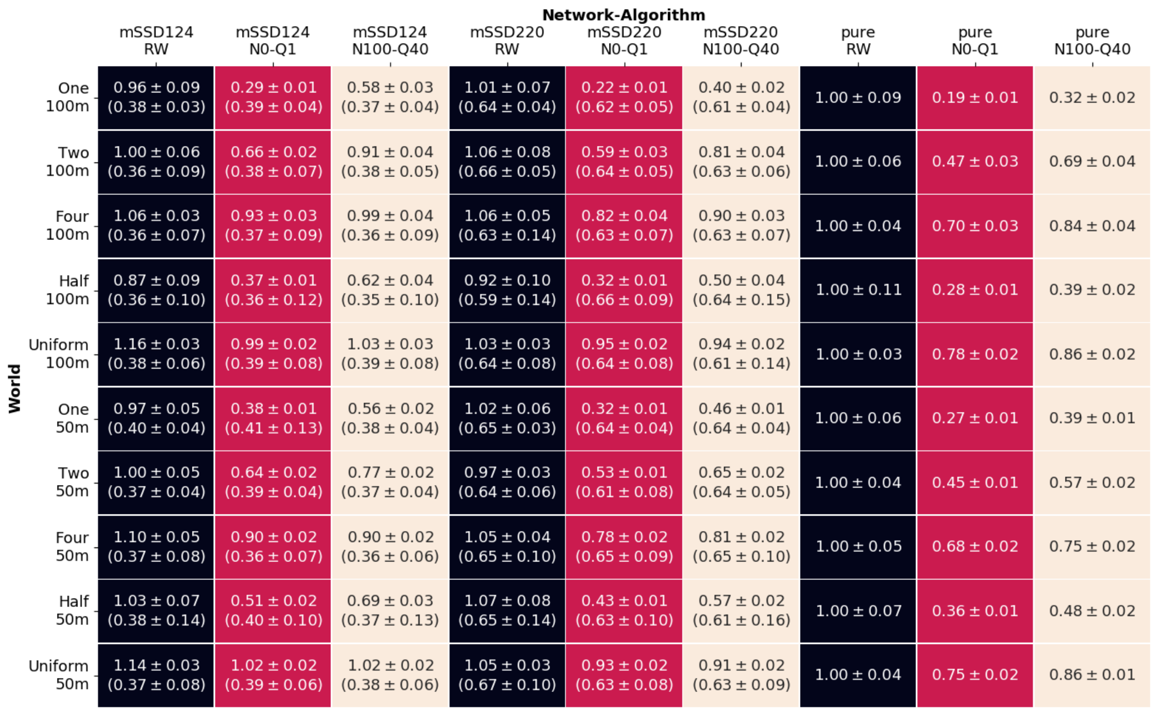
1. Simulation of swarm robots with perfect vision system that was updated at every time step. This was added as a baseline for understanding how imperfect vision affects Random Walk, N0-Q1 and N100-Q40.
2. Swarm robots with imperfect vision using model probabilities obtained from the analysis of all frames with robots vision system at every time step (Section 5.7.1).
3. Swarm robots with imperfect vision using model probabilities obtained from resampling the frames based on Raspberry Pi 4 inference time, and robots update their vision system based on the detector's inference time (Section 5.7.2).
4. Swarm robots with imperfect vision using model probabilities obtained from resampling the frames based on Raspberry Pi 3 inference time, and robots update their vision system based on the detector's inference time (Section 5.7.3).

In Section 5.7.4, the effect of vision update rate has on Random Walk, N0-Q1 and N100-Q40 will be discussed based on the results presented in Sections 5.7.1, 5.7.2 and 5.7.3.

5.7.1 Full frame rate model

For this simulation, modelling probabilities in Table 5.2 were used, where $P_{s2s} = 0.7896$ and $P_{u2s} = 0.1293$ were for mSSD124, and $P_{s2s} = 0.8558$ and $P_{u2s} = 0.2567$ for mSSD220. The robots updated their vision system at every time step (which was 40 Hz, based on simulation step size). The simulation results for using these MobileNet-SSD vision models on Random Walk, N0-Q1 and N100-Q40 are shown in Fig. 5.10, where the last three columns are versions of the algorithms with perfect vision that updated the robot's view at every time step.

The results in Fig. 5.10 show that with imperfect vision system, RepAtt still improved the swarm's coordination when foraging with perfect (N0-Q1) and noisy communication (N100-Q40). As observed in previous simulations, greater improvements were recorded in highly clustered environments than in less clustered ones. Introducing the imperfect vision model into Random Walk algorithm did not affect swarm foraging performance, except in the uniform worlds for mSSD124 vision model where there were increments of 16% and 14% in foraging times for Uniform100m and Uniform50m worlds respectively.

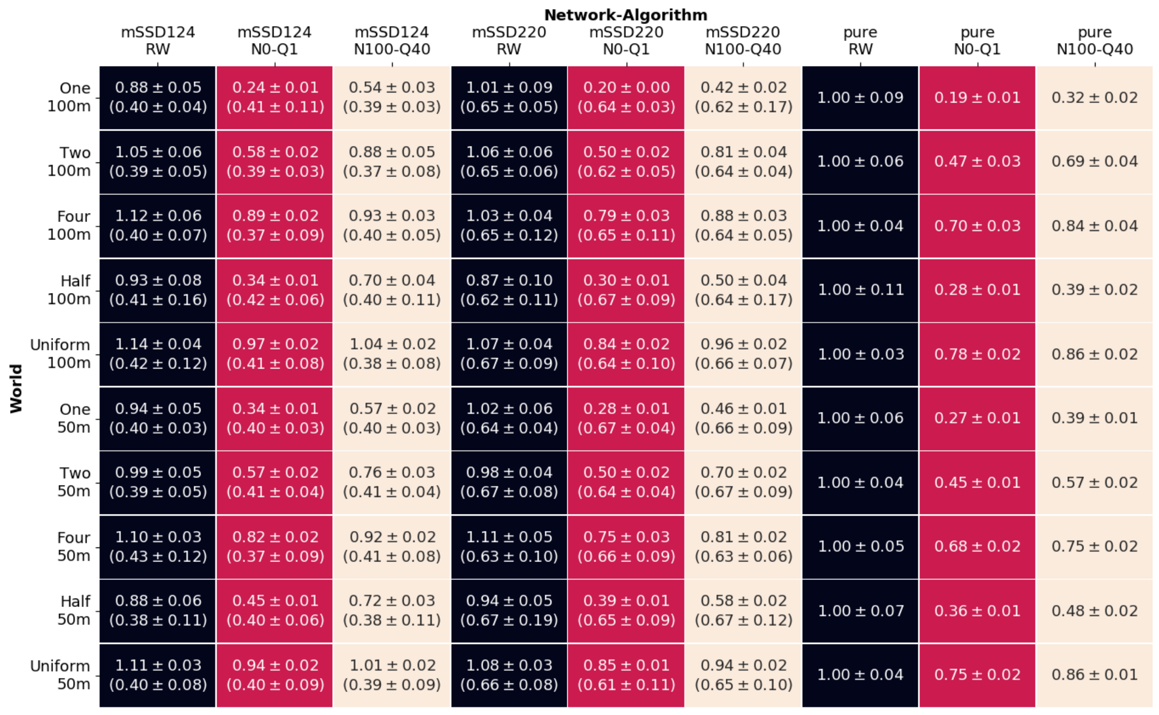


World	mSSD124 RW	mSSD124 N0-Q1	mSSD124 N100-Q40	mSSD220 RW	mSSD220 N0-Q1	mSSD220 N100-Q40	pure RW	pure N0-Q1	pure N100-Q40
One 100m	0.96 ± 0.09 (0.38 ± 0.03)	0.29 ± 0.01 (0.39 ± 0.04)	0.58 ± 0.03 (0.37 ± 0.04)	1.01 ± 0.07 (0.64 ± 0.04)	0.22 ± 0.01 (0.62 ± 0.05)	0.40 ± 0.02 (0.61 ± 0.04)	1.00 ± 0.09	0.19 ± 0.01	0.32 ± 0.02
Two 100m	1.00 ± 0.06 (0.36 ± 0.09)	0.66 ± 0.02 (0.38 ± 0.07)	0.91 ± 0.04 (0.38 ± 0.05)	1.06 ± 0.08 (0.66 ± 0.05)	0.59 ± 0.03 (0.64 ± 0.05)	0.81 ± 0.04 (0.63 ± 0.06)	1.00 ± 0.06	0.47 ± 0.03	0.69 ± 0.04
Four 100m	1.06 ± 0.03 (0.36 ± 0.07)	0.93 ± 0.03 (0.37 ± 0.09)	0.99 ± 0.04 (0.36 ± 0.09)	1.06 ± 0.05 (0.63 ± 0.14)	0.82 ± 0.04 (0.63 ± 0.07)	0.90 ± 0.03 (0.63 ± 0.07)	1.00 ± 0.04	0.70 ± 0.03	0.84 ± 0.04
Half 100m	0.87 ± 0.09 (0.36 ± 0.10)	0.37 ± 0.01 (0.36 ± 0.12)	0.62 ± 0.04 (0.35 ± 0.10)	0.92 ± 0.10 (0.59 ± 0.14)	0.32 ± 0.01 (0.66 ± 0.09)	0.50 ± 0.04 (0.64 ± 0.15)	1.00 ± 0.11	0.28 ± 0.01	0.39 ± 0.02
Uniform 100m	1.16 ± 0.03 (0.38 ± 0.06)	0.99 ± 0.02 (0.39 ± 0.08)	1.03 ± 0.03 (0.39 ± 0.08)	1.03 ± 0.03 (0.64 ± 0.08)	0.95 ± 0.02 (0.64 ± 0.08)	0.94 ± 0.02 (0.61 ± 0.14)	1.00 ± 0.03	0.78 ± 0.02	0.86 ± 0.02
One 50m	0.97 ± 0.05 (0.40 ± 0.04)	0.38 ± 0.01 (0.41 ± 0.13)	0.56 ± 0.02 (0.38 ± 0.04)	1.02 ± 0.06 (0.65 ± 0.03)	0.32 ± 0.01 (0.64 ± 0.04)	0.46 ± 0.01 (0.64 ± 0.04)	1.00 ± 0.06	0.27 ± 0.01	0.39 ± 0.01
Two 50m	1.00 ± 0.05 (0.37 ± 0.04)	0.64 ± 0.02 (0.39 ± 0.04)	0.77 ± 0.02 (0.37 ± 0.04)	0.97 ± 0.03 (0.64 ± 0.06)	0.53 ± 0.01 (0.61 ± 0.08)	0.65 ± 0.02 (0.64 ± 0.05)	1.00 ± 0.04	0.45 ± 0.01	0.57 ± 0.02
Four 50m	1.10 ± 0.05 (0.37 ± 0.08)	0.90 ± 0.02 (0.36 ± 0.07)	0.90 ± 0.02 (0.36 ± 0.06)	1.05 ± 0.04 (0.65 ± 0.10)	0.78 ± 0.02 (0.65 ± 0.09)	0.81 ± 0.02 (0.65 ± 0.10)	1.00 ± 0.05	0.68 ± 0.02	0.75 ± 0.02
Half 50m	1.03 ± 0.07 (0.38 ± 0.14)	0.51 ± 0.02 (0.40 ± 0.10)	0.69 ± 0.03 (0.37 ± 0.13)	1.07 ± 0.08 (0.65 ± 0.14)	0.43 ± 0.01 (0.63 ± 0.10)	0.57 ± 0.02 (0.61 ± 0.16)	1.00 ± 0.07	0.36 ± 0.01	0.48 ± 0.02
Uniform 50m	1.14 ± 0.03 (0.37 ± 0.08)	1.02 ± 0.02 (0.39 ± 0.06)	1.02 ± 0.02 (0.38 ± 0.06)	1.05 ± 0.03 (0.67 ± 0.10)	0.93 ± 0.02 (0.63 ± 0.08)	0.91 ± 0.02 (0.63 ± 0.09)	1.00 ± 0.04	0.75 ± 0.02	0.86 ± 0.01

Fig. 5.10 The swarm foraging performance and corresponding P_s shown in brackets. Foraging times were normalised by the performance of Random Walk algorithm, where robots used pure vision to forage. Full frame rate model probabilities were used and columns were colour-coded based on Random Walk, N0-Q1, N100-Q40 algorithm types.

This indicates that Random Walk was robust to the change in robots' vision model. This robustness is due to robots' independent search for targets without using memory to return to previous foraging sites. This meant that robots use a large proportion of their foraging times searching for targets and spent significantly shorter time in the other states (acquiring and homing). The reduction in foraging performance of the robots due to poorer vision quality was small in comparison to the time robots spend searching for targets using Random Walk, therefore, leading to negligible change in the swarm's foraging time. It would be expected that as the robots' vision system continues to degrade, it would reach a point where the poor vision system of the robots begin to impose a significant impact on the swarms' foraging performance. This data shows that that point is much lower for a swarm of 36 robots using Random Walk to forage targets in clustered environments. However, the vision system based on the mSSD124 model shows significant change in foraging time for the swarm in the Uniform100m and Uniform50m worlds, where foraging times increased by 16% and 14% respectively.

For the N0-Q1 and N100-Q40 setups of RepAtt, the quality of the robots vision impacts their communication because it causes them to send 'incorrect' signals to the rest of the



World	mSSD124 RW	mSSD124 N0-Q1	mSSD124 N100-Q40	mSSD220 RW	mSSD220 N0-Q1	mSSD220 N100-Q40	pure RW	pure N0-Q1	pure N100-Q40
One 100m	0.88 ± 0.05 (0.40 ± 0.04)	0.24 ± 0.01 (0.41 ± 0.11)	0.54 ± 0.03 (0.39 ± 0.03)	1.01 ± 0.09 (0.65 ± 0.05)	0.20 ± 0.00 (0.64 ± 0.03)	0.42 ± 0.02 (0.62 ± 0.17)	1.00 ± 0.09	0.19 ± 0.01	0.32 ± 0.02
Two 100m	1.05 ± 0.06 (0.39 ± 0.05)	0.58 ± 0.02 (0.39 ± 0.03)	0.88 ± 0.05 (0.37 ± 0.08)	1.06 ± 0.06 (0.65 ± 0.06)	0.50 ± 0.02 (0.62 ± 0.05)	0.81 ± 0.04 (0.64 ± 0.04)	1.00 ± 0.06	0.47 ± 0.03	0.69 ± 0.04
Four 100m	1.12 ± 0.06 (0.40 ± 0.07)	0.89 ± 0.02 (0.37 ± 0.09)	0.93 ± 0.03 (0.40 ± 0.05)	1.03 ± 0.04 (0.65 ± 0.12)	0.79 ± 0.03 (0.65 ± 0.11)	0.88 ± 0.03 (0.64 ± 0.05)	1.00 ± 0.04	0.70 ± 0.03	0.84 ± 0.04
Half 100m	0.93 ± 0.08 (0.41 ± 0.16)	0.34 ± 0.01 (0.42 ± 0.06)	0.70 ± 0.04 (0.40 ± 0.11)	0.87 ± 0.10 (0.62 ± 0.11)	0.30 ± 0.01 (0.67 ± 0.09)	0.50 ± 0.04 (0.64 ± 0.17)	1.00 ± 0.11	0.28 ± 0.01	0.39 ± 0.02
Uniform 100m	1.14 ± 0.04 (0.42 ± 0.12)	0.97 ± 0.02 (0.41 ± 0.08)	1.04 ± 0.02 (0.38 ± 0.08)	1.07 ± 0.04 (0.67 ± 0.09)	0.84 ± 0.02 (0.64 ± 0.10)	0.96 ± 0.02 (0.66 ± 0.07)	1.00 ± 0.03	0.78 ± 0.02	0.86 ± 0.02
One 50m	0.94 ± 0.05 (0.40 ± 0.03)	0.34 ± 0.01 (0.40 ± 0.03)	0.57 ± 0.02 (0.40 ± 0.03)	1.02 ± 0.06 (0.64 ± 0.04)	0.28 ± 0.01 (0.67 ± 0.04)	0.46 ± 0.01 (0.66 ± 0.09)	1.00 ± 0.06	0.27 ± 0.01	0.39 ± 0.01
Two 50m	0.99 ± 0.05 (0.39 ± 0.05)	0.57 ± 0.02 (0.41 ± 0.04)	0.76 ± 0.03 (0.41 ± 0.04)	0.98 ± 0.04 (0.67 ± 0.08)	0.50 ± 0.02 (0.64 ± 0.04)	0.70 ± 0.02 (0.67 ± 0.09)	1.00 ± 0.04	0.45 ± 0.01	0.57 ± 0.02
Four 50m	1.10 ± 0.03 (0.43 ± 0.12)	0.82 ± 0.02 (0.37 ± 0.09)	0.92 ± 0.02 (0.41 ± 0.08)	1.11 ± 0.05 (0.63 ± 0.10)	0.75 ± 0.03 (0.66 ± 0.09)	0.81 ± 0.02 (0.63 ± 0.06)	1.00 ± 0.05	0.68 ± 0.02	0.75 ± 0.02
Half 50m	0.88 ± 0.06 (0.38 ± 0.11)	0.45 ± 0.01 (0.40 ± 0.06)	0.72 ± 0.03 (0.38 ± 0.11)	0.94 ± 0.05 (0.67 ± 0.19)	0.39 ± 0.01 (0.65 ± 0.09)	0.58 ± 0.02 (0.67 ± 0.12)	1.00 ± 0.07	0.36 ± 0.01	0.48 ± 0.02
Uniform 50m	1.11 ± 0.03 (0.40 ± 0.08)	0.94 ± 0.02 (0.40 ± 0.09)	1.01 ± 0.02 (0.39 ± 0.09)	1.08 ± 0.03 (0.66 ± 0.08)	0.85 ± 0.01 (0.61 ± 0.11)	0.94 ± 0.02 (0.65 ± 0.10)	1.00 ± 0.04	0.75 ± 0.02	0.86 ± 0.01

Fig. 5.11 The swarm foraging performance and corresponding P_s shown in brackets. Foraging times were normalised by the performance of Random Walk algorithm, where robots used pure vision to forage. Raspberry Pi 4 update rate model probabilities were used and columns were colour-coded based on Random Walk, N0-Q1, N100-Q40 algorithm types.

swarm based on their faulty perception of their environment. This caused a noticeable reduction in foraging performance as vision quality degrade. Thus, for the 10 world setups in Fig. 5.10, swarm foraging with mSSD124 was consistently outperformed by mSSD220, which in turn was outperformed by robots using perfect vision during foraging. The results for N0-Q1 and N100-Q40 were, thus, consistent with the sort of behaviour expected from a swarm with reduced quality of local sensing.

5.7.2 Raspberry Pi 4 model

For this simulation, modelling probabilities in Table 5.3 were used, where $P_{s2s} = 0.7006$ and $P_{u2s} = 0.2038$ were for mSSD124, and $P_{s2s} = 0.7364$ and $P_{u2s} = 0.4955$ were for mSSD220. The robots updated their vision at a rate of 14.66 Hz and 6.49 Hz for mSSD124 and mSSD220 respectively. The simulation results for using these MobileNet-SSD vision models on Random Walk, N0-Q1 and N100-Q40 are shown in Fig. 5.11, where the last three columns are versions of the algorithms with perfect vision that updated a robot's view at every time step.

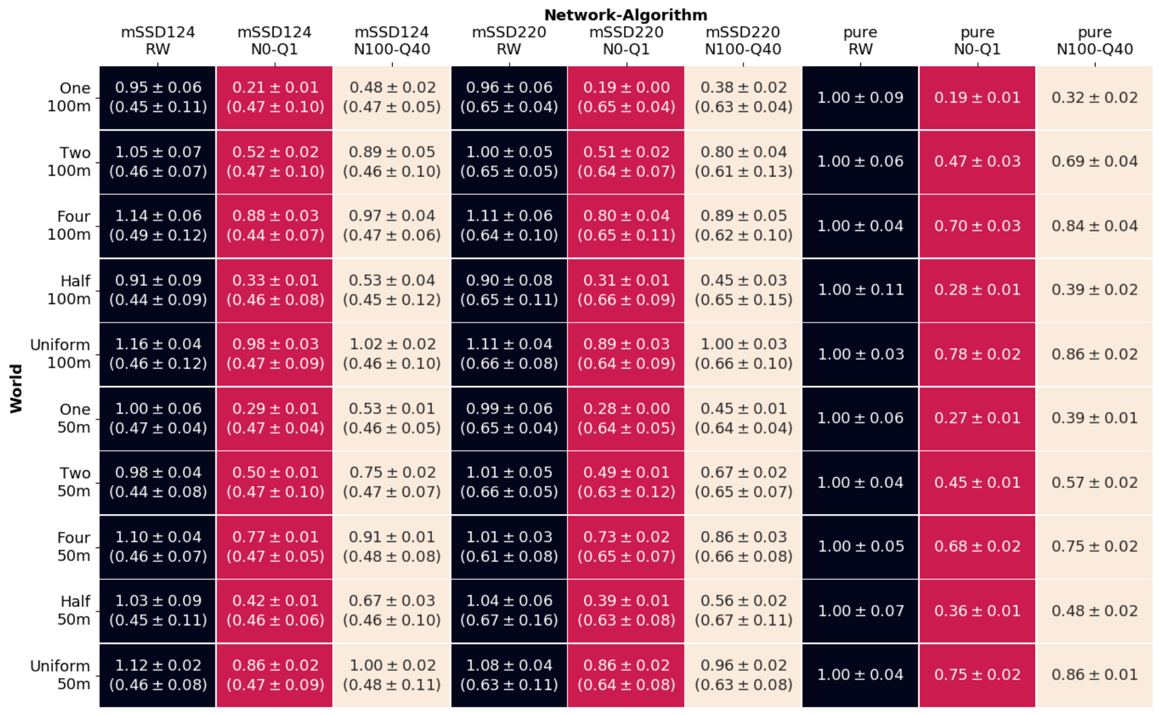
The trends in performance of Random Walk, N0-Q1 and N100-Q40 in Fig. 5.11 is similar to those already discussed in Section 5.7.1 for Fig. 5.10. One major difference observed for N0-Q1 and N100-Q40 is the performance improvement in foraging when using mSSD220 vision model in place of mSSD124 model has reduced. For example, N0-Q1 foraging time improved by 34.48% in the One100m in Fig. 5.10 when the vision system changed from mSSD124 to mSSD220, while for the same setting in Fig. 5.11, the improvement was 16.67%. Similar patterns were observed for N0-Q1 and N100-Q40 for the remaining 9 world setups. Seeing that all simulation parameters except vision update rates were the same in both scenarios and the P_s probabilities were similar in both figures, this deterioration in performance can be attributed to the change in the vision update rates. The results suggest that the faster vision update rate of mSSD124 (that is, 14.66 Hz) helps it compensate for its lower P_s accuracy in comparison to mSSD220 that has increased detection accuracy, with a halved vision update rate of 6.49 Hz. The second major observation when comparing Figs. 5.10 and 5.11 is the improvement in foraging times of the latter for N0-Q1 and N100-Q40 RepAtt algorithm setup. This aspect will be discussed separately in Section 5.7.4.

5.7.3 Raspberry Pi 3 model

For this simulation, modelling probabilities in Table 5.3 were used, where $P_{s2s} = 0.5823$ and $P_{u2s} = 0.3685$ were for mSSD124, and $P_{s2s} = 0.5962$ and $P_{u2s} = 0.7081$ were for mSSD220. The robots updated their vision system at a rate of 2.13 Hz for mSSD124 and 0.81 Hz for mSSD220. The simulation results for using these MobileNet-SSD vision models on Random Walk, N0-Q1 and N100-Q40 are shown in Fig. 5.12, where the last three columns are versions of the algorithms with perfect vision that updated the robots view at every time step.

The Raspberry Pi 3 based vision model results in Fig. 5.12 were consistent with what was discussed in Sections 5.7.1 and 5.7.2. The performance difference between mSSD124 and mSSD220 has diminished further such that in 6 of the 10 world setups, the foraging times were hardly distinguishable. For N100-Q40, the performance difference between mSSD124 and mSSD220 were minimal for the uniform distribution worlds (Uniform100m and Uniform50m).

One key observation in Fig. 5.12 (also reasonable agreement with the Monte Carlo simulation in the results shown in Table 5.4) is that probability of detection, P_s were generally larger than those derived from analysing the MobileNet-SSD vision experiment data. These values suggest that the probabilistic vision model does not ‘perfectly’ represent the experiment data. The difference between the modelled P_s and the value computed from the vision experiments increases as the frame rate decreases. It is still unclear whether this deviation



World	Network-Algorithm								
	mSSD124 RW	mSSD124 N0-Q1	mSSD124 N100-Q40	mSSD220 RW	mSSD220 N0-Q1	mSSD220 N100-Q40	pure RW	pure N0-Q1	pure N100-Q40
One 100m	0.95 ± 0.06 (0.45 ± 0.11)	0.21 ± 0.01 (0.47 ± 0.10)	0.48 ± 0.02 (0.47 ± 0.05)	0.96 ± 0.06 (0.65 ± 0.04)	0.19 ± 0.00 (0.65 ± 0.04)	0.38 ± 0.02 (0.63 ± 0.04)	1.00 ± 0.09	0.19 ± 0.01	0.32 ± 0.02
Two 100m	1.05 ± 0.07 (0.46 ± 0.07)	0.52 ± 0.02 (0.47 ± 0.10)	0.89 ± 0.05 (0.46 ± 0.10)	1.00 ± 0.05 (0.65 ± 0.05)	0.51 ± 0.02 (0.64 ± 0.07)	0.80 ± 0.04 (0.61 ± 0.13)	1.00 ± 0.06	0.47 ± 0.03	0.69 ± 0.04
Four 100m	1.14 ± 0.06 (0.49 ± 0.12)	0.88 ± 0.03 (0.44 ± 0.07)	0.97 ± 0.04 (0.47 ± 0.06)	1.11 ± 0.06 (0.64 ± 0.10)	0.80 ± 0.04 (0.65 ± 0.11)	0.89 ± 0.05 (0.62 ± 0.10)	1.00 ± 0.04	0.70 ± 0.03	0.84 ± 0.04
Half 100m	0.91 ± 0.09 (0.44 ± 0.09)	0.33 ± 0.01 (0.46 ± 0.08)	0.53 ± 0.04 (0.45 ± 0.12)	0.90 ± 0.08 (0.65 ± 0.11)	0.31 ± 0.01 (0.66 ± 0.09)	0.45 ± 0.03 (0.65 ± 0.15)	1.00 ± 0.11	0.28 ± 0.01	0.39 ± 0.02
Uniform 100m	1.16 ± 0.04 (0.46 ± 0.12)	0.98 ± 0.03 (0.47 ± 0.09)	1.02 ± 0.02 (0.46 ± 0.10)	1.11 ± 0.04 (0.66 ± 0.08)	0.89 ± 0.03 (0.64 ± 0.09)	1.00 ± 0.03 (0.66 ± 0.10)	1.00 ± 0.03	0.78 ± 0.02	0.86 ± 0.02
One 50m	1.00 ± 0.06 (0.47 ± 0.04)	0.29 ± 0.01 (0.47 ± 0.04)	0.53 ± 0.01 (0.46 ± 0.05)	0.99 ± 0.06 (0.65 ± 0.04)	0.28 ± 0.00 (0.64 ± 0.05)	0.45 ± 0.01 (0.64 ± 0.04)	1.00 ± 0.06	0.27 ± 0.01	0.39 ± 0.01
Two 50m	0.98 ± 0.04 (0.44 ± 0.08)	0.50 ± 0.01 (0.47 ± 0.10)	0.75 ± 0.02 (0.47 ± 0.07)	1.01 ± 0.05 (0.66 ± 0.05)	0.49 ± 0.01 (0.63 ± 0.12)	0.67 ± 0.02 (0.65 ± 0.07)	1.00 ± 0.04	0.45 ± 0.01	0.57 ± 0.02
Four 50m	1.10 ± 0.04 (0.46 ± 0.07)	0.77 ± 0.01 (0.47 ± 0.05)	0.91 ± 0.01 (0.48 ± 0.08)	1.01 ± 0.03 (0.61 ± 0.08)	0.73 ± 0.02 (0.65 ± 0.07)	0.86 ± 0.03 (0.66 ± 0.08)	1.00 ± 0.05	0.68 ± 0.02	0.75 ± 0.02
Half 50m	1.03 ± 0.09 (0.45 ± 0.11)	0.42 ± 0.01 (0.46 ± 0.06)	0.67 ± 0.03 (0.46 ± 0.10)	1.04 ± 0.06 (0.67 ± 0.16)	0.39 ± 0.01 (0.63 ± 0.08)	0.56 ± 0.02 (0.67 ± 0.11)	1.00 ± 0.07	0.36 ± 0.01	0.48 ± 0.02
Uniform 50m	1.12 ± 0.02 (0.46 ± 0.08)	0.86 ± 0.02 (0.47 ± 0.09)	1.00 ± 0.02 (0.48 ± 0.11)	1.08 ± 0.04 (0.63 ± 0.11)	0.86 ± 0.02 (0.64 ± 0.08)	0.96 ± 0.02 (0.63 ± 0.08)	1.00 ± 0.04	0.75 ± 0.02	0.86 ± 0.01

Fig. 5.12 The swarm foraging performance and corresponding P_s shown in brackets. Foraging times were normalised by the performance of Random Walk algorithm, where robots used pure vision to forage. Raspberry Pi 3 vision update rate model probabilities were used and columns were colour-coded based on Random Walk, N0-Q1, N100-Q40 algorithm types.

from experimental data was because the model was derived from a less-than-perfect ground truth data or due to some other parameter not included in the model.

5.7.4 Effect of vision update rate on swarm foraging

For a particular MobileNet-SSD target detection model, one of the interesting observations as the vision update rate reduces for N0-Q1, and to a lesser extent in N100-Q40, was that the foraging performance improved. Using the One100m world setup and vision model of mSSD124 as an example, in Fig. 5.10, the normalised foraging time was $0.29 \pm 0.01 (0.39 \pm 0.04)$, where the P_s is shown in parentheses. The corresponding foraging times for Fig. 5.11 and 5.12 are $0.24 \pm 0.01 (0.41 \pm 0.11)$ and $0.21 \pm 0.01 (0.47 \pm 0.10)$ respectively. Although the improvement in foraging time can be attributed to the increase in P_s as observed in Figs. 5.10, 5.11 and 5.12, it should also be noted that the P_s values all fall within the same standard deviation range, suggesting that the values were not so different from each other. One factor that could have had a major impact on the swarm foraging performance was the vision update rate, which was (for mSSD124) 40 Hz in Fig. 5.10, 14.66 Hz in Fig. 5.11 and 2.13 Hz in Fig.

5.12. Because of the varying P_s as the vision update rate decreased, the factors that caused this unexpected change in foraging performance was unclear. In Section 5.8, an extensive study that involved 196 combinations of P_{s2s} and P_{u2s} probabilities and 3 vision update rates were conducted. This helped to further explore the impact the vision model probabilities have on the performance of swarm of foraging robots.

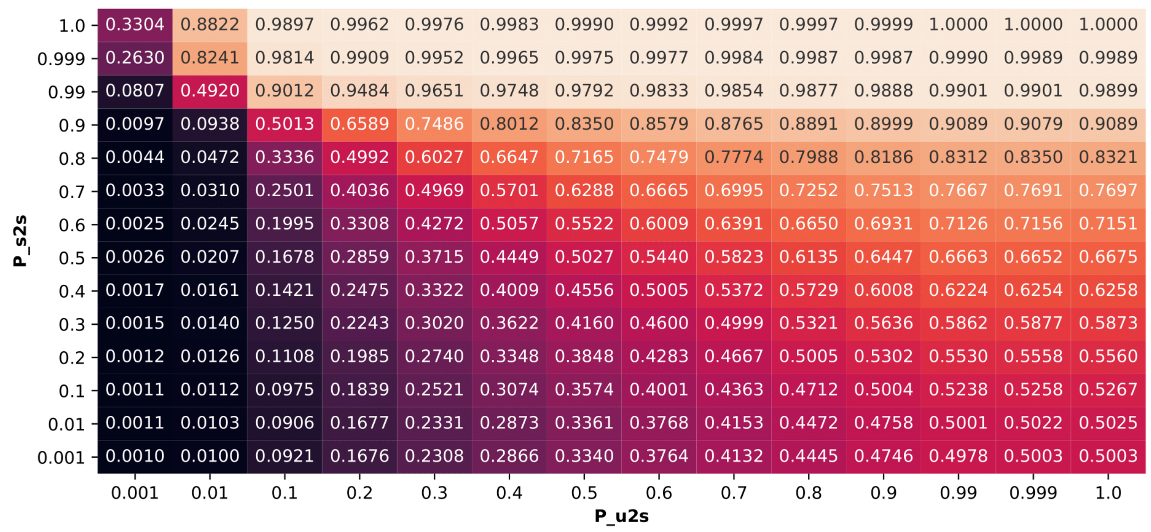
5.8 Exploring the Effects of Model Probabilities

This section explores the influence of model probabilities (P_{u2s} and P_{s2s}) on a robot's probability of detecting objects (P_s) and the robot's use of communication signals during RepAtt and swarm foraging performance. In all experiments, the P_{u2s} and P_{s2s} were varied from 0.001 to 1.0 to simulate variation in accuracy of the object detector while the vision update rate was varied from 40 Hz to 4 Hz to 1 Hz to simulate different qualities of computer vision hardware. These set of simulations investigate the trade-off between the quality of the object detection system and the robot's processing power and how these choices affect a swarm of foraging robots.

5.8.1 Effects of P_{u2s} and P_{s2s} on P_s

The Monte Carlo simulation described in the imperfect vision model validation in Section 5.6 was performed for several combinations of P_{u2s} and P_{s2s} for 100,000 time steps. The mean P_s resulting from the combinations of the modelling probabilities are shown in Fig. 5.13. The data shows that increasing P_{s2s} and P_{u2s} increases the P_s probability. When $P_{s2s} = 1$, P_s varied from 0.3304 ($P_{u2s} = 0.001$) and sharply approached 1.0 (when $P_{u2s} > 0.001$). This is a special case where once the object is detected by the agent, it will always be seen by the agent as long as it is within detection range. Thus, in this scenario, $P_s \rightarrow 1.0$ as time approaches infinity as long as $P_{u2s} > 0$ (when $P_{u2s} = 0$, the agent will never detect the object and P_s will always be 0). When $P_{s2s} = 0.001$, P_s gradually increased from 0.001 to 0.5003 as P_{u2s} varied from 0.001 to 1.0. The value of $P_s \rightarrow 0.5$ in this scenario because as $P_{s2s} \rightarrow 0$ the ability of the agent to detect the object for at least two consecutive frames reduces. The special case where $P_{s2s} = 0$ and $P_{u2s} = 1.0$ will lead to $P_s = 0.5$ because the agent is only able to detect the target once every two time steps. When $P_{u2s} = 0.001$, P_s had minor increments until P_{s2s} became greater than 0.99, where a steep rise in P_s was observed. This huge increase in P_s is attributed to the near perfect ability of the agent to retain its previously detected targets.

Along the diagonals of the linear range of 0.1 to 0.9, it is evident that there is a linear relationship between the contributions of P_{s2s} and P_{u2s} to the value of P_s. In addition,



P_s2s \ P_u2s	0.001	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.99	0.999	1.0
1.0	0.3304	0.8822	0.9897	0.9962	0.9976	0.9983	0.9990	0.9992	0.9997	0.9997	0.9999	1.0000	1.0000	1.0000
0.999	0.2630	0.8241	0.9814	0.9909	0.9952	0.9965	0.9975	0.9977	0.9984	0.9987	0.9987	0.9990	0.9989	0.9989
0.99	0.0807	0.4920	0.9012	0.9484	0.9651	0.9748	0.9792	0.9833	0.9854	0.9877	0.9888	0.9901	0.9901	0.9899
0.9	0.0097	0.0938	0.5013	0.6589	0.7486	0.8012	0.8350	0.8579	0.8765	0.8891	0.8999	0.9089	0.9079	0.9089
0.8	0.0044	0.0472	0.3336	0.4992	0.6027	0.6647	0.7165	0.7479	0.7774	0.7988	0.8186	0.8312	0.8350	0.8321
0.7	0.0033	0.0310	0.2501	0.4036	0.4969	0.5701	0.6288	0.6665	0.6995	0.7252	0.7513	0.7667	0.7691	0.7697
0.6	0.0025	0.0245	0.1995	0.3308	0.4272	0.5057	0.5522	0.6009	0.6391	0.6650	0.6931	0.7126	0.7156	0.7151
0.5	0.0026	0.0207	0.1678	0.2859	0.3715	0.4449	0.5027	0.5440	0.5823	0.6135	0.6447	0.6663	0.6652	0.6675
0.4	0.0017	0.0161	0.1421	0.2475	0.3322	0.4009	0.4556	0.5005	0.5372	0.5729	0.6008	0.6224	0.6254	0.6258
0.3	0.0015	0.0140	0.1250	0.2243	0.3020	0.3622	0.4160	0.4600	0.4999	0.5321	0.5636	0.5862	0.5877	0.5873
0.2	0.0012	0.0126	0.1108	0.1985	0.2740	0.3348	0.3848	0.4283	0.4667	0.5005	0.5302	0.5530	0.5558	0.5560
0.1	0.0011	0.0112	0.0975	0.1839	0.2521	0.3074	0.3574	0.4001	0.4363	0.4712	0.5004	0.5238	0.5258	0.5267
0.01	0.0011	0.0103	0.0906	0.1677	0.2331	0.2873	0.3361	0.3768	0.4153	0.4472	0.4758	0.5001	0.5022	0.5025
0.001	0.0010	0.0100	0.0921	0.1676	0.2308	0.2866	0.3340	0.3764	0.4132	0.4445	0.4746	0.4978	0.5003	0.5003

Fig. 5.13 Monte Carlo simulation for 100,000 time steps showing the mean P_s for various combinations of P_{u2s} and P_{s2s}.

these model probabilities complement each other in such a way that the P_s value can be approximated as the average of the P_{u2s} plus P_{s2s} probabilities.

5.8.2 Effects of P_{u2s}, P_{s2s} and vision update rate on RepAtt

A simplified one-dimensional simulation, similar to the version used in Section 3.5 to study the chemotaxis parameters, was employed to study the impact of P_{u2s} and P_{s2s} on RepAtt algorithm. In this simplified simulation setup, it was assumed that a robot acting as the signal source should be broadcasting attraction signal, but will repel instead if it fails to detect the target(s) within its detection proximity (the vision was controlled using P_{u2s} and P_{s2s} probabilities). The listening robot senses the intensity and type (attraction or repulsion) of this signal and uses it to perform chemotaxis-based choice to move forward (toward the signal source) or reverse (away from the signal source). The signal source's choice to broadcast attraction or repulsion was controlled by the P_{u2s} and P_{s2s} values such that combinations with low P_s caused the source to mostly broadcast repulsion while higher values of P_s would cause it to broadcast attraction most times. In addition, the rate at which the signal source applied the P_{u2s} and P_{s2s} probabilities was controlled by a detection frequency of 40 Hz, 4 Hz or 1 Hz to simulate varying inference times of the object detection vision system. The N0-Q1 version of RepAtt with $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ was used for all simulations and each simulation lasted 5,000 seconds.

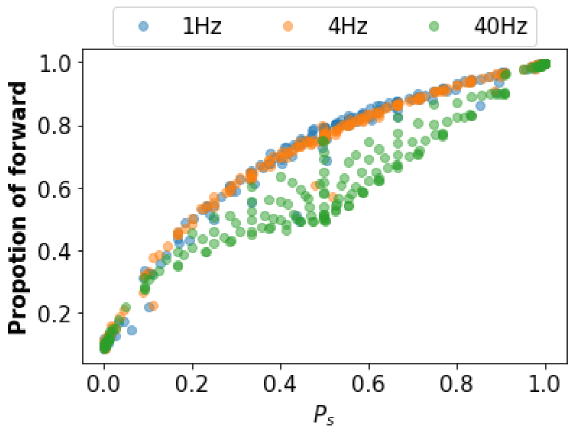
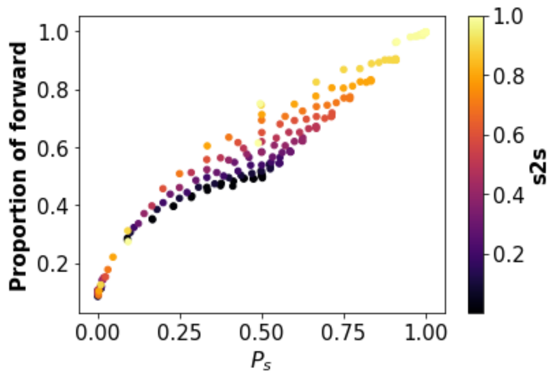


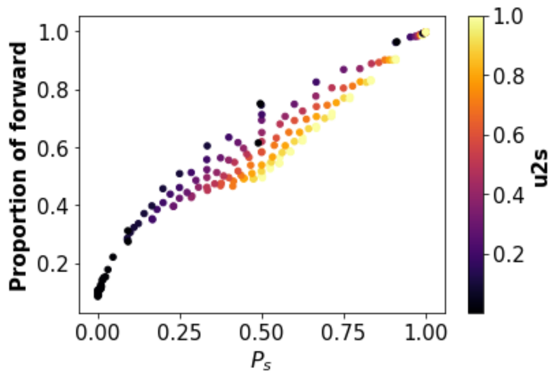
Fig. 5.14 The average proportion of forward movements by a one-dimensional robot as P_s increased. The P_{u2s} and P_{s2s} probabilities were applied at either 1 Hz, 4 Hz or 40 Hz. The robot used N0-Q1 with chemotaxis parameters $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ RepAtt setup to respond to the signal source.

The ratio of forward movements (that is, towards the signal source) by the robot to total distance travelled (sum total of all forward and reverse movements) is shown in Fig. 5.14. The data shows that proportion of forward movements consistently increased with increasing P_s probability for vision update rates of 1 Hz and 4 Hz. For the 40 Hz update rate, the proportion of forward movements were the same as those observed for the lower update rates when $P_s < 0.2$ and $P_s > 0.9$. However, for $0.2 < P_s < 0.9$, there was a noticeable distribution of proportion of forward movements such that many of the P_{u2s} and P_{s2s} combinations for a specific P_s consistently gave lower path lengths. This distribution peaked at $P_s = 0.5$ for 40 Hz and continuously reduced leftwards and rightwards from the $P_s = 0.5$ until its performance approached that of 1 Hz and 4 Hz update rates. A further look at the data for 40 Hz vision update rate indicates that the distribution is due to different combinations of the model probabilities that result in similar P_s (for example, in Fig. 5.13, most combinations of P_{u2s} and P_{s2s} along the top-left to bottom-right diagonal result in $P_s \approxeq 0.5$). In Fig. 5.15, a heat map was used to colour-code the P_{s2s} (in Fig. 5.15a) and P_{u2s} (in Fig. 5.15b) probabilities. The data shows that for a specific P_s value, a high P_{s2s} probability combined with a low P_{u2s} value performed better than combinations where P_{s2s} was low and P_{u2s} was high.

As alluded to in Chapter 3, robots using RepAtt respond to change in intensity of the attraction and repulsion signals. In N0-Q1 setup, robots respond to change between two consecutive time steps. Thus, for the robot to effectively utilise an attraction (or repulsion) signal it needs to sense it for at least two consecutive time steps before it can appropriately adapt its tumble probability. A high P_{s2s} value will help the signal source to continue broadcasting a specific signal, thus reducing the frequency at which it switches between



(a) Coloured according to P_{s2s}

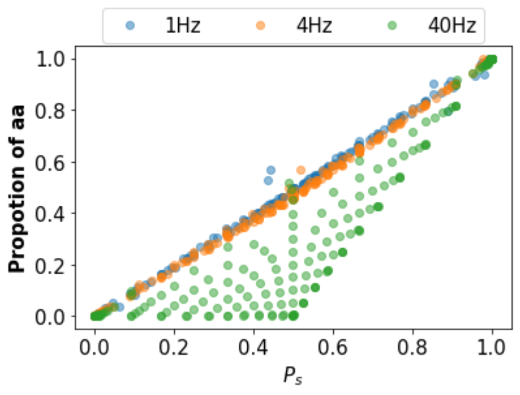


(b) Coloured according to P_{u2s}

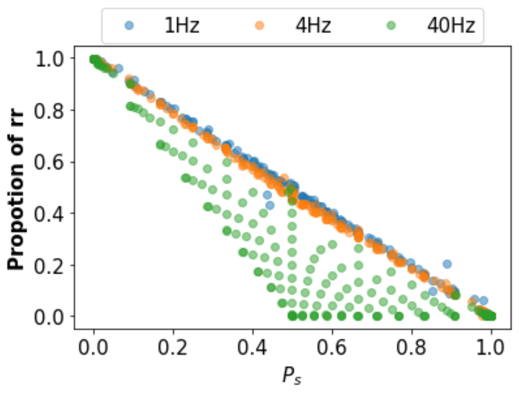
Fig. 5.15 Percentage of forward movement with varying P_{s2s} and P_{u2s} probabilities at 40Hz vision update rate. Colour gradients show the variation in model probabilities.

broadcasting attraction or repulsion signal. The effect on a robot listening to this signal and searching using N0-Q1 RepAtt setup is that the robot will be able to measure the intensity of the attraction or repulsion signal for multiple consecutive time steps. On the other hand, when $P_{s2s} \to 0$, the signal source increases its frequency of switching between broadcasting repulsion or attraction (worst case is when $P_{s2s} = 0$, where it would alternate between attraction and repulsion signal broadcasting). The effect this has on a listening robot is such that it measures an intensity that alternates between zero and some other value (based on listening robot's distance from signal source). It is the frequency of this alternation between attraction and repulsion the robot senses when P_{u2s} is high and P_{s2s} is low that causes the significant reduction in the proportion of forward movements the robot makes toward the signal source.

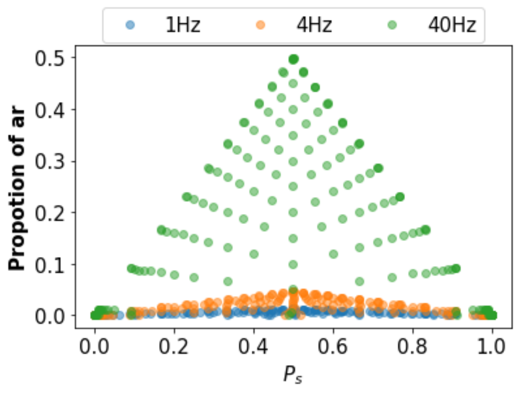
One way to minimise the impact of this alternating effect is to reduce the rate at which the signal source switches between broadcasting attraction and repulsion signals. This was achieved by reducing the vision update rate from 40 Hz to 4 Hz or 1 Hz (which was originally meant to simulate constrained computational environments that will naturally process object detection algorithms at slower rates). The proportion of transitions logged during the simulations are shown in Fig. 5.16. The transitions for vision update rates of 1 Hz and 4 Hz have a near linear relationship with change in P_s: attraction-to-attraction transitions were directly proportional to P_s; repulsion-to-repulsion transitions were inversely proportional to P_s; and attraction-to-repulsion and repulsion-to-attraction transitions varied only slightly as P_s increased. For the 40 Hz update rate, however, the 'spread', due to different combinations of P_{u2s} and P_{s2s} that give a specific P_s value, increased as $P_s \to 0.5$ and decreased beyond 0.5 as $P_s \to 1.0$. At both extremes where P_s is close to 0 or 1, the



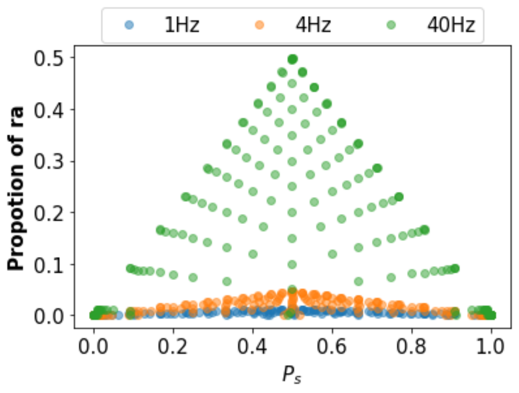
(a) attraction-to-attraction ($\frac{aa}{aa+ar+ra+rr}$)



(b) repulsion-to-repulsion ($\frac{rr}{aa+ar+ra+rr}$)



(c) attraction-to-repulsion ($\frac{ar}{aa+ar+ra+rr}$)



(d) repulsion-to-attraction ($\frac{ra}{aa+ar+ra+rr}$)

Fig. 5.16 The transitions of the signal source when it is updated at 1Hz, 4Hz or 40Hz. The data indicates that for combinations of P_{u2s} and P_{s2s} that result in a particular P_s, the variability of the transitions increases with increase in vision update frequency.

proportion of each of the transitions was similar to those observed at lower update rates. The level of spread observed in the 40 Hz update rate transitions is similar to those observed when measuring the proportion of forward movements made by the robot.

The data from this simplified simulation shows that when selecting an object detection system for a swarm of robots, the detection accuracy and inference time are important. For very poor ($P_s \approxeq 0$) and excellent ($P_s \approxeq 1.0$) detection system, the inference time will have little or no impact on the swarm's ability to use communicated information for chemotaxis. However, for detection systems with performance somewhere between the extreme cases, the inference time and detection persistence (P_{s2s}) become more relevant as $P_s \rightarrow 0.5$.

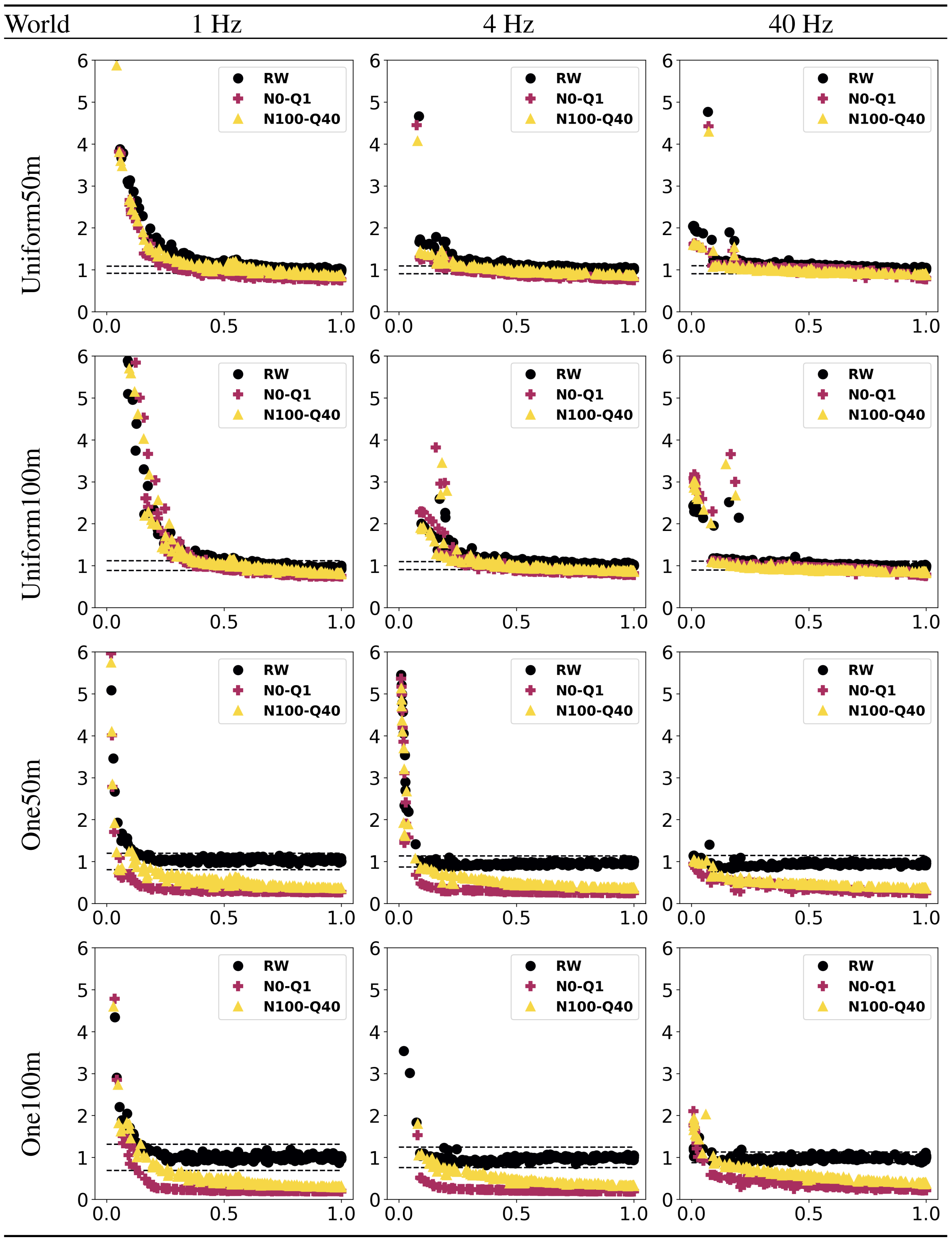


Fig. 5.17 Mean normalised time to pick up 90% of 200 targets by a swarm of 36 robots (y-axis) and P_s on the x-axis. Foraging times were normalised using the time taken by Random Walk with $P_{u2s} = P_{s2s} = 1.0$ for the corresponding world and vision update rate setups.

5.8.3 Effects of P_{u2s}, P_{s2s} and vision update rate on swarm foraging

The combinations of the P_{s2s} and P_{u2s} model probabilities were also used to examine the foraging performance for a swarm of 36 robots foraging 200 targets in One50m, One100m, Uniform50m and Uniform100m environments. The vision update rate of the robots in each simulation was either 1 Hz, 4 Hz or 40 Hz in order to study the effects of object detection update rates on the swarm. In all simulations, the chemotaxis parameters of $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$ were used. Simulations for each model probabilities combination was repeated 30 times and the time it took for the swarm to pick up 180 targets was averaged across the repeated simulation runs.

The foraging times for vision update rates of 1 Hz, 4 Hz and 40 Hz are shown in Fig. 5.17 for Random Walk, N0-Q1 and N100-Q40 algorithms. The foraging times were normalised using the performance of Random Walk with perfect vision for the corresponding worlds. For Random Walk, the impact of imperfect vision was more obvious in the uniform worlds and vision update rate of 1 Hz. However, at higher vision update rates (and in the single cluster worlds), imperfect vision had no noticeable effect until $P_s < 0.2$, which was the point at which search time increases due to poor targets detection, resulting in a longer total foraging time.

For the RepAtt algorithm, imperfect vision affects both the robots' ability to sense the presence of targets and choice of signal to communicate. A very low P_s means robots will broadcast repulsion signals when they ought to be attracting, which in turn negatively impacts the recruitment of other swarm members to clusters of litter objects. This is why the impact of imperfect vision on swarm foraging is noticeable for slight reductions in P_s values, with stronger effects observed in the one cluster environments. RepAtt outperformed Random Walk for $P_s \geq 0.1$ in all the test environments and vision update rates, which indicates that the swarm was still able to exhibit some level of cooperation to help improve their foraging efficiency.

The impact of vision update rates on Random Walk, N0-Q1 and N100-Q40 uniform and one cluster environments is shown in Fig. 5.18 (this is same data as in Fig. 5.17, but grouped differently). For Random Walk, the general trend was that in all the test environments, increasing the vision update rates reduced foraging time for specific P_s values. This behaviour was expected because the robots had more attempts at detecting the targets as vision update rate increased. In the single cluster environments, increasing the vision update rate negatively impacted N0-Q1 foraging performance. The reason for this interesting behaviour was because the ability of searching robots to respond to attraction signals was significantly impacted negatively by the alternating attraction-repulsion behaviours of the robots that located targets cluster (as discussed in Section 5.8.2). For N100-Q40, the effect

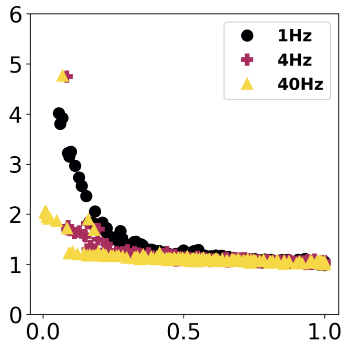
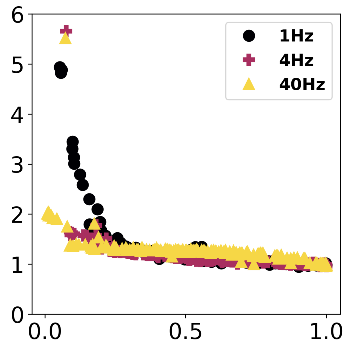
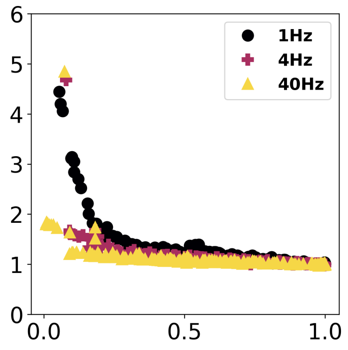
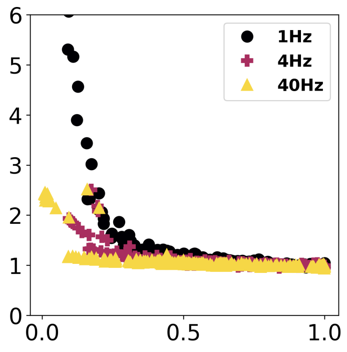
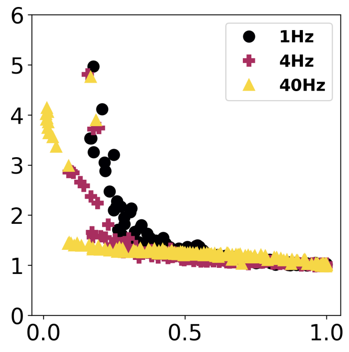
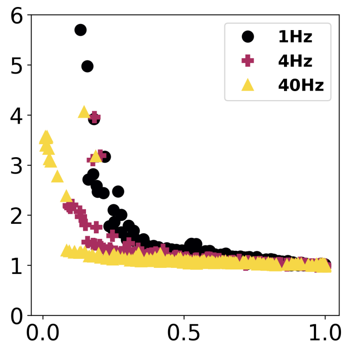
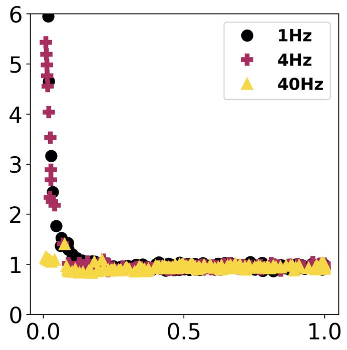
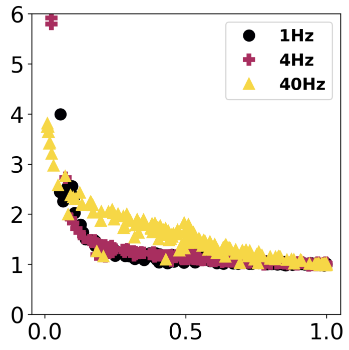
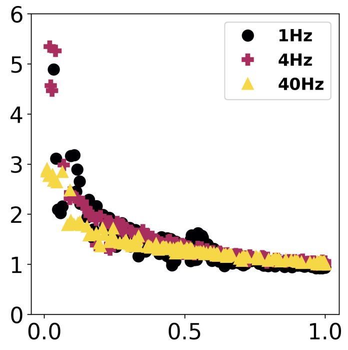
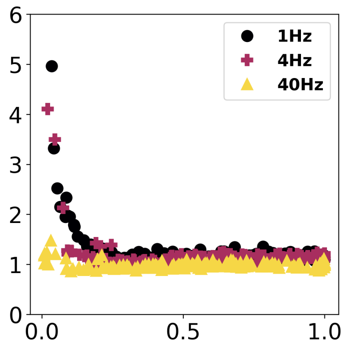
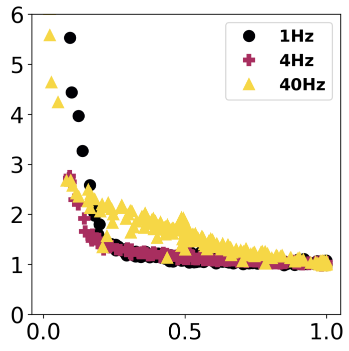
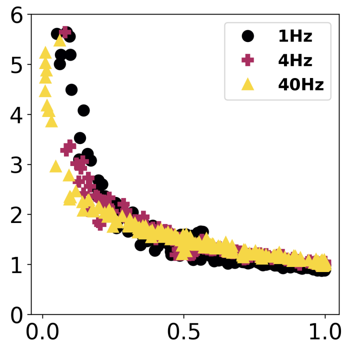
World	RW	N0-Q1	N100-Q40
Uniform50m			
Uniform100m			
One50m			
One100m			

Fig. 5.18 Mean normalised time to pick up 90% of 200 targets by a swarm of 36 robots (y-axis) and P_s on the x-axis. Foraging times were normalised using the time taken by the 40 Hz vision update rates version of the algorithms with perfect vision ($P_{u2s} = P_{s2s} = 1.0$) for the corresponding world and vision update rate setups.

of increasing the vision update rate was similar to those observed for the Random Walk algorithm. This 'normalised' effect was mainly due to the average filter used by the searching robots to reduce the influence of communication noise. By computing the average of multiple intensity measurements, the robots reduced both attraction-to-repulsion and repulsion-to-attraction transitions, thus, helping them to use single gradient computation for longer periods (in comparison to per time step update of N0-Q1). As a result, reducing the fluctuations in transitions between attraction and repulsion, whether at the source of the signal or by the listening robot, generally had an overall effect of improving the swarm recruitment using RepAtt to targets cluster, consequently reducing swarm foraging time.

5.9 Summary

This chapter has discussed the steps taken to implement a probabilistic vision model using state-of-the-art machine learning-based object detection networks. The vision model is based on two transition probabilities (P_{u2s} and P_{s2s}), which control a robot's ability to detect a target to forage. Using real-world data, this model was validated against the observations in computer vision experiments. Through extensive simulation experiments, it was found that robots that foraged using RepAtt algorithms need a mechanism that minimises fluctuations between attraction and repulsion signals in order to overcome the effect of the imperfect vision system on swarm foraging. This can be achieved by reducing the vision update rate of the broadcasting robot or using an average filter by the listening robot. Therefore, when designing a vision system for a swarm of robots, both the detection accuracy and vision update rate need to be carefully accounted for in order to minimise the impact of imperfect vision on the foraging performance of the swarm.

Chapter 6: Chemotaxis-Based Virtual Fence

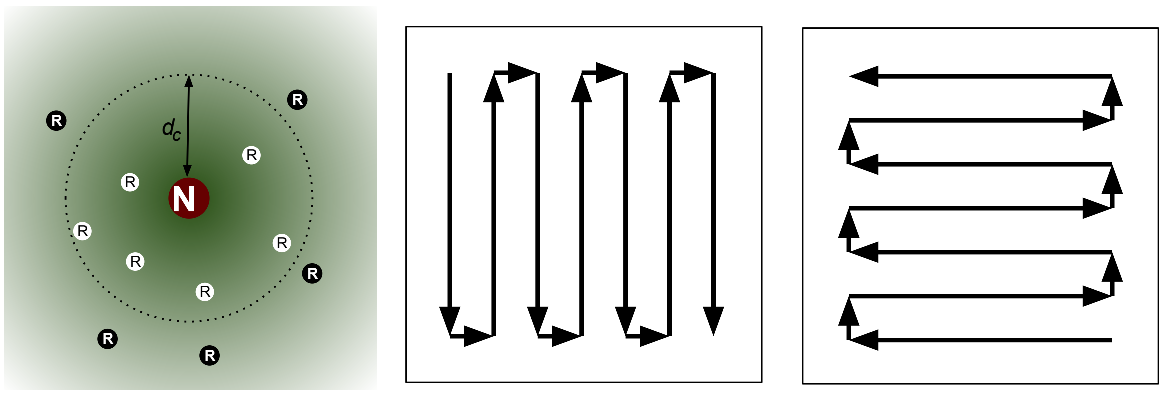
6.1 Introduction

In practice, swarm robots can be deployed to areas where there is no infrastructure to keep them within a desired region of interest. For example, robots deployed in outdoor search and rescue operations, planetary exploration or picking litter in an unfenced park. Much work in swarm robotics have developed algorithms that focused on task execution with an underlying assumption that there is a physical structure or boundary that keeps the swarm within a working area. Examples include works presented in [63] and [73], where swarm robots foraged within bounded simulation environments. This approach is unrealistic given that a fencing structure will usually not be feasible in many applications and it does not take into consideration that the swarm working area can change over time. In addition, swarm robots typically are incapable of performing simultaneous localisation and mapping of the search space because the complexity and memory requirements would be excessive for a single robot in the swarm to handle. This problem has been resolved by proposing a chemotaxis-based virtual fencing mechanism for a swarm of robots deployed in an unbounded search area. The algorithm is simple, scalable and can be implemented on hardware robot platforms. It represents a different, but related, application of chemotactic search to a swarm robotics.

The chemotaxis-based algorithm is introduced in section 6.2, followed by the description of the simulation setup in section 6.3. In Section 6.4, experiments to develop a model equation for a number of robots within the work area are presented, while the effects of chemotaxis parameters and nest velocity on the chemotactic fence are explored in section 6.5. Section 6.6 investigates the robots' ability to effectively explore a work area within unbounded environments for both stationary and moving nests. Experiments using real robots are described in section 6.7, and section 6.8 summarises the findings of this chapter.

6.2 Exploration in Unbounded Environments

The design involves a nest that broadcasts a homing signal that degrades with distance and a swarm of mobile robots that actively sense this signal's intensity while executing their individual tasks as shown in Fig. 6.1a. These mobile robots perform pure random walk (with constant tumble probability, P_t) or chemotactic-search depending on which region around the nest they are situated (a robot's region is informed by the intensity of the homing signal it senses). The region surrounding the nest (also known as guide robot when it is mobile) is



(a) The swarm (b) Vertical pattern (c) Horizontal pattern

Fig. 6.1 In (a), the nest (N) broadcasts a homing signal that degrades with distance (green gradient), which mobile robots (R) listen to. Robots within the working area ($< d_c$) perform random walk with constant tumble probability P_t, while robots within the chemotaxis region ($\geq d_c$) use chemotactic search to return to the working area. White circles represent robots in working area, while black circles represent robots in chemotaxis region. During nest exploration, the nest moves in vertical (b) and horizontal (c) pattern in the search area.

divided into a work area ($< d_c$) and a chemotactic region ($\geq d_c$), where d_c is the distance that corresponds to chemotaxis activation threshold, $A(d_c)$. Within the work area, the robot executes the swarm task (abstracted in the current work as random exploration), however, when a robot enters the chemotactic region, it uses the *C. elegans* inspired chemotactic behaviour to return to the work area. Thus, the chemotactic region serves as a fencing mechanism to prevent swarm robots from continuously drifting away from the work area around the nest. Algorithm 3 shows the steps executed by individual mobile robot within each time step.

The robot first senses the intensity of the nest signal, A_t, from its current location and sets its tumble probability, P_t, to a pre-determined base probability, P_b. If A_t is below the chemotaxis activation threshold, ($A(d_c)$), the robot updates its P_t based on whether there is an increase or decrease in the signal's intensity between the current intensity (A_t) and the last measurement ($A_{(t-1)}$). The attraction multiplier, a_m, and divisor, a_d, chemotaxis parameters are the only parameters the robot needs in this algorithm since the nest broadcasts only an attractive homing signal. The robot uses P_t to then determine whether to tumble (rand(0,1) $< P_t$) or swim (rand(0,1) $\geq P_t$) at a constant velocity v_r. The rand(0,1) term is a function that randomly generates numbers from a uniform distribution having minimum and maximum values of 0 and 1 respectively. During a tumble, the robot randomly selects the size of angle to rotate from a normal distribution with mean $\mu = 180°$ and standard deviation $\sigma = 90°$.

Algorithm 3 Random Walk with chemotaxis activation

```
1:  Sense nest signal, $A_t$
2:  Initialize $P_t = P_b$ to default value
3:  **if** $A_t < A(d_c)$ **then**
4:      **if** $A_t < A_{t-1}$ **then**
5:          $P_t = P_b \times a_m$
6:      **else if** $A_t > A_{t-1}$ **then**
7:          $P_t = P_b \div a_d$
8:  **if** rand(0,1) $< P_t$ **then**
9:      make random turn of $N(\mu, \sigma^2)$
10: **else**
11:     make straight motion
12: $A_{t-1} = A_t$
```

In the basic simulation setup, the nest is stationary within the environment. However, to extend the work area to cover a wider search space beyond those around a stationary nest. Without a physical boundary to restrict the swarm, the nest itself can move within the unbounded environment. While doing this, the nest continuously updates the work area as it broadcasts its homing signal, thereby guiding the exploration of the mobile robots during the process. In the work presented here, the nest's basic movement is linear from a starting location to a destination point, where it moves at a constant velocity, v_n, that is relative to that of the exploring swarm robots. The nest waits (stops briefly) whenever it senses robots within d_n metres of its front region in order to avoid collisions with exploring robots. Given a two-dimensional search area, the nest robot repeatedly performs a sweep of the environment using the search pattern illustrated in Fig. 6.1b and 6.1c until a stopping criteria is met. Maximum simulation time, t_{max}, is used as the stopping criterion for the swarm to halt the sweep.

The movement of the nest extends the swarm's capability to explore a wide search area in an unbounded environment. However, it also raises concerns regarding optimal nest velocity that will balance the swarm's exploration speed, accuracy (or efficiency or thoroughness of the exploration) and minimise the number of robots that go adrift of the search area due to their inability to reliably sense the gradient of the nest signal. These questions are investigated in subsequent sections.

6.3 Simulation Setup

Gazebo simulation platform was also used to analyse the swarm performance. A swarm size of 10 mobile robots moving at $v_r = 0.605$ m/s was used. Simulation time step of 25 ms and base tumble probability was $P_b = 0.0025$ per time step were used in all simulations. A single nest with velocity, v_n is expressed as being relative to v_r in all experiments. The nest stops whenever a robot is within $d_n = 0.1$ m in its front region. The nest signal used in all simulations was the noisy version of the negative exponential model of the sound signal (Equation 4.3 in Chapter 4), where the simulation uses the distance of robots from the nest to compute the signal intensity each robot senses. An average filter with queue size of 40 was used in all simulations.

6.4 Determining the Chemotactic Region

The first set of simulations use a stationary nest and 10 exploration robots that start out from the nest and perform random walk to explore the nest region. The simulation's goal was to determine the chemotaxis activation intensity, $A(d_c)$, needed to keep swarm robots within a specified radius (d_r) around the the nest. In these simulations, the chemotaxis activation distance, d_c, was varied from 6 to 14 metres from the nest. The maximum simulation time $t_{max} = 1500$ seconds for each d_c was used. Each simulation setup was repeated 30 times and the average number of robots within varied distances from the nest (d_r) was computed as shown in Table 6.1.

Table 6.1 Chemotaxis activation distance d_c versus robots distances from nest d_r. Each value represents the mean number of robots and 95% confidence interval for 30 repetitions of each simulation. Maximum simulation time was $t_{max} = 1500$ seconds, swarm size was 10, $a_m = 10$ and $a_d = 1000$.

d_c / d_r	6 m	8 m	10 m	12 m	14 m
6 m	4.4±0.03	-	-	-	-
8 m	7.9±0.02	5.3±0.04	-	-	-
10 m	9.4±0.01	8.3±0.02	6.0±0.04	-	-
12 m	9.8±0.01	9.5±0.01	8.6±0.02	6.5±0.04	-
14 m	10.0±0.00	9.8±0.01	9.6±0.01	8.8±0.02	6.7±0.04
16 m	10.0±0.00	10.0±0.00	9.9±0.01	9.7±0.01	8.7±0.03
18 m	10.0±0.00	10.0±0.00	10.0±0.00	9.9±0.00	9.6±0.01
20 m	10.0±0.00	10.0±0.00	10.0±0.00	10.0±0.00	9.9±0.01
22 m	10.0±0.00	10.0±0.00	10.0±0.00	10.0±0.00	10.0±0.00

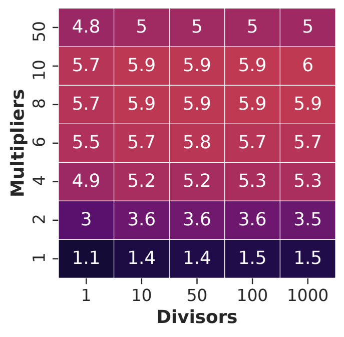
The results show that as d_c increases, the number of robots within the work area also increases, starting from about 44% of swarm size when $d_c = 6$ m to 67% of exploring robots when $d_c = 14$ m. This has more to do with reduction in inter-robot interference as d_c increases. When d_c is small, the work area is small, and robots collide more frequently with each other and end up frequenting the chemotaxis region more often. However, when d_c is large the mobile robots have a wider area to explore, which reduces interference from other robots. The results also indicate that the chemotactic region is effective in keeping approximately 95% of the exploring robots within $d_c + 4$ metres radius around the nest for all the d_c tested in Table 6.1. Thus, for design purposes, when the goal is to keep at least 95% of swarm robots within d_w meters around a stationary nest, the chemotaxis activation distance can be computed using Equation 6.1.

$$d_c = d_w - 4 \tag{6.1}$$

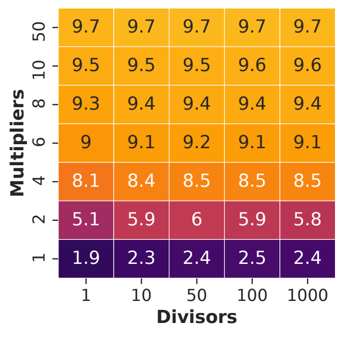
6.5 Effects of Probability Multiplier and Divisor

As previously discussed in Chapters 3 and 4, the chemotaxis parameters (a_m and a_d in this case) play an important role in the robots' ability to make use of the signal intensity they sense. For a stationary nest, only multiplier a_m, divisor a_d and chemotaxis activation distance impact the virtual fence's effectiveness. However, for a mobile nest, the nest's velocity, v_n, relative to the exploring robots is also important, because if the nest is too fast, the exploring robots could get left behind and trapped within the chemotactic region. To investigate the effect of these parameters, the chemotaxis activation threshold distance was set to $d_c = 10$ m, nest's relative velocities of $v_n = 0, 0.125v_r$ or $0.25v_r$, tumble probability multiplier (a_m) varied from 1 - 50 and divisors a_d varying from 1 to 1000 were used. The heat maps of Fig. 6.2 show the result of these simulations, where each cell in the heat map represents the mean number of robots within d_r metres of the nest, averaged across 30 repetitions for each parameter setting. When $v_n > 0$ in Fig. 6.2, the nest was programmed to follow a linear path for 100 metres.

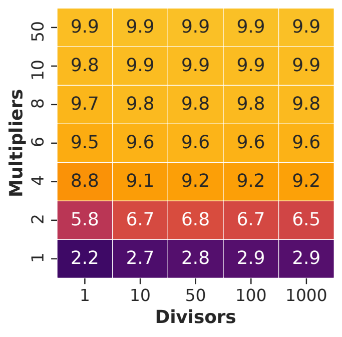
As expected, the data in Fig. 6.2 show that the tumble probability multiplier a_m and divisor a_d can have significant impact on the number of robots that are able to remain within the working area around the nest robot. A small value of a_m reduced the robots' responsiveness to decreasing intensity of the nest's attraction signal when they are in the chemotactic region. This makes the virtual fence elastic since exploring robots are able to move further into the chemotactic region before making a tumble that helps them return to the work area. On the other hand, when a_m is large, exploring robots become more



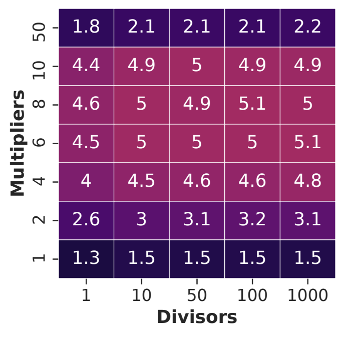
(a) $v_n = 0$, $d_r = 10$ m



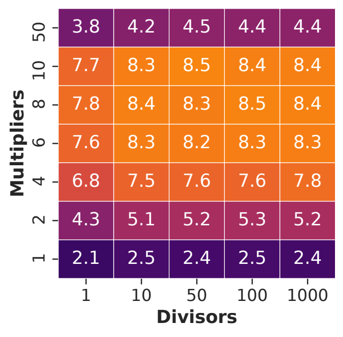
(b) $v_n = 0$, $d_r = 14$ m



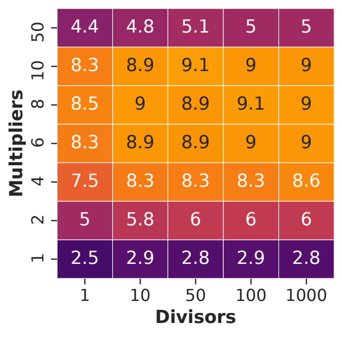
(c) $v_n = 0$, $d_r = 16$ m



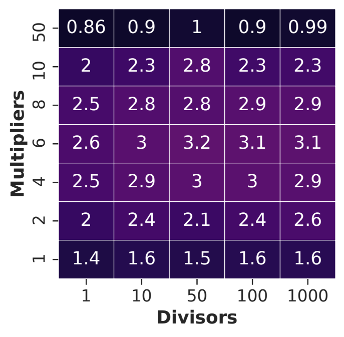
(d) $v_n = 0.125v_r$, $d_r = 10$ m



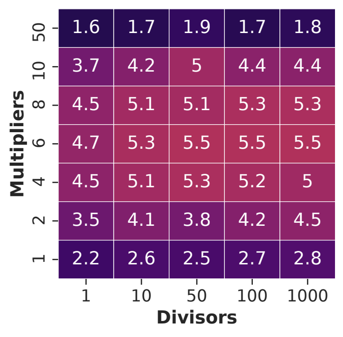
(e) $v_n = 0.125v_r$, $d_r = 14$ m



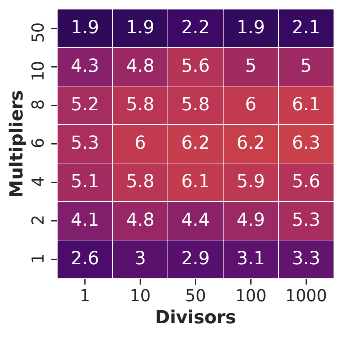
(f) $v_n = 0.125v_r$, $d_r = 16$ m



(g) $v_n = 0.25v_r$, $d_r = 10$ m



(h) $v_n = 0.25v_r$, $d_r = 14$ m



(i) $v_n = 0.25v_r$, $d_r = 16$ m

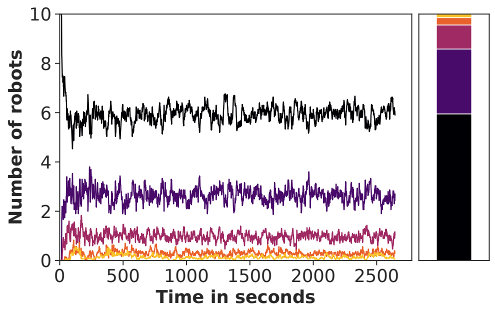
Fig. 6.2 Effects of nest relative velocity, v_n, probability multiplier, a_m, and divisor, a_d, on the average number of robots within a specific distance, d_r from the nest/guide robot. Chemotaxis activation distance was 10m for all simulations. The background colours range from dark to light, indicating low to high values respectively.

responsive to decreasing nest signal intensity once in the chemotactic region. This means that within a short distance of entering the chemotactic region, exploring robots make tumble(s) that change their direction in order to help them return to the working area. However, an excessively large a_m, caused the robots to tumble too frequently in the chemotactic region, thus preventing them from making sufficient linear motion to compute a reliable gradient for the noisy nest signal. This is the reason for the sudden decrease in the number of robots within the work area when $a_m = 50$ in comparison to when $a_m = 10$ (that is, number of robots within 10 metres of the nest in Fig. 6.2a dropped from approximately 6 to 5). Note that if the nest's signal is perfect (noiseless), a large a_m poses no problem since the robot will always compute a reliable gradient.

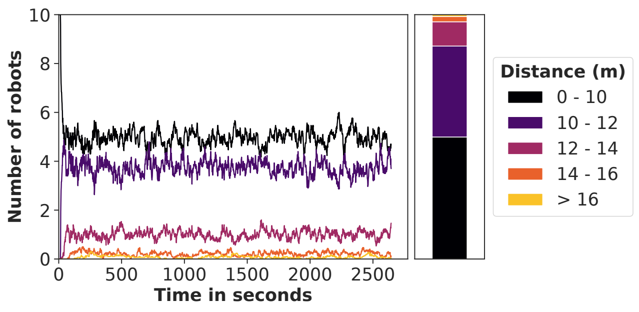
The tumble probability divisor a_d, has comparatively less impact on the number of exploring robots in the working area. The results in Fig. 6.2 show that the number of robots within the work area generally increase with a_d. A low a_d will reduce a robot's ability to swim longer distances when the robot senses positive gradient while in the chemotactic region. Thus, a large a_d is generally good as it will guarantee that the robot makes longer swims when moving toward the working area. These descriptions of the impact of a_m and a_d on the effectiveness of the chemotactic fence for stationary nest suggests that parameter setting $a_m = 10$ and $a_d = 1000$ will give best results. This is indeed the case when nest's relative $v_n = 0$ and to a large extent $v_n = 0.125v_r$. However, when values for $v_n = 0.25v_r$, $a_m = 6$ and $a_d = 1000$ produced the best performance in retaining robots within the working area.

It is intuitive to expect more robots to find it difficult to 'catch up' with the nest when the nest's velocity increases (as shown in Fig. 6.2). In terms of the impact of a_m and a_d, as the nest relative velocity increases, the value of a_m typically offering good performance reduces (from 10 when $v_n = 0$ to 6 when $v_n = 0.25v_r$), while $a_d = 1000$ remained a good choice. During the simulation, the number of robots around the nest varies over time. For a stationary nest, a good combination of a_m and a_d will guarantee that no robot in the swarm gets trapped in a chemotactic region where it would be unable to make its way back to the work area near the nest. However, when the nest is mobile, the velocity of the nest v_n can play a vital role in determining how quickly robots get trapped within a chemotactic region where they are unable to use chemotaxis to return to the nest's work area. Such a scenario is illustrated in Fig. 6.3.

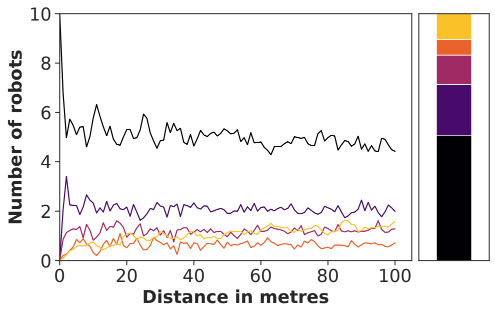
In Figs. 6.3a and 6.3b, $v_n = 0$, and throughout the 2,500 seconds long simulation, approximately all the 10 exploring robots remained within 16 metres of the nest's position. However, when $v_n > 0$ in Figs. 6.3c - 6.3f, the swarm's ability to remain near the nest changes dramatically depending on the choice of a_m and a_d. By using a lower attraction multiplier $a_m = 6$ (Fig. 6.3c and 6.3e), the swarm was able to minimise the rate at which robots got left behind as the nest makes its 100 metres journey. The negative impact of a large attraction multiplier value $a_m = 50$ became more pronounced as v_n increases. As mentioned earlier, this is because as robot's distance from the nest increases, its computation of intensity gradient of the nest attraction signal becomes less reliable due to noise in the signal. The large a_m, thus, caused the robot to tumble almost immediately when it sensed a negative gradient (which



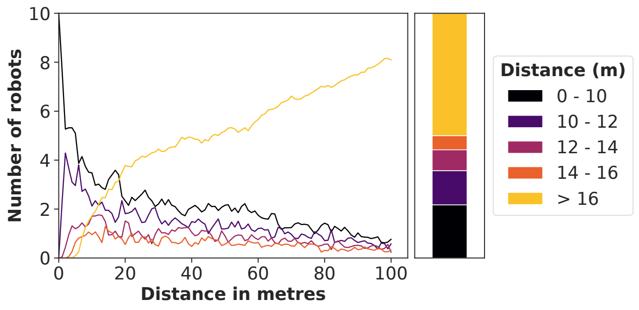
(a) $v_n = 0$, $a_m = 10$, $a_d = 1000$



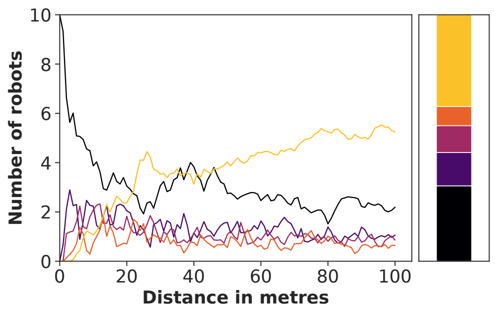
(b) $v_n = 0$, $a_m = 50$, $a_d = 1000$



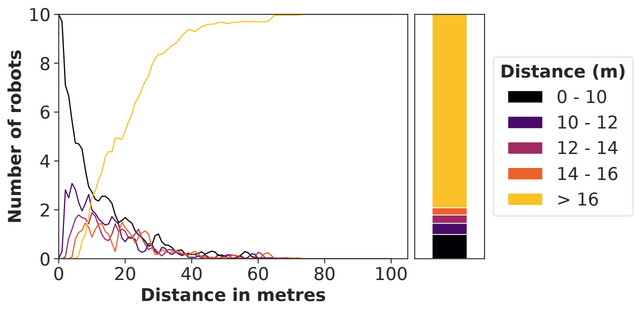
(c) $v_n = 0.125v_r$, $a_m = 6$, $a_d = 1000$



(d) $v_n = 0.125v_r$, $a_m = 50$, $a_d = 1000$



(e) $v_n = 0.25v_r$, $a_m = 6$, $a_d = 1000$



(f) $v_n = 0.25v_r$, $a_m = 50$, $a_d = 1000$

Fig. 6.3 Effects of nest relative velocity, v_n, tumble probability multiplier, a_m, and divisor, a_d, on the average number of robots within a specific distance, d_r ranges from the nest/guide robot.

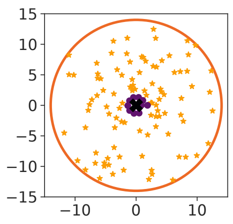
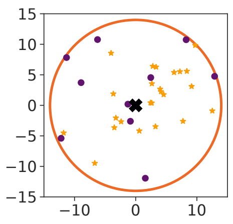
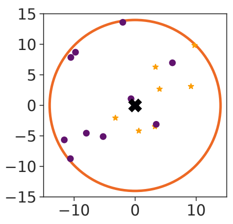
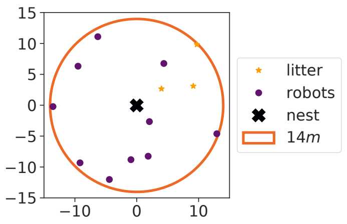
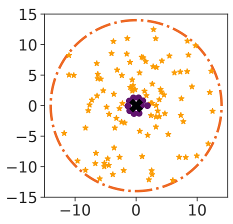
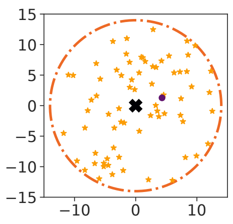
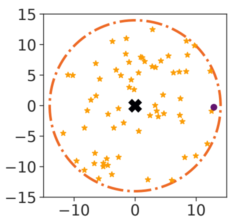
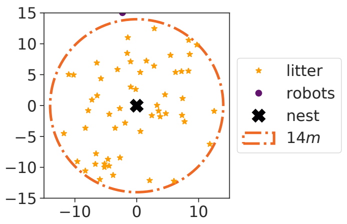
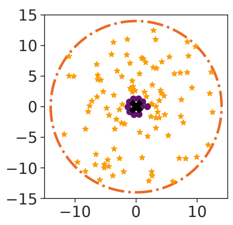
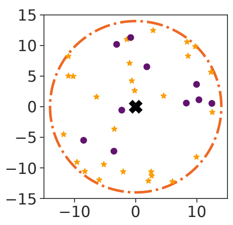
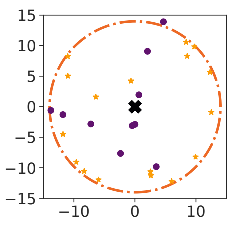
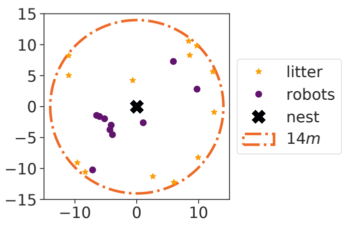
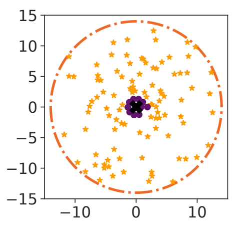
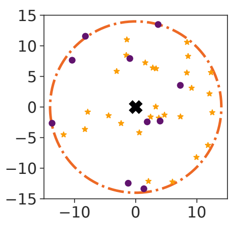
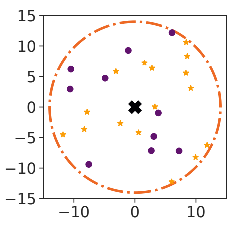
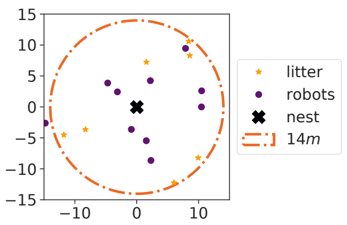
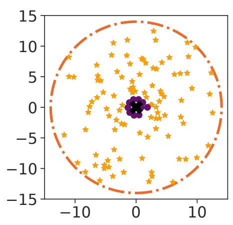
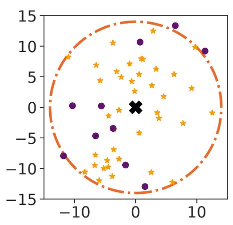
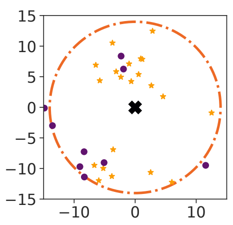
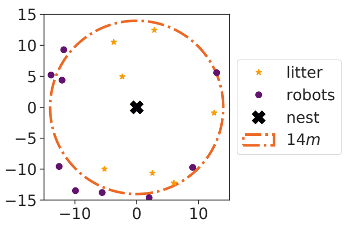
in reality could be a positive gradient if the robot had delayed the tumble). As a result, the lower $a_m = 6$ was more successful in helping exploring robots keep up with the moving nest because a low value of a_m meant robots were able to delay tumbles for much longer time (and distance), which increased their chances of computing more reliable gradient values when in the chemotactic region.

6.6 Investigating Exploration Effectiveness

The chemotactic fence is important for keeping exploration robots from continuously drifting away from the nest region, as has been previously elaborated on. However, being close to the nest region is not a sufficient or decisive factor because robots also need to be able to fully explore the search space around the nest. This means robots need to minimise the time they spend in the chemotactic region in order to effectively utilise their time within the working area. This section tests the swarm's ability to explore the work area by giving the exploration robots a target search and pick up task for both stationary and moving nest. The task in each simulation is for the swarm of 10 robots to locate and pick up 100 targets that are uniformly distributed within the search space. Robots in this simulation had unlimited capacity and were only able to detect targets that were directly beneath them. Two environmental setups were used as baseline tests: *Bounded* where robots used random walk to search for targets within a physically bounded area; *Unbounded* where the physical boundary was removed and robots searched using random walk without chemotaxis activation. In all simulations with chemotaxis activation, the physical boundary was removed and robots relied on the chemotactic behaviour to remain near the nest. Each simulation was repeated 30 times and averaged.

In the first set of simulations, the nest was stationary and the 100 targets were uniformly distributed within a 14 m radius search area around the nest. Chemotaxis activation distance, d_c, of 10, 12 and 14 metres were used to test the exploration performance of the robots using $a_m = 10$ and $a_d = 1000$ to perform chemotactic search of the work area once they found themselves within the chemotactic region. Table 6.2 shows simulation snapshots for all the five test cases (the maximum simulation time $t_{max} = 1200$ seconds). With a physical boundary (*Bounded* row) of 14 m radius, the exploring robots always remain within search space and 'bounce' off the walls whenever they encounter it. However, when the boundary was removed (*Unbounded* row), the robots quickly drift from the 14 m region where the targets were located. This is expected because there was no mechanism with which to restrict the robots within the 14 m region, and the robots only had local knowledge of their environment and no localisation capability. Thus, they continuously searched for targets using random walk within an unlimited search space and over time drifted away from where they could easily locate the targets. When $d_c = 10$ m (in third row of Table 6.2), the robots were able to remain within the 14 m search area. They used chemotaxis to ensure that they did not continue to stray from the nest area, thus restricting their search to where they could locate targets. Similarly, when $d_c = 12$ and 14 metres, the exploring robots also remained within the nest area.

Table 6.2 Scatter plot of a sample simulation showing a swarm of 10 robots searching for 100 targets uniformly distributed within a 14 m radius search area.

d_c \ t	1 s	400 s	800 s	1200 s
Bounded				
Unbounded				
10 m				
12 m				
14 m				

Measurements of the swarm's exploration performance after 30 repetitions of the simulations are shown in Table 6.3. The results indicate that the when a boundary is used to restrict swarm movement so that they always remained within the desired area, the robots were able to locate and pick up 95.2% of the targets after 1000 seconds. In the absence of the wall restriction and no chemotaxis, the swarm's exploratory ability dropped, and robots were only able to pick up 50.5% of the targets. The results also imply that the presence

Table 6.3 Comparison of number of targets found within a circular search area of radius 14 m and d_c of 10, 12 and 14 metres in unbounded world. Chemotaxis parameters were $a_m = 10$ and $a_d = 1000$. Each value represents the mean of 30 independent simulations and 95% confidence interval.

t \ d_c	Bounded	Unbounded	10 m	12 m	14 m
200 s	49.4 ± 2.32	34.0 ± 1.72	49.2 ± 1.32	46.7 ± 1.43	45.3 ± 1.12
400 s	71.2 ± 2.27	39.0 ± 2.18	69.1 ± 1.66	68.3 ± 1.72	64.7 ± 1.50
600 s	84.5 ± 1.81	43.9 ± 3.11	79.3 ± 1.33	81.1 ± 1.40	77.0 ± 1.55
800 s	91.2 ± 1.25	47.0 ± 3.43	85.5 ± 1.07	88.4 ± 1.27	85.7 ± 1.47
1000 s	95.2 ± 0.97	50.5 ± 3.79	88.7 ± 0.98	93.5 ± 0.78	90.7 ± 1.17

of the chemotactic fence significantly improved the swarm's performance in the absence of a wall, causing them to pick up 93.5% when $d_c = 12$ m. In the presence of the virtual chemotactic fence, $d_c = 12$ m offered best balance between searching for targets within the chemotactic region (12 to 14 metres from the nest) and the work area (which is within a 12-metre radius around the nest), making it almost as good as the swarm's performance when there was a physical boundary (*Bounded* world setup). The performance drop when $d_c = 10$ m was because it only allowed the swarm to effectively pick up nearby targets (within 10 m of the nest), but made the robots' exploration less efficient for picking up targets that were further into the chemotactic region (between 10 m and 14 m from the nest). However, when $d_c = 14$ m, robots had a wider search area where they only performed random walk in addition to a chemotactic region that was more than 14 m from the nest. Another factor that negatively impacted exploration when $d_c = 14$ m is the reliability of the nest signal intensity the robots sensed becomes less accurate as they moved further away from the nest (the rate of change for the negative exponential equation decreases with distance), which is further compounded by the noise in the communicated signal.

In the second set of exploration simulations, the task was for the swarm to forage 100 targets within a 100 m $\times$ 100 m search space. To reliably explore this wide area, the nest was modelled to follow a predefined pattern within the environment (nest movement pattern is as shown in Fig. 6.1), and as such, making the working area of the swarm dynamic. As stated earlier, this is one of the advantages of the virtual chemotactic fence. In the simulation setup, the nest robot continuously makes vertical and horizontal sweeps within the 100 m by 100 m search space for $t_{max} = 10,000$ seconds. The simulations were conducted where nest's velocity, v_n was between $0.1v_r$ and $0.25v_r$, $d_c = 12$ m, $a_m = 6$ and $a_d = 1000$. Just as in the stationary nest simulations, the baseline conditions used here were *Bounded* and *Unbounded* versions of the world where robots performed pure random walk without making

Table 6.4 Scatter plot of a sample simulation showing swarm searching for 100 targets uniformly distributed within a 100 m × 100 m search area.

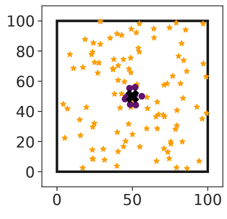
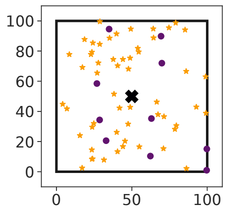
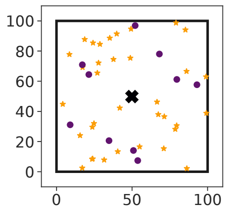
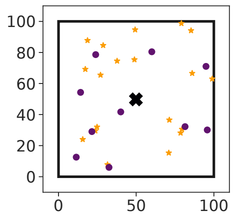
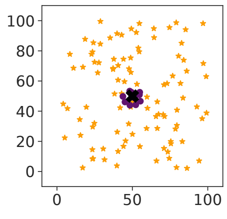
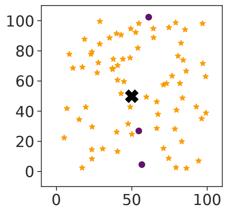
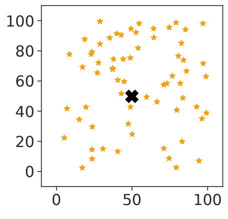
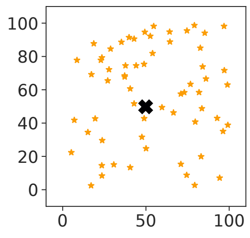
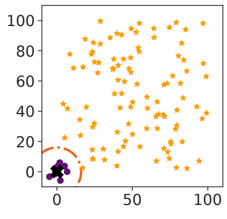
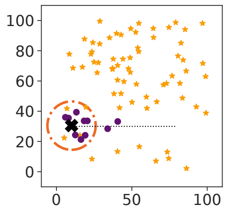
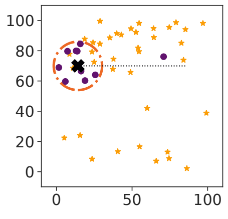
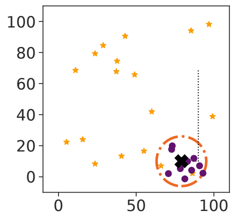
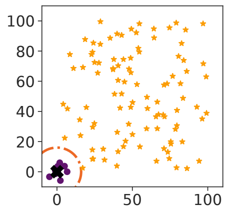
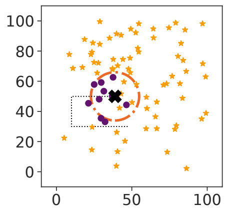
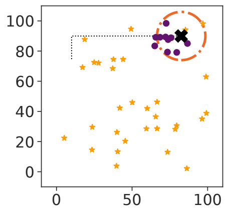
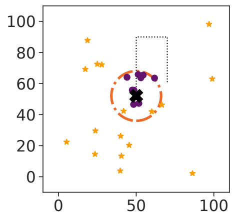
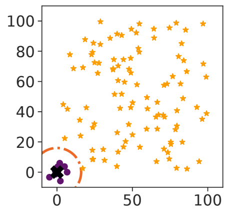
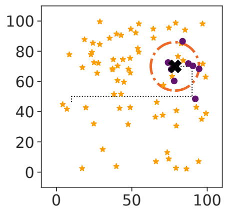
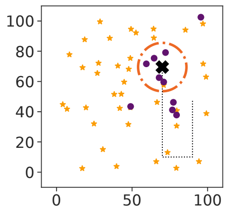
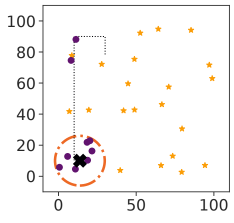
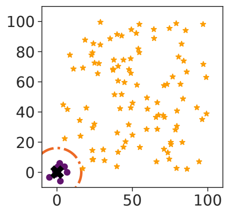
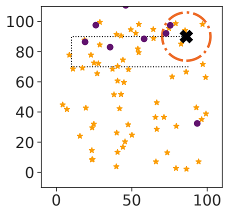
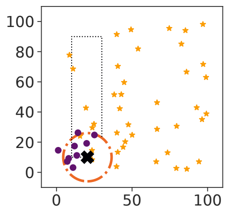
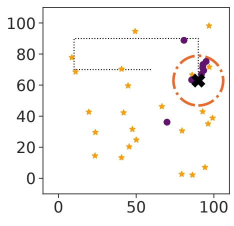
v_n \ t	10 s	3,500 s	7,000 s	10,000 s	
Bounded					nest path ★ targets ● robots ✖ nest ▭ 100m
Unbounded					★ litter ● robots ✖ nest
$v_n = 0.1v_r$					nest path ★ targets ● robots ✖ nest 16m
$v_n = 0.125v_r$					nest path ★ targets ● robots ✖ nest 16m
$v_n = 0.167v_r$					nest path ★ targets ● robots ✖ nest 16m
$v_n = 0.25v_r$					nest path ★ targets ● robots ✖ nest 16m

Table 6.5 Targets found as exploration time, t, progresses. $d_c = 12$ m, $a_m = 6$ and $a_d = 1000$ for chemotaxis based approach. Each value represents the mean of 30 independent simulations and 95% confidence interval.

t \ v_n	Bounded	Unbounded	$0.1v_r$	$0.125v_r$	$0.167v_r$	$0.25v_r$
2,000 s	35.1 ± 1.21	21.3 ± 1.57	17.9 ± 0.84	19.3 ± 1.29	22.9 ± 1.30	21.9 ± 1.45
4,000 s	56.0 ± 1.65	27.0 ± 2.02	34.8 ± 0.87	38.9 ± 1.45	45.2 ± 1.79	41.3 ± 2.06
6,000 s	69.0 ± 1.69	31.4 ± 2.28	51.7 ± 1.11	57.9 ± 1.60	61.9 ± 2.03	55.2 ± 2.18
8,000 s	78.6 ± 1.37	34.3 ± 2.45	72.5 ± 1.38	71.3 ± 1.75	72.4 ± 1.87	62.8 ± 2.62
10,000 s	86.1 ± 1.30	36.9 ± 2.53	80.8 ± 1.09	78.4 ± 1.61	79.1 ± 1.53	70.9 ± 2.68

use of any chemotactic signal. Each simulation setup was also repeated 30 times in order to obtain reliable results of the exploration metric.

Sample simulation snapshots are shown in Table 6.4 for all the tested setups. As expected, the *Bounded* and *Unbounded* tests behaved similarly to those of the first set of simulations where the nest was stationary. The wall kept robots within the search area, while its absence meant robots drifted away from the 100 m by 100 m targets distribution region over time in the *Bounded* and *Unbounded* cases respectively. When the nest was mobile and chemotactic fence activated, the robots were more successful in remaining within the 100 m by 100 m working region. Within the 10,000 seconds simulation time, the nest was barely able to complete one sweep of the environment when $v_n = 0.1v_r$; when $v_n = 0.125v_r$, the nest's sweep was approximately 1.5; when $v_n = 0.167v_r$, the nest completed 2 sweeps; and when $v_n = 0.25v_r$, the nest was able to conveniently complete 2.5 sweeps. While faster movements can help the nest complete multiple sweeps of the environment, it is also true that the exploration robots would not be able to keep up with the nest's pace. In addition, the exploratory ability of a robot is less effective when it is in the chemotactic region because it tumbles more frequently, limiting its exploratory capability to only localised regions. Therefore, a slowly moving nest ends up being more effective in confining exploring robots to its working area. This consequently helps the robots thoroughly and more efficiently explore the area compared to a nest moving at a faster velocity.

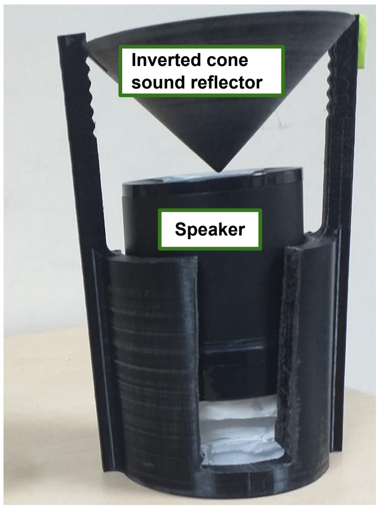
Table 6.5 shows the mean number of targets picked up by the swarm over time. With the physical wall present, the swarm was able to pick up 86% of targets, while they were only able to pick up 37% when the wall was removed. With the chemotactic fence mechanism, the exploration performance of the swarm improved significantly in unbounded environments. They were able to pick up 81% of targets when the nest's velocity was $v_n = 0.1v_r$. The relationship between v_n and the number of targets picked up can be seen by observing the exploration performance of the swarm at different velocities shown in Table 6.4 (more evident when comparing $v_n = 0.1v_r$ and $0.167v_r$). After 2,000 seconds, $v_n = 0.167v_r$ did much better than $v_n = 0.1v_r$ because the nest was able to explore more area of the search space. However, as the simulation progressed, the performance of $v_n = 0.1v_r$ caught up in exploration performance because the swarm was able to carry out a more thorough sweep of the environment than when the nest moved at $v_n = 0.167v_r$. In some applications, it is important to find the right balance between maximising the number of robots close to the nest and fast exploration of the search space. A slow moving nest more efficiently utilises robots close to the nest, resulting in thorough exploration of the work area, while a fast moving nest minimises exploration time at the expense of the number of robots that are able to remain within the nest's work area. Multiple sweeps of the search space by the fast moving nest can compensate for the poor performance of swarm exploration. However, this is not always possible in some applications. The exploration of the search space using a nest guiding swarm of robots is significant because it is a realistic solution for deploying robots in open or boundless areas, where it can be impractical to build walls around such regions or erect a mobile wall to guide the swarm's work area. It is also useful for applications where robots are constrained to be near the nest while tasked to search areas further from the nest's region.

6.7 Real Robot Experiments

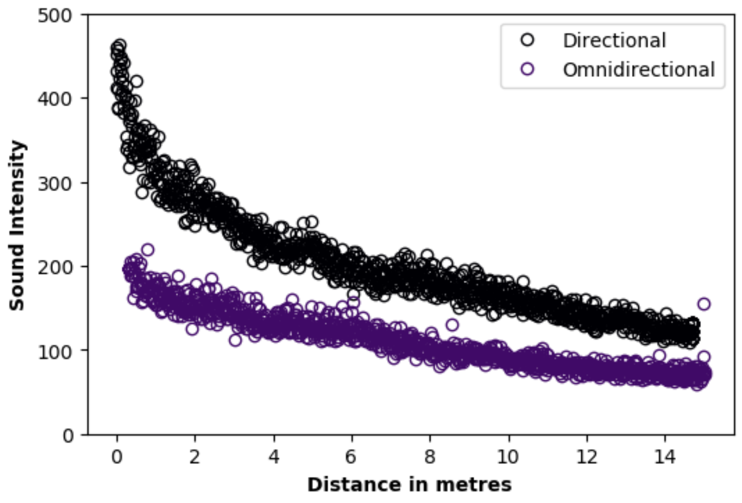
Two Turtlebot2 robots were used to validate the chemotaxis behaviour on hardware-based platforms. One robot was equipped with a microphone and programmed to execute Algorithm 3, while the other acted as the nest. Since the nest's signal was broadcast radially around the nest, the directional speaker was housed within a 3D printed platform that reflected sound signals from an upward facing speaker. The setup, shown in Fig. 6.4a, uses an inverted cone that is supported on at two ends on a 3D-printed speaker casing. Reflecting the sound from the speaker resulted in a drop in the sound intensity sensed by the microphone (and used to develop the sound model used in the simulations) as shown in Fig. 6.4b. Notwithstanding, Fig. 6.4b shows a gradual drop in sound intensity with distance for the omnidirectional speaker (at a much slower rate in comparison to directional sound data points)

The exploration robot's velocity, v_r was set to 0.1m/s, attraction multiplier $a_m = 6$, attraction divisor $a_d = 1000$, base tumble probability $P_b = 0.0025$ per time step (the size of each time step was 25 ms). The exploration and nest robots used for the experiments are shown in Fig. 6.5a. Since the signal from the omnidirectional speaker is much lower than those used in the simulation model, chemotaxis activation intensity $A(d_c)$ was set to 180 for both stationary and moving nest experiments. Each experiment setup was repeated five times.

For the stationary nest experiment ($v_n = 0$), the nest was placed at the centre of a 3 m by 6.4 m space and $t_{max} = 600$ seconds. The exploration robot was then left to move within the

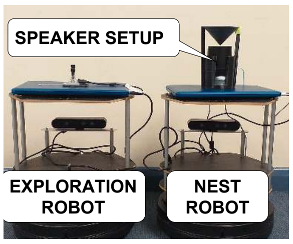


(a)

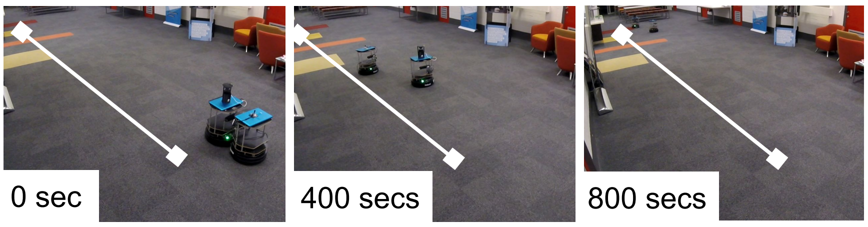


(b)

Fig. 6.4 (a) Omnidirectional speaker setup (b) Switching from directional to omnidirectional speaker resulted in general reduction in the sound intensity sensed by the mobile robot.



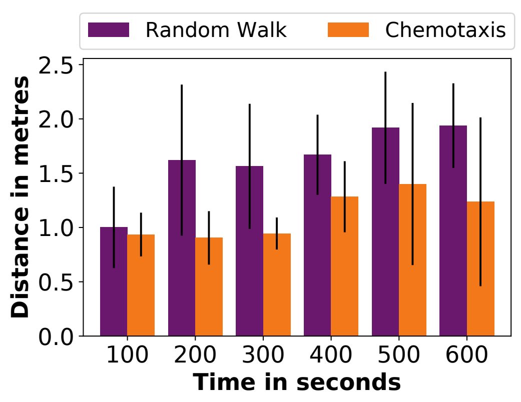
(a) Experiment robots



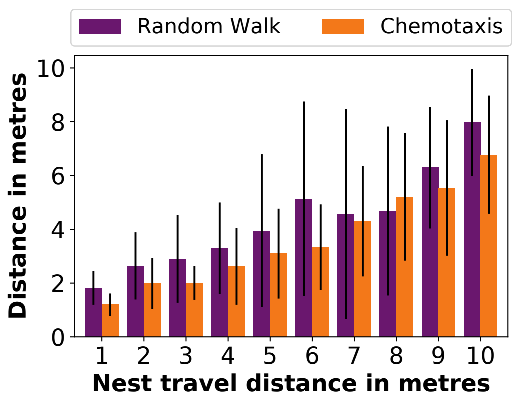
(b) white line is approximately 10 metres

Fig. 6.5 Robots used for the experiment and photos of exploration robot following the moving nest

environment using random walk while sensing the sound signal using its microphone. As a baseline, a version of the experiment was repeated where the exploration robot performed pure random walk without chemotaxis. The mean distance of the exploration robot from the nest robot in blocks of 100 seconds is shown in Fig. 6.6a. The result indicates that, by using chemotaxis, the exploration robot was reasonably successful in remaining near the stationary nest. When using chemotaxis, the mean distance of the robot from the nest was of 0.9 metre for the first 100 seconds and 1.2 metres between the 500 to 600-second period of the experiment. However when using random walk, the corresponding distances were 1 metre and 1.9 metres respectively. The result shown in Fig. 6.6a reveals that chemotaxis moderately aided the exploring robot in remaining close to the nest, and that the robot was not just simply performing a random exploration of the environment.



(a) Stationary nest



(b) Moving nest

Fig. 6.6 Real robot validation of chemotaxis behaviour to remain close to a nest. Experiments were repeated 5 times, and error bars represent 95% confidence interval.

In the second phase of hardware experiments, the robot's ability to remain close to a moving nest of relative velocity $v_n = 0.125v_r$ for 10 metres was tested. The arena for the experiment was a 3-metre wide hallway, where the nest started at one blocked edge of the arena and moved toward the open end. Fig. 6.5b shows photos of a sample experiment of the nest making the 10-metre journey while the exploring robot used chemotaxis to follow it. Fig. 6.6b shows the mean distance of the exploring robot from the nest in one metre windows for both chemotaxis-based and pure random walk exploration algorithms. In comparison to the distances recorded for random walk, within the first 6 metres, the chemotaxis-based algorithm recorded less distance from the nest robot. However, beyond 6 metres, the chemotaxis was less effective. Several factors could have contributed to this poor performance. Some of these include impact of echoes from the surrounding structures, reduced intensity of the sound signal for the omnidirectional speaker and sudden changes in ambient noise. These factors suggest that the conditions of the experiment's environmental setup need to be improved or better controlled to mitigate such negative influences. Notwithstanding, the results obtained from hardware-based experiments were promising and show that the algorithm can be deployed for real robots.

Statistical test for significant difference between the results of the chemotaxis-based and random walk algorithms were not conducted. This was because the experimental data were few (five experiment repetitions for each algorithm) and would make any T-test computation unreliable. Thus, further experiment repetitions would be required before drawing conclusions about any significant differences between the chemotaxis-based and random walk algorithms.

6.8 Summary

When swarms are deployed in real world environments, they do not necessarily have to operate within a confined (walled) search space assumed in many research publications in the area. This chapter has presented a swarm design where exploring robots use the intensity of the nest signal they sense as a virtual fencing mechanism that prevents them from drifting from the nest over time. The extensive simulation studies show that with appropriate selection of chemotaxis parameters, the virtual fence is as effective as a physical wall when the nest is stationary, while also providing competitive results when the nest is mobile. Using white noise sound as the nest's signal, experiments on hardware-based robot platforms show that the algorithm can be replicated in real world scenarios. The results from the hardware tests show that further work is required to improve the experimental setup, and research into other technologies such as Wi-Fi, RFID and ultrasound could prove useful as alternate broadcast signals for the nest. Nonetheless, the results from the hardware experiments are encouraging and represent the first steps in the deployment of chemotaxis-based virtual fence for real robots (and by extension, could be modified to demonstrate the practical implementation of the swarm foraging RepAtt algorithm).

Chapter 7: Conclusions and Future Directions

7.1 Summary

The focus of this research has been on the development of chemotaxis-based swarm foraging algorithm followed by the analysis of the impact of imperfections in communication and target detection on the collective behaviour of the swarm robots. This study has provided valuable insight into chemotaxis parameters that substantially bolstered the cooperative operation and functionality of a swarm of robots while undertaking foraging tasks. Through realistic modelling of communication noise based on hardware experiments, the results from this research revealed the extent of the impact that this realistic communication has on the swarm. State-of-the-art machine learning based object detection was used as a fundamental tool for developing probabilistic vision model for robots swarm that conforms with recorded observations of real-world object detection data. The study also extended the application of the chemotaxis-based navigation technique to cover and address a swarm robotics area that has seen only little research in previous years, that is, the use of a virtual fencing mechanism to keep robots within a working area in unbounded environments.

In the RepAtt swarm foraging algorithm, robots searched for targets to forage using biased random walk and adaptively changed their turn/tumble probabilities based on attraction and repulsion signals they sensed. The algorithm presented a new approach for robots to use chemotaxis to improve their foraging performance. In RepAtt, the robots themselves acted as the source for attraction and repulsion signals whose intensity degraded exponentially with distance. This exponential degradation of the signal was modelled from real world experiments using white noise sound signal. The performance of RepAtt was tested on ten environmental setups that varied in size and nature of targets distribution in the search space. At the start of the foraging task, the robots leave the central nest in search for targets to pick up and, during the search phase, broadcast a repulsion signal to help the swarm spread out quickly to cover the search area. Once a robot locates a large deposit of targets beyond its carrying capacity, it broadcasts an attraction signal, which unsuccessful neighbouring swarm members sense and then use chemotaxis to locate the target cluster(s). The parameters that optimises a robot's chemotaxis behaviour were subdivided into attraction multiplier a_m, attraction divisor a_d, repulsion multiplier r_m and repulsion divisor r_d. An extensive parameters search of 900 combination of these chemotaxis parameters was conducted to obtain suitable values that optimise the swarm's foraging performance in the ten environment setups. By using optimised parameters, the swarm's foraging performance was significantly improved over the initial parameters. The most significant improvements were observed in highly clustered environments where communication was important for the swarm to exploit targets that were difficult to locate. In addition, scalability and robustness tests on RepAtt showed that it improved foraging efficiency with increasing swarm size and was robust to changes in targets distribution and world size.

Noisy communication was included in RepAtt to investigate its impact and how robots in the swarm could mitigate this unwanted but realistically present parameter. The noise model is based on the amount of deviation computed from real world sound experiments, which showed that it was approximately 6% of the average intensity of the sound signal at specific distance from the sound source. The integration of this noise model into the RepAtt algorithm initially had a significant negative impact on the swarm's foraging performance, making it no better than the uncoordinated Random Walk algorithm. However, with the inclusion of a simple noise filtering system based on computing average of limited queue of instantaneous signal intensity measurements, the swarm's performance during foraging notably improved. This improvement, which did not eliminate the negative impact the noise had on swarm foraging, was substantial, reducing foraging time by up to 70% when compared to Random Walk. In addition, the improvements in foraging efficiency as swarm size increased and the swarm's robustness to change in world setups showed that the RepAtt algorithm can significantly boost swarm foraging under realistic noisy communication conditions.

A comprehensive study on the effects of imperfect vision on a swarm of foraging robots has also been presented in this thesis. For this particular area of the research, a vision model based on observations of detection patterns of deep neural networks object detection models (tiny-YOLOv3 and MobileNet-SSD) was developed. The vision system model used two transition probabilities (P_{u2s} and P_{s2s}) to determine what a robot detected at every time step. The inference times of the object detection networks on Raspberry Pi 3 and 4 computers was also taken into consideration within swarm foraging simulations in order to ensure and maintain realistic conditions. The results show that RepAtt demonstrated further improvements on swarm foraging by a substantial magnitude when compared to Random Walk. The vision model was extended to cover a wider range of object detection quality and computational power to study their influence on swarm foraging and determine the point at which RepAtt's improvement over Random Walk would be insignificant. The variation in vision quality was achieved by performing simulations with about 196 different combinations of P_{u2s} and P_{s2s} probabilities. The variation in robots computational power was achieved by simulating vision update rates of 1 Hz, 4 Hz and 40 Hz. The results indicated that RepAtt still exhibited superior swarm foraging performance in comparison to Random Walk at detection probabilities as low as 0.2. This is good because it shows that RepAtt is able to adequately support swarm coordination in highly uncertain vision environments. The results also indicated that lower computational processors, where vision system was updated at lower frequency, impact swarm foraging in different ways depending on how robots used communicated signals. With no communication, increasing vision update rate helped the swarm improve performance during swarm foraging as robots vision quality degrades. However, when robots communicate with each other, increasing the vision update rate could have a negative impact on the swarm in certain scenarios. As the object detection probability P_s approaches 0.5, fast vision update rate could be detrimental for swarm's ability to cooperate during foraging because it increases the frequency of fluctuations between attraction and repulsion signal communication among the robots. These fluctuations could be minimised by reducing the rate at which robots decide which signal they broadcast (reducing vision update rates) or by applying a smoothing/average filter on the listening robot. Thus, improvement of vision quality is more important than investing in more expensive hardware that increases vision update rate (especially when no mechanism is in place to reduce the negative effect of imperfect vision on swarm recruitment).

In practical applications of swarm robotics such as space exploration, search and rescue operations, agriculture and other areas of field robotics, physical fencing mechanisms may not be present or feasible. In Chapter 6, a chemotaxis-based approach for keeping swarm of exploring robots within a desired work environment is described. Extensive simulations were used to show that the chemotactic fencing mechanism was as effective as a physical wall, and admissibly more practical. The chemotaxis approach used a nest to broadcast a homing signal that was sensed by exploring swarm robots. These robots activated a chemotactic navigation scheme which helped them return to the nest area whenever the signal intensity reduced beyond a threshold. The virtual fence was effective for both stationary and mobile nest. The system was also implemented on hardware robot platforms which involved using two robots (one nest and one explorer). The nest used an inverted cone to broadcast white noise sound, which the explorer robot sensed and used to perform chemotaxis to maximise its intensity instead of deviating from the nest over time.

The works presented in this thesis from Chapter 3 to 6 and summarised in the preceding paragraphs were instrumental in achieving the key objectives set out for this doctoral research. The following objectives have been attained:

1. A swarm foraging algorithm, called RepAtt, was implemented using inspiration from chemotaxis search behaviour of *C. elegans* nematode.

2. A swarm communication model was developed based on real world experiments with white noise sound signal. The model represented how sound signal degrades with

distance, quantified the level of noise in the communication channel and also specified how sound from multiple sources constructively superimpose on each other.

3. The swarm communication model was used in the RepAtt algorithm to analyse how it improves the swarm's foraging performance, scalability and robustness. The algorithm used an average filtering system to mitigate the impact of noise on the swarm's coordination.

4. A robot vision system was developed from analysing the detection characteristics of deep neural network based object detection algorithms. Real world experiments with a litter dataset was used to test the object detection algorithms and use the test analysis to implement a probabilistic robot vision model.

5. The vision model was used to extensively study the impact a robot's vision quality and processing power have on the collective behaviour of swarm foraging robots.

7.2 Conclusion

The extensive studies conducted on the chemotaxis search behaviour within the context of swarm robotics provide the basis for the following conclusions:

1. RepAtt provides a means of simplifying swarm foraging algorithms and robot hardware. The algorithm is scalable given that it does not require central coordination, neither does it require the use of ad hoc networks, nor beacons. It does not impose any complex mapping or localisation requirements on the swarm, thus simplifying hardware requirements for the swarm.

2. Studying RepAtt under noisy communication and imperfect vision revealed that the algorithm is robust under these conditions, thus making it more attractive for real-world implementations. The resulting vision model and to a lesser extent, a communication model will serve as a useful tool for other researchers in the field with which to test and examine the effect of realistic noise models on the performance of their algorithms. Therefore, providing beneficial and informative insight on the practicality of their proposed algorithmic solutions.

3. For swarm robots applications in unbounded environments, chemotaxis-based virtual fencing is an effective means of restraining robots from drifting from their work areas.

7.3 Further Work

The promising results obtained from the chemotaxis-based algorithm strongly suggests there is great potential for further work in this technically intriguing area. Recommendations for future advancements are summarised as follows:

1. An important aspect of the RepAtt algorithm is the communication of attraction and repulsion signals that degrade with increasing distance. In this thesis, white noise sound was used to represent this signal. However, an extensive comparative study of alternative technologies such as radio frequency, ultrasound, and Zigbee will be useful for finding the most suitable or scalable communication technology.

2. Implementing swarm foraging algorithm on hardware platform to solve real-world problems has always been the ultimate goal for swarm robotics. Since RepAtt lends itself to basic hardware features for the swarm robots, it would be beneficial to conduct thorough hardware-based experiments involving multiple robots.

3. Extending the RepAtt algorithm to include other features such as memory of previously visited sites is another area of interest and potential improvement. This additional feature could enhance RepAtt's robustness and foraging performance for certain applications where localisation is feasible and beneficial.

4. The RepAtt algorithm could also be applied to a foraging task in three-dimensional world. For example, the goal could be for a swarm of unmanned aerial vehicles tasked with foraging some desired targets in the environment. This could be interesting because Pólya's random walk constants states that random walk has less than unity probability of reaching all points in 3-dimensional search spaces [138, 139]. Thus, swarm foraging ability could be much worse in this search space, thus making a stronger case for the need of communication among swarm members.

5. One aspect that was not considered in this thesis is how the swarm robots reach a consensus on the time to stop or end the foraging task. This could be at an individual level where a robot gives up searching for targets and returns to the nest, or the swarm robots collectively decide that the foraging goal has been achieved. Further work can be done to explore different possibilities of achieving this behaviour in an optimal and realistic way.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, jun 2010. ISBN 978-1-4244-3992-8. doi: 10.1109/cvpr.2009.5206848.

[2] Avinash Gautam and Sudeept Mohan. A review of research in multi-robot systems. In *2012 IEEE 7th International Conference on Industrial and Information Systems, ICIIS 2012*, 2012. ISBN 9781467326056. doi: 10.1109/ICIInfS.2012.6304778.

[3] Lynne E Parker, Daniela Rus, and Gaurav S. Sukhatme. Multiple mobile robot systems. In *Springer Handbook of Robotics*, pages 1335–1379. Springer, 2016. ISBN 9783319325521. doi: 10.1007/978-3-319-32552-1_53.

[4] I-Ming Chen and Mark Yim. Modular robots. In *Springer Handbook of Robotics*, pages 531–542. Springer, 2016. doi: 10.1007/978-3-319-32552-1_22.

[5] Fabrizio Caccavale and Masaru Uchiyama. Cooperative manipulation. In *Springer Handbook of Robotics*, pages 989–1006. Springer, 2016. doi: 10.1007/978-3-319-32552-1_39.

[6] Vito Trianni and Alexandre Campo Alexandre. Fundamental collective behaviors in swarm robotics. In *Springer Handbook of Computational Intelligence*, pages 1377–1394. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 9783662435052. doi: 10.1007/978-3-662-43505-2_71.

[7] Eric. Bonabeau, Marco. Dorigo, and Guy. Theraulaz. *Swarm intelligence : from natural to artificial isystems*. Oxford University Press, 1999. ISBN 0195131592. URL http://jasss.soc.surrey.ac.uk/4/1/reviews/kluegl.html.

[8] Madeleine Beekman, Gregory A. Sword, and Stephen J. Simpson. *Biological Foundations of Swarm Intelligence*, pages 3–41. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-74089-6. doi: 10.1007/978-3-540-74089-6_1.

[9] Lori Lach, Catherine L. Parr, and Kirsti L. Abbott. *Ant Ecology*. Oxford University Press, nov 2010. ISBN 9780191720192. doi: 10.1093/acprof:oso/9780199544639.001.0001.

[10] Jan Carlo Barca and Y. Ahmet Sekercioglu. Swarm robotics reviewed. *Robotica*, (July 2012):1–15, 2012. ISSN 0263-5747. doi: 10.1017/S026357471200032X.

[11] Erol Sahin. Swarm Robotics: From Source to Inspiration to Domains of Application. *Special Issue", Autonomous Robots*, 3342:10–20, 2005. ISSN 0302-9743. doi: 10.1007/b105069.

[12] Ying Tan and Zhong-yang Zheng. Research Advance in Swarm Robotics. *Defence Technology*, 9(1):18–39, 2013. ISSN 22149147. doi: 10.1016/j.dt.2013.03.001.

[13] Levent Bayindir. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016. ISSN 18728286. doi: 10.1016/j.neucom.2015.05.116.

[14] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1): 1–41, 2013. ISSN 19353812. doi: 10.1007/s11721-012-0075-2.

[15] Kristina Smith. Self-repairing cities: Leeds' quest for an autonomous-robot maintenance army. *Construction Research and Innovation*, 9(4):91–94, oct 2018. ISSN 2045-0249. doi: 10.1080/20450249.2018.1556500.

[16] C. Baroni Urbani. The diversity and evolution of recruitment behaviour in ants, with a discussion of the usefulness of parsimony criteria in the reconstruction of evolutionary histories. *Insectes Sociaux*, 40(3):233–260, sep 1993. ISSN 00201812. doi: 10.1007/BF01242361.

[17] Klaus Jaffe. Negentropy and the evolution of chemical recruitment in ants (Hymenoptera: Formicidae). *Journal of Theoretical Biology*, 106(4):587–604, feb 1984. ISSN 10958541. doi: 10.1016/0022-5193(84)90009-2.

[18] Elizabeth L. Franklin and Nigel R. Franks. Individual and social learning in tandem-running recruitment by ants. *Animal Behaviour*, 84(2):361–368, aug 2012. ISSN 00033472. doi: 10.1016/j.anbehav.2012.05.002.

[19] I. D. Couzin and N. R. Franks. Self-organized lane formation and optimized traffic flow in army ants. *Proceedings of the Royal Society B: Biological Sciences*, 270(1511): 139–146, jan 2003. ISSN 14712970. doi: 10.1098/rspb.2002.2210.

[20] Carl Anderson. Teams in animal societies. *Behavioral Ecology*, 12(5):534–540, sep 2001. ISSN 1045-2249. doi: 10.1093/beheco/12.5.534.

[21] The Honeybee Conservancy. Honey Bees: Heroes of Our Planet - The Honeybee ConservancyThe Honeybee Conservancy, 2017. URL https://thehoneybeeconservancy.org/2017/06/22/honey-bees-heroes-planet/.

[22] Jerry Bromenshenk, Colin Henderson, Robert Seccomb, Phillip Welch, Scott Debnam, and David Firth. Bees as Biosensors: Chemosensory Ability, Honey Bee Monitoring Systems, and Emergent Sensor Technologies Derived from the Pollinator Syndrome. *Biosensors*, 5(4):678–711, oct 2015. ISSN 2079-6374. doi: 10.3390/bios5040678.

[23] HF Abou-Shaara. The foraging behaviour of honey bees, Apis mellifera: a review. *Veterinární Medicína*, 59(No. 1):1–10, feb 2014. ISSN 03758427. doi: 10.17221/7240-VETMED.

[24] Madeleine Beekman and Jie Bin Lew. Foraging in honeybees - When does it pay to dance? *Behavioral Ecology*, 19(2):255–262, 2008. ISSN 10452249. doi: 10.1093/beheco/arm117.

[25] Randolf Menzel, Uwe Greggers, Alan Smith, Sandra Berger, Robert Brandt, Sascha Brunke, Gesine Bundrock, Sandra Hü lse, Tobias Plü mpe, Frank Schaupp, Elke Schü ttler, Silke Stach, Jan Stindt, Nicola Stollhoff, and Sebastian Watzl. Honey bees navigate according to a map-like spatial memory. Technical report, 2005. URL www.pnas.orgcgidoi10.1073pnas.0408550102.

[26] Jacobus C Biesmeijer and E Judith Slaa. Information flow and organization of stingless bee foraging. *Apidologie*, 35(2):143–157, mar 2004. ISSN 0044-8435. doi: 10.1051/apido:2004003.

[27] James C Nieh and David W Roubik. A stingless bee (Melipona panamica) indicates food location without using a scent trail. *Behavioral Ecology and Sociobiology*, 37(1): 63–70, 1995. ISSN 14320762. doi: 10.1007/BF00173900.

[28] Angeles Mena Granero, José M. Guerra Sanz, Francisco J. Egea Gonzalez, José L. Martinez Vidal, Anna Dornhaus, Junaid Ghani, Ana Roldán Serrano, and Lars Chittka. Chemical compounds of the foraging recruitment pheromone in bumblebees. *Naturwissenschaften*, 92(8):371–374, aug 2005. ISSN 00281042. doi: 10.1007/s00114-005-0002-0.

[29] Mathieu Molet, Lars Chittka, Ralph J. Stelzer, Sebastian Streit, and Nigel E. Raine. Colony nutritional status modulates worker responses to foraging recruitment pheromone in the bumblebee Bombus terrestris. *Behavioral Ecology and Sociobiology*, 62(12):1919–1926, oct 2008. ISSN 03405443. doi: 10.1007/s00265-008-0623-3.

[30] Andy M. Reynolds. Lévy flight patterns are predicted to be an emergent property of a bumblebees' foraging strategy. *Behavioral Ecology and Sociobiology*, 64(1):19–23, 2009. ISSN 03405443. doi: 10.1007/s00265-009-0813-7.

[31] T. Hillen and K. J. Painter. A user's guide to PDE models for chemotaxis, jan 2009. ISSN 03036812.

[32] Vincenzo Fioriti, Fabio Fratichini, Stefano Chiesa, and Claudio Moriconi. Levy Foraging in a Dynamic Environment – Extending the Levy Search. *International Journal of Advanced Robotic Systems*, 12(7):98, jul 2015. ISSN 1729-8814. doi: 10.5772/60414.

[33] Surya G. Nurzaman, Yoshio Matsumoto, Yutaka Nakamura, Satoshi Koizumi, and Hiroshi Ishiguro. 'Yuragi'-based adaptive mobile robot search with and without gradient sensing: From bacterial chemotaxis to a Levy walk. *Advanced Robotics*, 25 (16):2019–2037, 2011. ISSN 01691864. doi: 10.1163/016918611X590229.

[34] Melinda D. Baker, Peter M. Wolanin, and Jeffry B. Stock. Systems biology of bacterial chemotaxis, apr 2006. ISSN 13695274.

[35] Samuel Ward. Chemotaxis by the nematode Caenorhabditis elegans: identification of attractants and analysis of the response by use of mutants. *Proceedings of the National Academy of Sciences of the United States of America*, 70(3):817–21, 1973. ISSN 0027-8424. doi: 10.1073/pnas.70.3.817.

[36] Surya G. Nurzaman, Yoshio Matsumoto, Yutaka Nakamura, Satoshi Koizumi, and Hiroshi Ishiguro. Biologically inspired adaptive mobile robot search with and without gradient sensing. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pages 142–147, St. Louis, USA, 2009. ISBN 9781424438044. doi: 10.1109/IROS.2009.5353998.

[37] G. M. Viswanathan, Sergey V. Buldyrev, Shlomo Havlin, M. G.E. Da Luz, E. P. Raposo, and H. Eugene Stanley. Optimizing the success of random searches. *Nature*, 401(6756):911–914, oct 1999. ISSN 00280836. doi: 10.1038/44831.

[38] E.N. Allen, Jing Ren, Yun Zhang, and Joy Alcedo. Sensory systems: their impact on C. elegans survival. *Neuroscience*, 296:15–25, jun 2015. ISSN 03064522. doi: 10.1016/j.neuroscience.2014.06.054.

[39] Noriyuki Ohnishi, Atsushi Kuhara, Fumiya Nakamura, Yoshifumi Okochi, and Ikue Mori. Bidirectional regulation of thermotaxis by glutamate transmissions in Caenorhabditis elegans. *EMBO Journal*, 30(7):1376–1388, apr 2011. ISSN 02614189. doi: 10.1038/emboj.2011.13.

[40] Thomas C. Ferrée and Shawn R. Lockery. Computational rules for chemotaxis in the nematode C. elegans. *Journal of Computational Neuroscience*, 6(3):263–277, 1999. ISSN 09295313. doi: 10.1023/A:1008857906763.

[41] Tau Mu Yi, Yun Huang, Melvin I. Simon, and John Doyle. Robust perfect adaptation in bacterial chemotaxis through integral feedback control. *Proceedings of the National Academy of Sciences of the United States of America*, 97(9):4649–4653, apr 2000. ISSN 00278424. doi: 10.1073/pnas.97.9.4649.

[42] Hans G Othmer and Angela Stevens. Aggregation, blowup, and collapse: The ABC'S of taxis in reinforced random walks. *SIAM Journal on Applied Mathematics*, 57(4): 1044–1081, 1997. ISSN 00361399. doi: 10.1137/s0036139995288976.

[43] Gabriela R. Andrade and Jordan H. Boyle. A minimal biologically-inspired algorithm for robots foraging energy in uncertain environments. *Robotics and Autonomous Systems*, 128:103499, jun 2020. ISSN 09218890. doi: 10.1016/j.robot.2020.103499.

[44] Ayusman Sen, Michael Ibele, Yiying Hong, and Darrell Velegol. Chemo and phototactic nano/microbots. *Faraday Discussions*, 143(0):15–27, sep 2009. ISSN 13596640. doi: 10.1039/b900971j.

[45] Onno D. Broekmans, Jarlath B. Rodgers, William S. Ryu, and Greg J. Stephens. Resolving coiled shapes reveals new reorientation behaviors in C. elegans. *eLife*, 5 (September), sep 2016. ISSN 2050084X. doi: 10.7554/eLife.17227.

[46] Jesse M. Gray, Joseph J. Hill, and Cornelia I. Bargmann. A circuit for navigation in Caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 102(9): 3184–3191, mar 2005. ISSN 0027-8424. doi: 10.1073/pnas.0409009101.

[47] Liliana C.M. Salvador, Frederic Bartumeus, Simon A. Levin, and William S. Ryu. Mechanistic analysis of the search behaviour of Caenorhabditis elegans. *Journal of the Royal Society Interface*, 11(92):20131092, mar 2014. ISSN 17425662. doi: 10.1098/rsif.2013.1092.

[48] Knut Drescher, Raymond E. Goldstein, and Idan Tuval. Fidelity of adaptive phototaxis. *Proceedings of the National Academy of Sciences of the United States of America*, 107 (25):11171–11176, jun 2010. ISSN 00278424. doi: 10.1073/pnas.1000901107.

[49] Daniela Kengyel, Thomas Schmickl, Heiko Hamann, Ronald Thenius, and Karl Crailsheim. Embodiment of honeybee's thermotaxis in a mobile robot swarm. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5778 LNAI, pages 69–76. Springer, Berlin, Heidelberg, 2011. ISBN 9783642213137. doi: 10.1007/978-3-642-21314-4_9.

[50] Mason Klein, Bruno Afonso, Ashley J. Vonner, Luis Hernandez-Nunez, Matthew Berck, Christopher J. Tabone, Elizabeth A. Kane, Vincent A. Pieribone, Michael N. Nitabach, Albert Cardona, Marta Zlatic, Simon G. Sprecher, Marc Gershow, Paul A. Garrity, Aravinthan D.T. Samuel, and Paul W. Sternberg. Sensory determinants of behavioral dynamics in Drosophila thermotaxis. *Proceedings of the National Academy of Sciences of the United States of America*, 112(2):E220–E229, jan 2015. ISSN 10916490. doi: 10.1073/pnas.1416212112.

[51] Andrew D. Horchler, Richard E. Reeve, Barbara Webb, and Roger D. Quinn. Robot phonotaxis in the wild: A biologically inspired approach to outdoor sound localization. In *Advanced Robotics*, volume 18, pages 801–816. Taylor & Francis Group, 2004. doi: 10.1163/1568553041738095.

[52] Jun Xian Shen, Albert S. Feng, Zhi Min Xu, Zu Lin Yu, Victoria S. Arch, Xin Jian Yu, and Peter M. Narins. Ultrasonic frogs show hyperacute phonotaxis to female courtship calls. *Nature*, 453(7197):914–916, jun 2008. ISSN 14764687. doi: 10.1038/nature06719.

[53] Ronald C. Arkin. Cooperation without Communication: Multi-Agent Schema-Based Robot Navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992. doi: 10.1002/rob.4620090304.

[54] R.C. Arkin, T. Balch, and E. Nitz. Communication of behavorial state in multiagent retrieval tasks. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 588–594. IEEE Comput. Soc. Press, 1993. ISBN 0-8186-3450-2. doi: 10.1109/ROBOT.1993.291841.

[55] Tucker Balch and Ronald C Arkin. Communication in Reactive Multiagent Robotic Systems. *Autonomous Robots*, 1:27–52, 1994. URL https://link.springer.com/content/pdf/10.1007{%}2FBF00735341.pdf.

[56] Lenka Pitonakova, Richard Crowder, and Seth Bullock. Understanding the role of recruitment in collective robot foraging. In *Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14)*, pages 264–271. The MIT Press, jul 2014. ISBN 9780262326216. doi: 10.7551/978-0-262-32621-6-ch043.

[57] Lenka Pitonakova, Richard Crowder, and Seth Bullock. Information flow principles for plasticity in foraging robot swarms. *Swarm Intelligence*, 10(1):33–63, mar 2016. ISSN 1935-3812. doi: 10.1007/s11721-016-0118-1.

[58] Iñaki Fernández Pérez, Amine Boumaza, and François Charpillet. Learning Collaborative Foraging in a Swarm of Robots using Embodied Evolution. In *ECAL 2017 – 14th European Conference on Artificial Life*, Lyon, sep 2017. Inria. URL https://hal.archives-ouvertes.fr/hal-01534242/.

[59] Yifan Cai and Simon X. Yang. A PSO-based approach to cooperative foraging tasks of multi-robots in completely unknown environments. In *2014 World Automation Congress (WAC)*, pages 813–822. IEEE, aug 2014. ISBN 978-1-8893-3549-0. doi: 10.1109/WAC.2014.6936157.

[60] Joshua P. Hecker and Melanie E. Moses. Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms. *Swarm Intelligence*, 9(1):43–70, feb 2015. ISSN 1935-3812. doi: 10.1007/s11721-015-0104-z.

[61] Joshua P. Hecker, Justin Craig Carmichael, and Melanie E. Moses. Exploiting clusters for complete resource collection in biologically-inspired robot swarms. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 434–440. IEEE, sep 2015. ISBN 978-1-4799-9994-1. doi: 10.1109/IROS.2015.7353409.

[62] Qi Lu, Joshua P. Hecker, and Melanie E. Moses. The MPFA: A multiple-place foraging algorithm for biologically-inspired robot swarms. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3815–3821. IEEE, oct 2016. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7759561.

[63] Qi Lu, Joshua P. Hecker, and Melanie E. Moses. Multiple-place swarm foraging with dynamic depots. *Autonomous Robots*, 42(4):909–926, 2018. ISSN 15737527. doi: 10.1007/s10514-017-9693-2.

[64] Nicholas Hoff. *Multi-Robot Foraging for Swarms of Simple Robots*. Doctoral thesis, Harvard University, 2011. URL http://www.eecs.harvard.edu/ssr/papers/phd11-hoff.pdf.

[65] Nicholas Hoff, Robert Wood, and Radhika Nagpal. Distributed colony-level algorithm switching for robot swarm foraging. In *Springer Tracts in Advanced Robotics*, volume 83 STAR, pages 417–430. Springer Berlin Heidelberg, 2012. ISBN 9783642327223. doi: 10.1007/978-3-642-32723-0_30.

[66] Ouarda Zedadra, Hamid Seridi, Nicolas Jouandeau, and Giancarlo Fortino. A distributed foraging algorithm based on artificial potential field. In *12th International Symposium on Programming and Systems, ISPS 2015*, pages 201–206. IEEE, apr 2015. ISBN 9781479976997. doi: 10.1109/ISPS.2015.7244986.

[67] G. Matthew Fricke, Joshua P. Hecker, Antonio D. Griego, Linh T. Tran, and Melanie E. Moses. A distributed deterministic spiral search algorithm for swarms. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4430–4436. IEEE, oct 2016. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7759652.

[68] Hanyi Dai. *Adaptive Control in Swarm Robotic Systems*, volume 3. Western Michigan University, Graduate Student Advisory Committee, sep 2009. URL http://scholarworks.wmich.edu/hilltopreview/vol3/iss1/7.

[69] Eduardo Castello, Tomoyuki Yamamoto, Fabio Dalla Libera, Wenguo Liu, Alan F. T. Winfield, Yutaka Nakamura, and Hiroshi Ishiguro. Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach. *Swarm Intelligence*, 10(1):1–31, mar 2016. ISSN 1935-3812. doi: 10.1007/s11721-015-0117-7.

[70] Guang Zhou, Farokh Bastani, Wei Zhu, and I-Ling Yen. A Self-Stabilizing Algorithm for the Foraging Problem in Swarm Robotic Systems. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2907–2912. IEEE, oct 2016. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7759450.

[71] Giovanni Pini, Arne Brutschy, Carlo Pinciroli, Marco Dorigo, and Mauro Birattari. Autonomous task partitioning in robot foraging: An approach based on cost estimation. *Adaptive Behavior*, 21(2):118–136, 2013. ISSN 10597123. doi: 10.1177/1059712313484771.

[72] Jong Hyun Lee and Chang Wook Ahn. Improving energy efficiency in cooperative foraging swarm robots using behavioral model. In *Proceedings - 2011 6th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2011*, pages 39–44. IEEE, sep 2011. ISBN 9780769545141. doi: 10.1109/BIC-TA.2011.69.

[73] Danielli A. Lima and Gina M. B. Oliveira. A probabilistic cellular automata ant memory model for a swarm of foraging robots. In *2016 14th International Conference on Control, Automation, Robotics and Vision, ICARCV 2016*, pages 1–6. IEEE, nov 2017. ISBN 9781509035496. doi: 10.1109/ICARCV.2016.7838615.

[74] Danielli A. Lima and Gina M. B. Oliveira. A cellular automata ant memory model of foraging in a swarm of robots. *Applied Mathematical Modelling*, 47:551–572, jul 2017. ISSN 0307904X. doi: 10.1016/j.apm.2017.03.021.

[75] Ralf Mayet, Jonathan Roberz, Thomas Schmickl, and Karl Crailsheim. Antbots: A feasible visual emulation of pheromone trails for swarm robots. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6234 LNCS, pages 84–94. Springer, Berlin, Heidelberg, 2010. ISBN 3642154603. doi: 10.1007/978-3-642-15461-4_8.

[76] Ali Abdul Khaliq, Maurizio Di Rocco, and Alessandro Saffiotti. Stigmergic algorithms for multiple minimalistic robots on an RFID floor. *Swarm Intelligence*, 8(3):199–225, sep 2014. ISSN 19353820. doi: 10.1007/s11721-014-0096-0.

[77] Anna Font Llenas, Mohamed S. Talamali, Xu Xu, James A.R. Marshall, and Andreagiovanni Reina. Quality-sensitive foraging by a robot swarm through virtual pheromone trails. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11172 LNCS, pages 135–149. Springer Verlag, oct 2018. ISBN 9783030005320. doi: 10.1007/978-3-030-00533-7_11.

[78] Mohamed S. Talamali, Thomas Bose, Matthew Haire, Xu Xu, James A.R. Marshall, and Andreagiovanni Reina. Sophisticated collective foraging with minimalist agents: a swarm robotics test. *Swarm Intelligence*, 14(1):25–56, mar 2020. ISSN 19353820. doi: 10.1007/s11721-019-00176-9.

[79] Katherine Russell, Michael Schader, Kevin Andrea, and Sean Luke. Swarm robot foraging with wireless sensor motes. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, volume 1, pages 287–295, Istanbul, 2015. ISBN 9781450337694. URL www.ifaamas.org.

[80] Seongin Na, Mohsen Raoufi, Ali Emre Turgut, Tomáš Krajník, and Farshad Arvin. Extended Artificial Pheromone System for Swarm Robotic Applications. In *The 2019 Conference on Artificial Life*, pages 608–615, Cambridge, MA, jul 2019. MIT Press. ISBN 978-0-262-35844-6. doi: 10.1162/isal_a_00228.

[81] Xuelong Sun, Tian Liu, Cheng Hu, Qinbing Fu, and Shigang Yue. ColCOS Φ: AAA multiple pheromone communication system for swarm robotics and social insects research. In *2019 4th IEEE International Conference on Advanced Robotics and Mechatronics, ICARM 2019*, pages 59–66. Institute of Electrical and Electronics Engineers Inc., jul 2019. ISBN 9781728100647. doi: 10.1109/ICARM.2019.8833989.

[82] Simon Garnier, Fabien Tâche, Maud Combe, Anne Grimal, and Guy Theraulaz. Alice in pheromone land: An experimental setup for the study of ant-like robots. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium, SIS 2007*, pages 37–44, 2007. ISBN 1424407087. doi: 10.1109/SIS.2007.368024.

[83] Thomas Schmickl and Heiko Hamann. BEECLUST: A swarm algorithm derived from honeybees: Derivation of the algorithm, analysis by mathematical models, and implementation on a robot swarm. In Yang Xiao, editor, *Bio-Inspired Computing and Networking*, pages 95–137. 2011. ISBN 9781420080339.

[84] R. Andrew Russell. Ant trails - an example for robots to follow? *Proceedings - IEEE International Conference on Robotics and Automation*, 4:2698–2703, 1999. ISSN 10504729. doi: 10.1109/robot.1999.774005.

[85] Ryusuke Fujisawa, Shigeto Dobata, Ken Sugawara, and Fumitoshi Matsuno. Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. *Swarm Intelligence*, 8(3):227–246, sep 2014. ISSN 19353820. doi: 10.1007/s11721-014-0097-z.

[86] Frederick Ducatelle, Gianni A Di Caro, Carlo Pinciroli, Luca M Gambardella, F Ducatelle, Ga Di Caro, LM Gambardella, and C Pinciroli. Self-organized cooperation between robotic swarms. *Swarm Intell*, 5:73–96, 2011. doi: 10.1007/s11721-011-0053-0.

[87] Alexandre Campo, Álvaro Gutiérrez, Shervin Nouyan, Carlo Pinciroli, Valentin Longchamp, Simon Garnier, and Marco Dorigo. Artificial pheromone for path selection by a foraging swarm of robots. *Biological Cybernetics*, 103(5):339–352, nov 2010. ISSN 03401200. doi: 10.1007/s00422-010-0402-x.

[88] T. Schmickl and K. Crailsheim. Trophallaxis within a robotic swarm: Bio-inspired communication among robots in a swarm. *Autonomous Robots*, 25(1-2):171–188, aug 2008. ISSN 09295593. doi: 10.1007/s10514-007-9073-4.

[89] Shervin Nouyan, Roderich Gross, Michael Bonani, Francesco Mondada, and Marco Dorigo. Teamwork in Self-Organized Robot Colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711, aug 2009. ISSN 1089-778X. doi: 10.1109/TEVC.2008.2011746.

[90] P. Valdastri, P. Corradi, A. Menciassi, T. Schmickl, K. Crailsheim, J. Seyfried, and P. Dario. Micromanipulation, communication and swarm intelligence issues in a swarm microrobotic platform. *Robotics and Autonomous Systems*, 54(10):789–804, oct 2006. ISSN 09218890. doi: 10.1016/j.robot.2006.05.001.

[91] Roman Miletitch, Vito Trianni, Alexandre Campo, and Marco Dorigo. Information Aggregation Mechanisms in Social Odometry. In *Advances in Artificial Life, ECAL 2013*, pages 102–109. MIT Press, sep 2013. ISBN 9780262317092. doi: 10.7551/978-0-262-31709-2-ch016.

[92] Michael Bonani, Valentin Longchamp, Stéphane Magnenat, Philippe Rétornaz, Daniel Burnier, Gilles Roulet, Florian Vaussard, Hannes Bleuler, and Francesco Mondada. The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 4187–4193. IEEE, oct 2010. ISBN 9781424466757. doi: 10.1109/IROS.2010.5649153.

[93] Álvaro Gutiérrez, Alexandre Campo, Félix Monasterio-Huelin, Luis Magdalena, and Marco Dorigo. Collective decision-making based on social odometry. *Neural Computing and Applications*, 19(6):807–823, sep 2010. ISSN 09410643. doi: 10.1007/s00521-010-0380-x.

[94] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object Detection in 20 Years: A Survey. may 2019. URL http://arxiv.org/abs/1905.05055.

[95] Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, may 2004. ISSN 09205691. doi: 10.1023/B:VISI.0000013087.49260.fb.

[96] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, volume I, pages 886–893, 2005. ISBN 0769523722. doi: 10.1109/CVPR.2005.177.

[97] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. ISSN 01628828. doi: 10.1109/TPAMI.2009.167.

[98] Stephen Gould, Tianshi Gao, and Daphne Koller. Region-based Segmentation and Object Detection. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 655–663. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3766-region-based-segmentation-and-object-detection.pdf.

[99] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, apr 2001. ISSN 01628828. doi: 10.1109/34.917571.

[100] Gong Cheng and Junwei Han. A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:11–28, jul 2016. ISSN 09242716. doi: 10.1016/j.isprsjprs.2016.03.014.

[101] Nima Razavi, Juergen Gall, and Luc Van Gool. Scalable multi-class object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1505–1512. IEEE Computer Society, 2011. ISBN 9781457703942. doi: 10.1109/CVPR.2011.5995441.

[102] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 6517–6525. IEEE, jul 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.690.

[103] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587. IEEE Computer Society, sep 2014. ISBN 9781479951178. doi: 10.1109/CVPR.2014.81.

[104] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 1440–1448. IEEE, apr 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.169.

[105] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, jun 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2577031.

[106] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8691 LNCS, pages 346–361. Springer Verlag, 2014. ISBN 9783319105772. doi: 10.1007/978-3-319-10578-9_23.

[107] Tsung Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 936–944. IEEE, jul 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.106.

[108] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2939201.

[109] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 779–788. IEEE Computer Society, dec 2016. ISBN 9781467388504. doi: 10.1109/CVPR.2016.91.

[110] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. apr 2018. URL http://arxiv.org/abs/1804.02767.

[111] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9905 LNCS, pages 21–37. Springer Verlag, 2016. ISBN 9783319464473. doi: 10.1007/978-3-319-46448-0_2.

[112] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, feb 2020. ISSN 19393539. doi: 10.1109/TPAMI.2018.2858826.

[113] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. apr 2017. URL http://arxiv.org/abs/1704.04861.

[114] Jing Ma, Li Chen, and Zhiyong Gao. Hardware implementation and optimization of tiny-YOLO network. In *Communications in Computer and Information Science*, volume 815, pages 224–234. Springer Verlag, 2018. ISBN 9789811081071. doi: 10.1007/978-981-10-8108-8_21.

[115] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, jun 2010. ISSN 09205691. doi: 10.1007/s11263-009-0275-4.

[116] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8693 LNCS, pages 740–755. Springer Verlag, 2014. doi: 10.1007/978-3-319-10602-1_48.

[117] Kent Gauen, Ryan Dailey, John Laiman, Yuxiang Zi, Nirmal Asokan, Yung-Hsiang Lu, George K Thiruvathukal, Mei-Ling Shyu, and Shu-Ching Chen. Comparison of Visual Datasets for Machine Learning. In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, volume 2017-Janua, pages 346–355. IEEE, aug 2017. ISBN 978-1-5386-1562-1. doi: 10.1109/IRI.2017.59.

[118] Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. Collecting Image Annotations Using Amazon's Mechanical Turk. In *NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, number June, pages 139–147, 2010.

[119] Boris Sekachev, Zhavoronkov Andrey, and Manovich Nikita. Computer Vision Annotation Tool: A Universal Approach to Data Annotation, 2019. URL https://software.intel.com/content/www/us/en/develop/articles/computer-vision-annotation-tool-a-universal-approach-to-data-annotation.htmlhttps://github.com/opencv/cvat.

[120] J. Cartucho, R. Ventura, and M. Veloso. Robust object recognition through symbiotic deep learning in mobile robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2336–2341, 2018.

[121] Peng Sun, Wansen Feng, Ruobing Han, Shengen Yan, and Yonggang Wen. Optimizing Network Performance for Distributed DNN Training on GPU Clusters: ImageNet/AlexNet Training in 1.5 Minutes. feb 2019. URL http://arxiv.org/abs/1902.06855.

[122] Andrea-Orsolya Fulop and Levente Tamas. Lessons learned from lightweight CNN based object recognition for mobile robots. In *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–5. IEEE, may 2018. ISBN 978-1-5386-2205-6. doi: 10.1109/AQTR.2018.8402778.

[123] Tucker Balch, Gary Boone, Thomas Collins, Harold Forbes, Doug MacKenzie, and Juan Carlos Santamar. Io, Ganymede, and Callisto A Multiagent Robot Trash-Collecting Team. *AI Magazine*, 16(2):39, jun 1995. ISSN 0738-4602. doi: 10.1609/AIMAG.V16I2.1132.

[124] E Prassler, E Stroulia, and M Strobe1. Office Waste Cleanup: An Application For Service Robots - Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on. *IEEE International Conference on Robotics and Automation Albuquerque*, (April):1863–1868, 1997.

[125] Gabriele Ferri, Alessandro Manzi, Pericle Salvini, Barbara Mazzolai, Cecilia Laschi, and Paolo Dario. DustCart, an autonomous robot for door-to-door garbage collection: From DustBot project to the experimentation in the small town of Peccioli. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 655–660, 2011. ISSN 10504729. doi: 10.1109/ICRA.2011.5980254.

[126] Matteo Reggente, Alessio Mondini, Gabriele Ferri, Barbara Mazzolai, Alessandro Manzi, Matteo Gabelletti, Paolo Dario, and Achim J. Lilienthal. The DustBot system: Using mobile robots to monitor pollution in pedestrian area. *Chemical Engineering Transactions*, (2002), 2010. ISSN 19749791. doi: 10.3303/CET1023046.

[127] Jordi Palacín, José Antonio Salse, Ignasi Valganón, and Xavi Clua. Building a mobile robot for a floor-cleaning operation in domestic environments. *IEEE Transactions on Instrumentation and Measurement*, 53(5):1418–1424, oct 2004. ISSN 00189456. doi: 10.1109/TIM.2004.834093.

[128] Shreyas Kulkarni and Sarang Junghare. Robot based indoor autonomous trash detection algorithm using ultrasonic sensors. In *CARE 2013 - 2013 IEEE International Conference on Control, Automation, Robotics and Embedded Systems, Proceedings*, 2013. ISBN 9781467361538. doi: 10.1109/CARE.2013.6733698.

[129] Mun Cheon Kang, Kwang Shik Kim, Dong Ki Noh, Jong Woo Han, and Sung Jea Ko. A robust obstacle detection method for robotic vacuum cleaners. *IEEE Transactions on Consumer Electronics*, 60(4):587–595, nov 2014. ISSN 00983063. doi: 10.1109/TCE.2014.7027291.

[130] M. A.Viraj J. Muthugala, S. M.Bhagya P. Samarakoon, and Mohan Rajesh Elara. Tradeoff between Area Coverage and Energy Usage of a Self-Reconfigurable Floor Cleaning Robot Based on User Preference. *IEEE Access*, 8:76267–76275, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.2988977.

[131] Stephen Balakirsky, Elena Messina, Craig Schlenoff, Scott Smith, and Michael Uschold. Knowledge representation for a trash collecting robot: Results from the 2004 AAAI Spring Symposium. *Robotics and Autonomous Systems*, 49(1-2 SPEC. ISS.):7–12, 2004. ISSN 09218890. doi: 10.1016/j.robot.2004.07.013.

[132] Jinqiang Bai, Shiguo Lian, Zhaoxiang Liu, Kai Wang, and Dijun Liu. Deep Learning Based Robot for Automatically Picking Up Garbage on the Grass. *IEEE Transactions on Consumer Electronics*, 64(3):382–389, aug 2018. ISSN 15584127. doi: 10.1109/TCE.2018.2859629.

[133] Antonio Luca Alfeo, Eduardo Castello Ferrer, Yago Lizarribar Carrillo, Arnaud Grignard, Luis Alonso Pastor, Dylan T. Sleeper, Mario G.C.A. Cimino, Bruno Lepri, Gigliola Vaglini, Kent Larson, Marco Dorigo, and Alex Sandy Pentland. Urban swarms: A new approach for autonomous waste management. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2019-May, pages 4233–4240. IEEE, may 2019. ISBN 9781538660263. doi: 10.1109/ICRA.2019.8794020.

[134] Lenka Pitonakova, Richard Crowder, and Seth Bullock. The Information-Cost-Reward framework for understanding robot swarm foraging. *Swarm Intelligence*, 12(1):71–96, 2018. ISSN 19353820. doi: 10.1007/s11721-017-0148-3.

[135] Stefan Ropke and David Pisinger. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 40 (4):455–472, nov 2006. ISSN 0041-1655. doi: 10.1287/trsc.1050.0135.

[136] Peng Yu, Rua Yan, and Li Yao. Measurement of Acoustic Attenuation Coefficient of Stored Grain. In *3rd International Conference on Control, Automation and Robotics*, pages 551–554, 2017. ISBN 9781509060887.

[137] Serena Ivaldi, Vincent Padois, and Francesco Nori. Tools for dynamics simulation of robots: a survey based on user feedback. *CoRR*, 2014. URL https://arxiv.org/pdf/1402.7050.pdf.

[138] Michael Kozdron. An Introduction to Random Walks from P ´ olya to. pages 1–22, 1998.

[139] Eric W. Weisstein. Random Walk–3-Dimensional. URL https://mathworld.wolfram.com/RandomWalk3-Dimensional.html.

Appendix A: RepAtt: Noiseless Communication

This appendix contains all results for parameters optimization and scalability simulations in Chapter 3. Only a few of the results were included in the body of the thesis for the purpose of readability. All results presented here are based on 30 repetitions of each simulation.

A.1 Parameters Optimisation

Task is for swarm of 36 robots to forage 180 out of 200 targets in the specified environments. The chemotaxis parameters optimization involves a parameter sweep experiment where a_m and r_m were selected from 1, 2, 4, 6, 8 and 10, while a_d and r_d were selected from 1, 10, 50, 100, 1000. This resulted in 900 different combinations of the chemotaxis parameters. All the foraging time results were normalised using Random Walk foraging time.

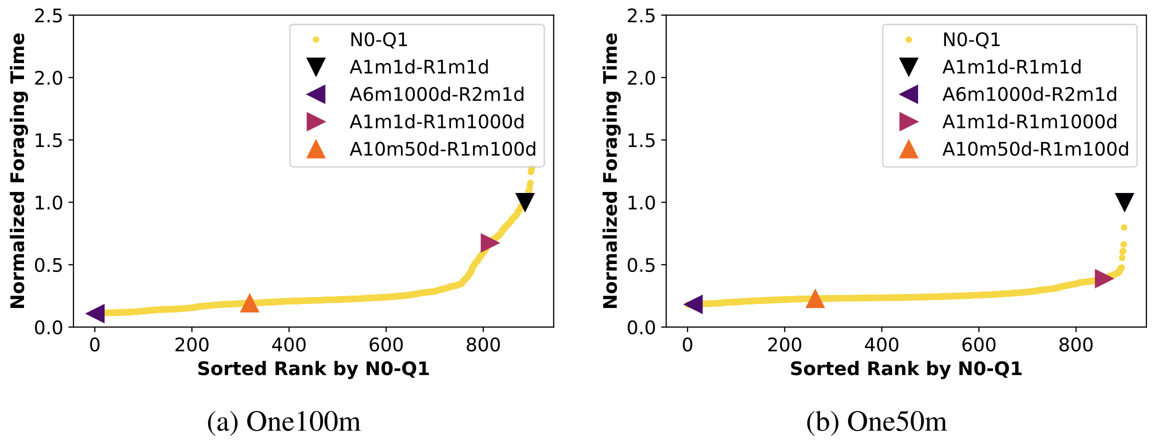
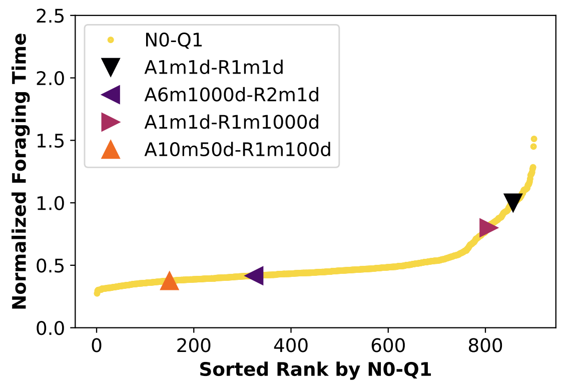
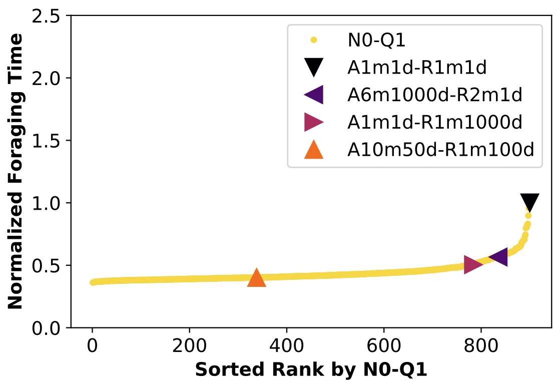


Fig. A.1 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.

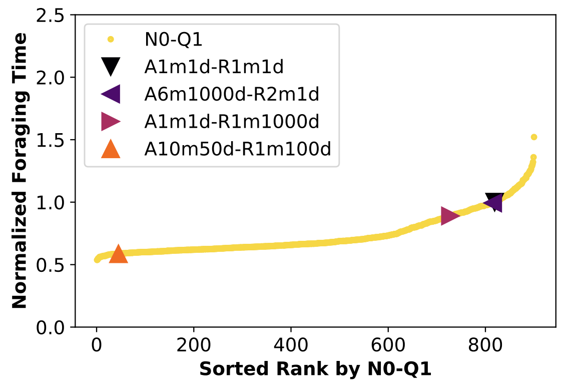


(a) Two100m

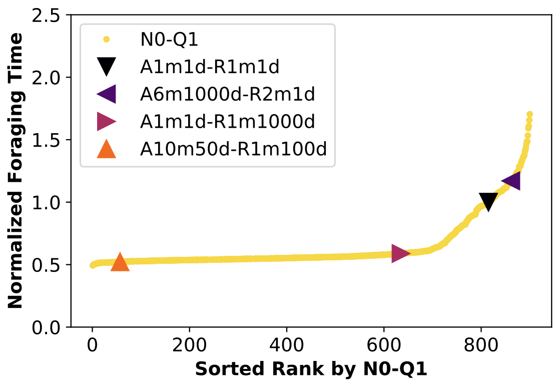


(b) Two50m

Fig. A.2 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.

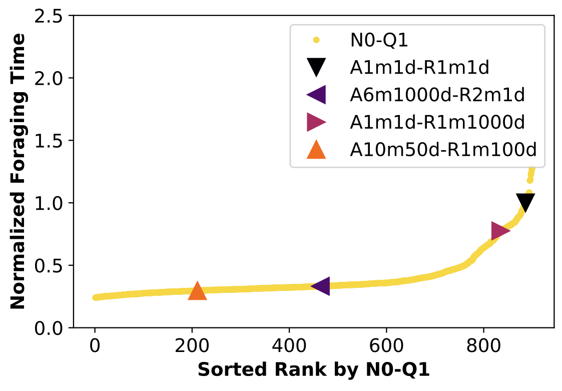


(a) Four100m

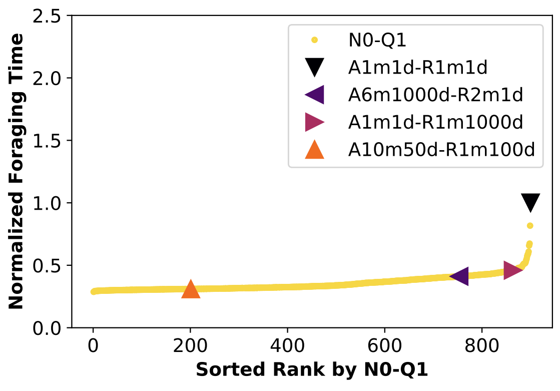


(b) Four50m

Fig. A.3 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.

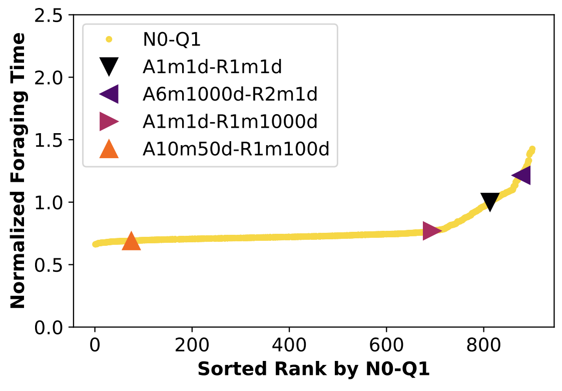


(a) Half100m

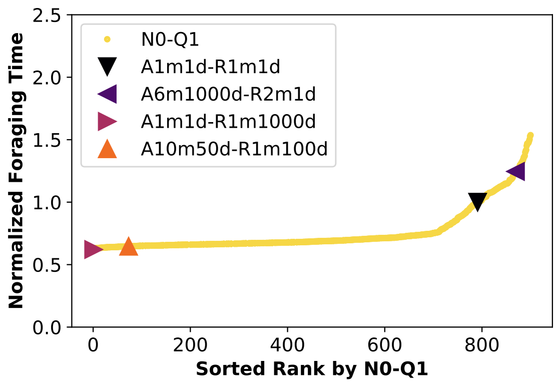


(b) Half50m

Fig. A.4 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.



(a) Uniform100m



(b) Uniform50m

Fig. A.5 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.

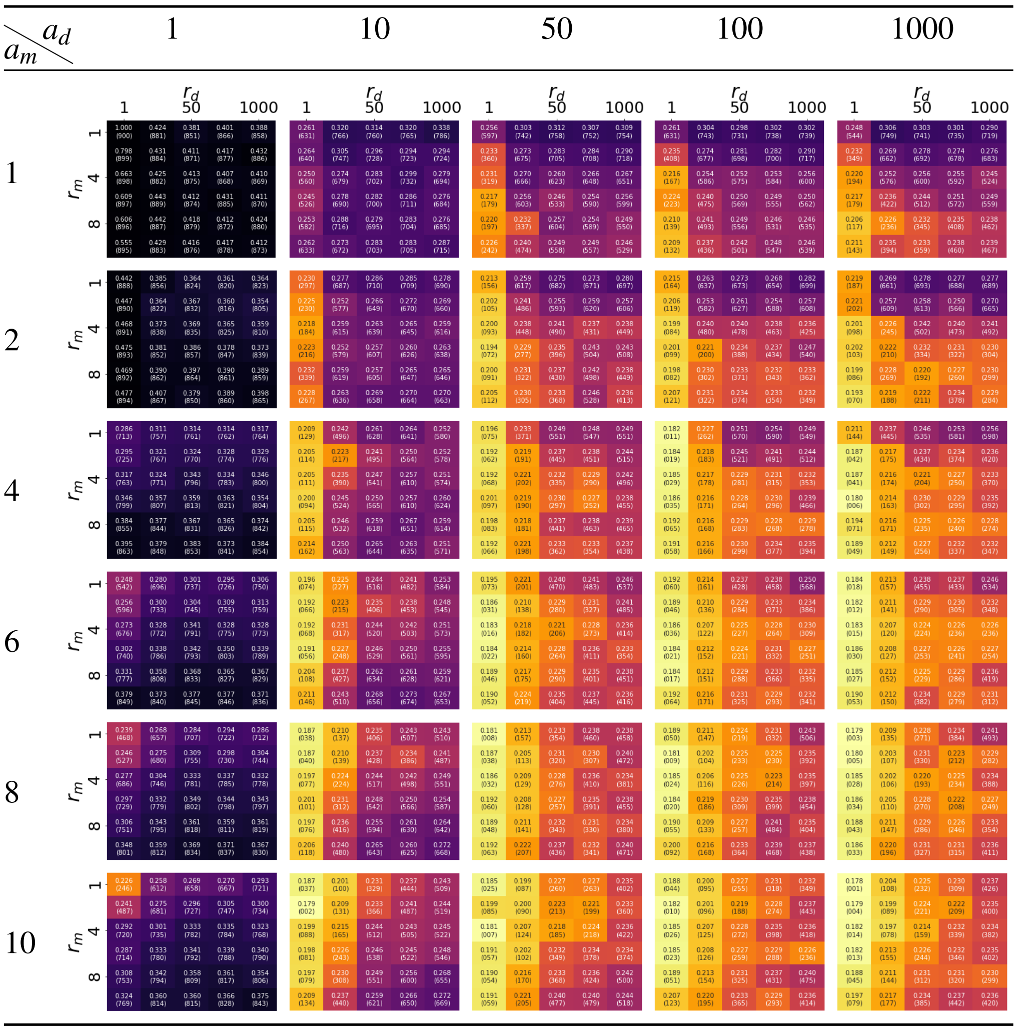


Fig. A.6 Effect of chemotaxis parameters on foraging performance in the One50m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 1000$, $r_m = 1$, $r_d = 1$.

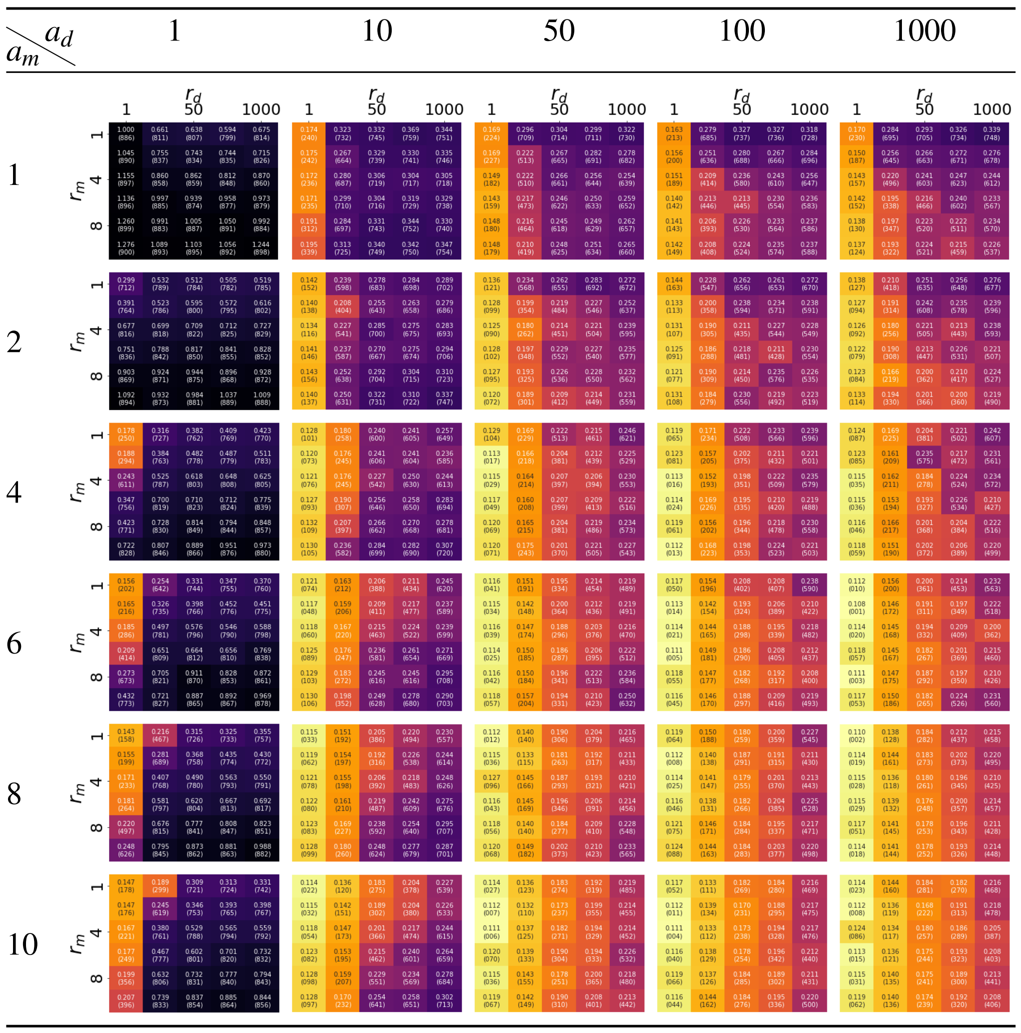


Fig. A.7 Effect of chemotaxis parameters on foraging performance in the One100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 2$, $r_d = 1$.

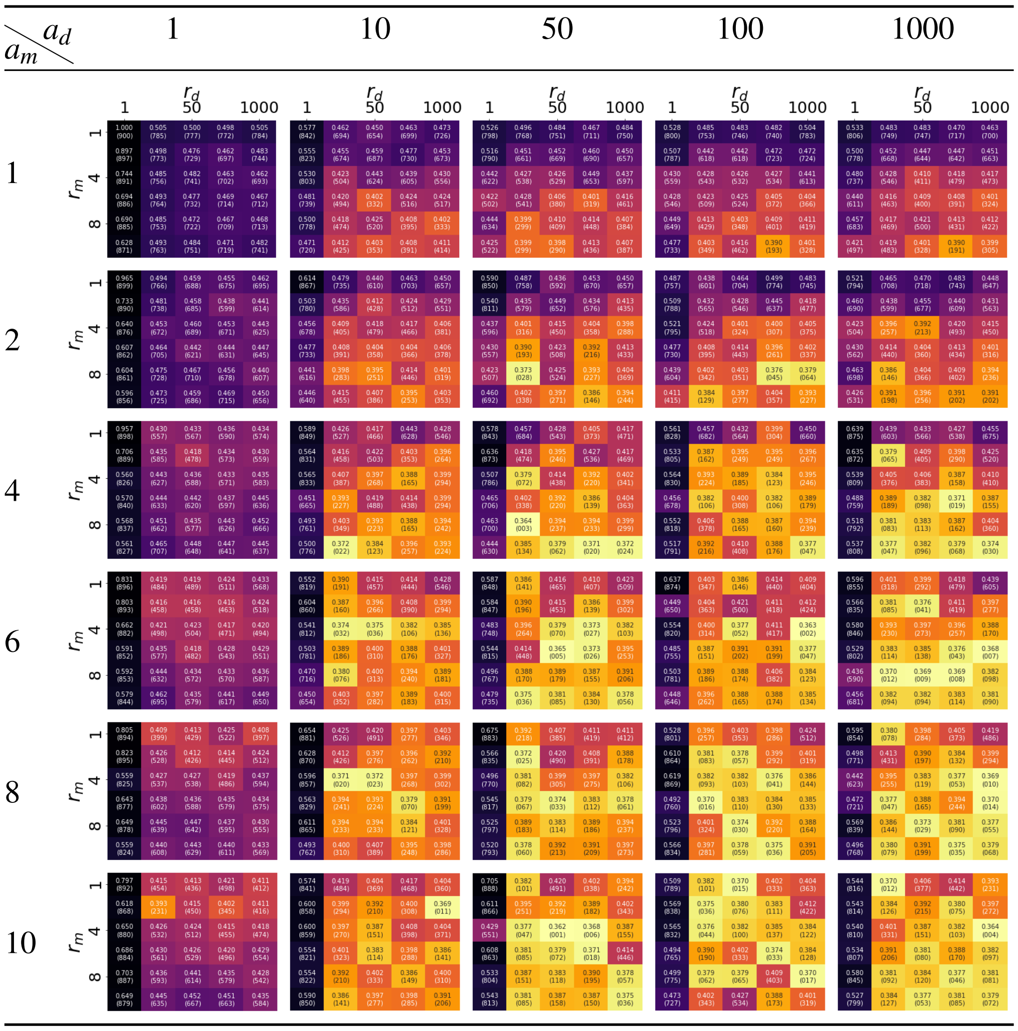


Fig. A.8 Effect of chemotaxis parameters on foraging performance in the Two50m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 4$, $r_d = 50$.

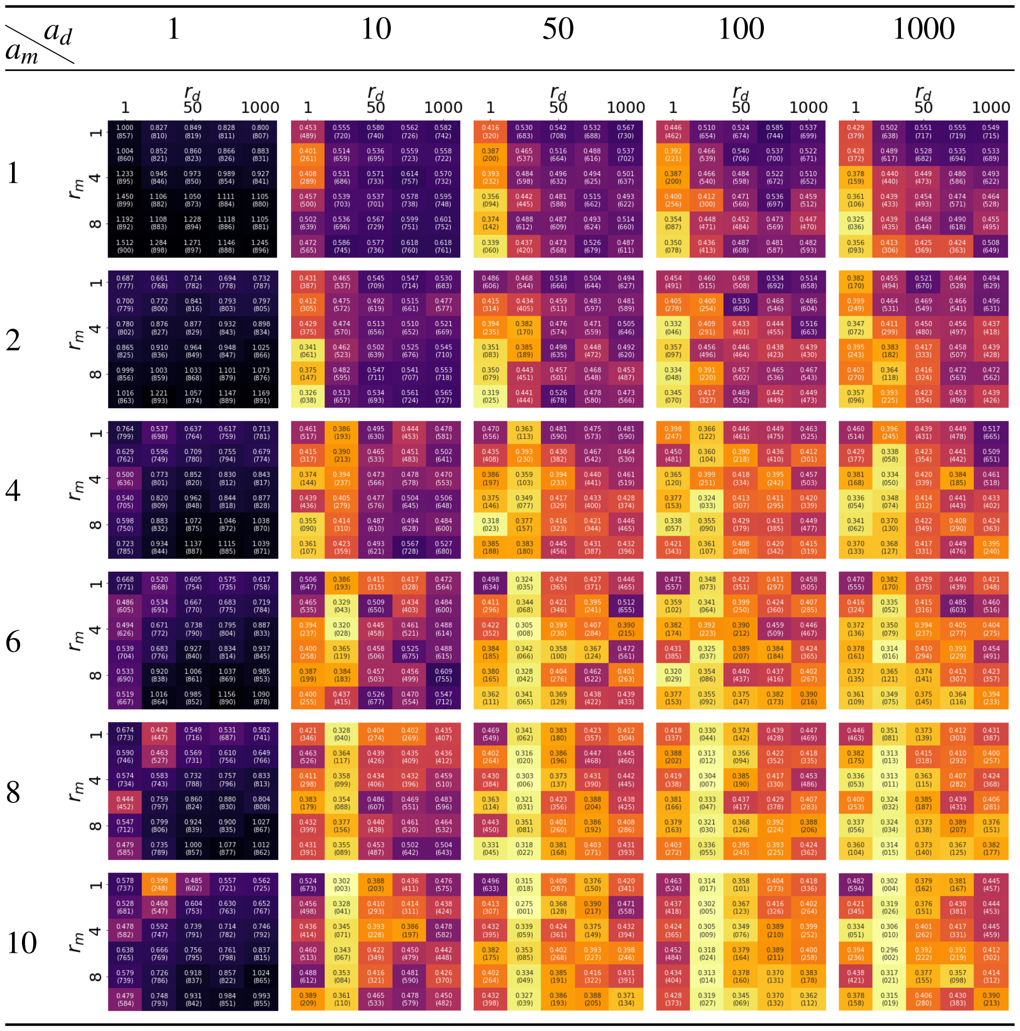


Fig. A.9 Effect of chemotaxis parameters on foraging performance in the Two100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 2$, $r_d = 10$.

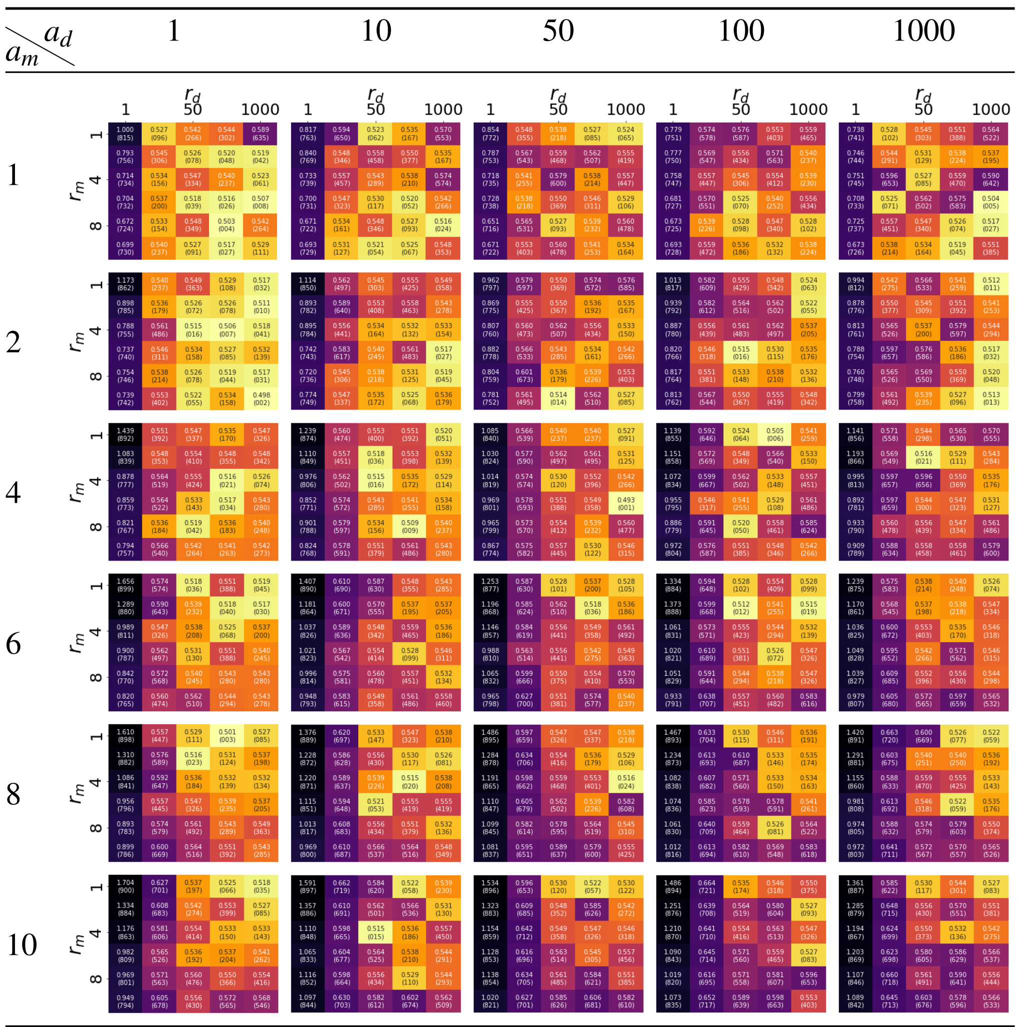


Fig. A.10 Effect of chemotaxis parameters on foraging performance in the Four50m targets distribution. Best combination for this distribution is $a_m = 4$, $a_d = 50$, $r_m = 6$, $r_d = 1000$.

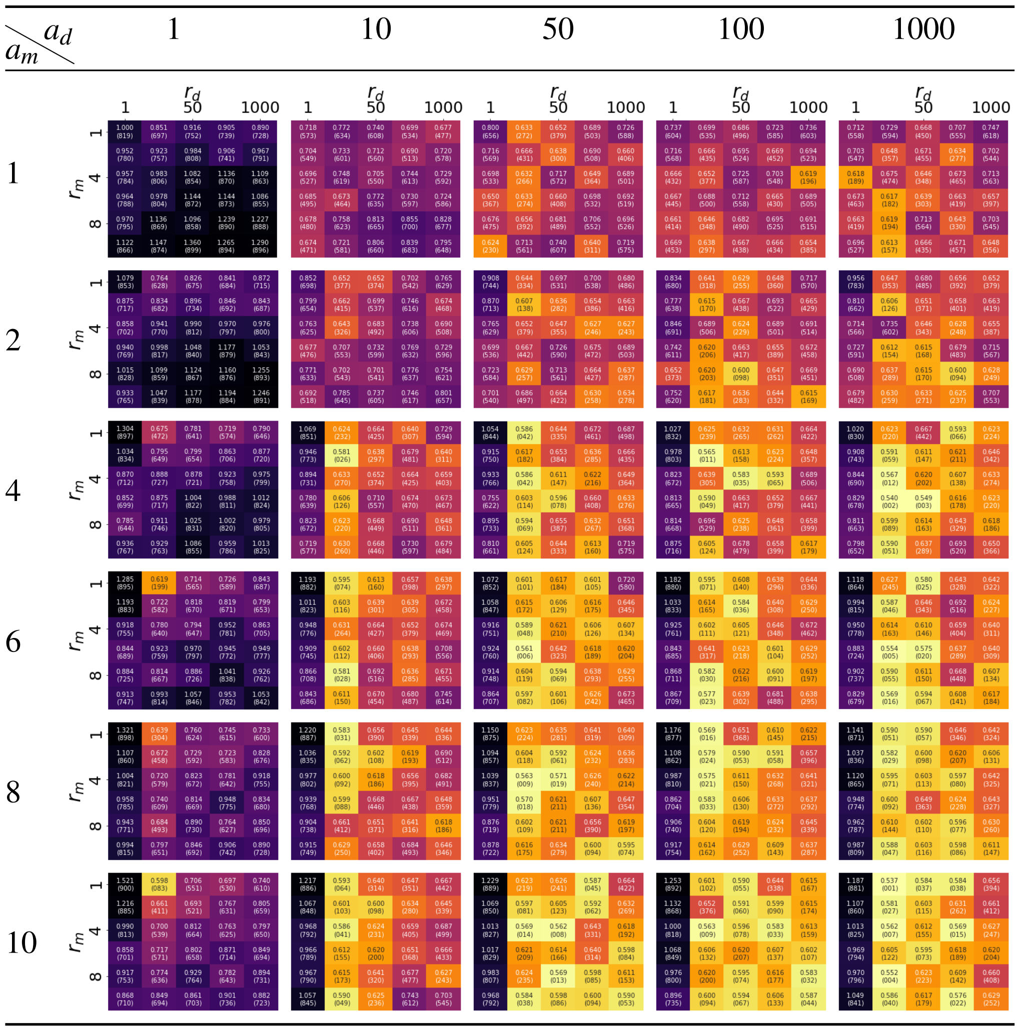


Fig. A.11 Effect of chemotaxis parameters on foraging performance of swarm in the Four100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 1000$, $r_m = 1$, $r_d = 10$.

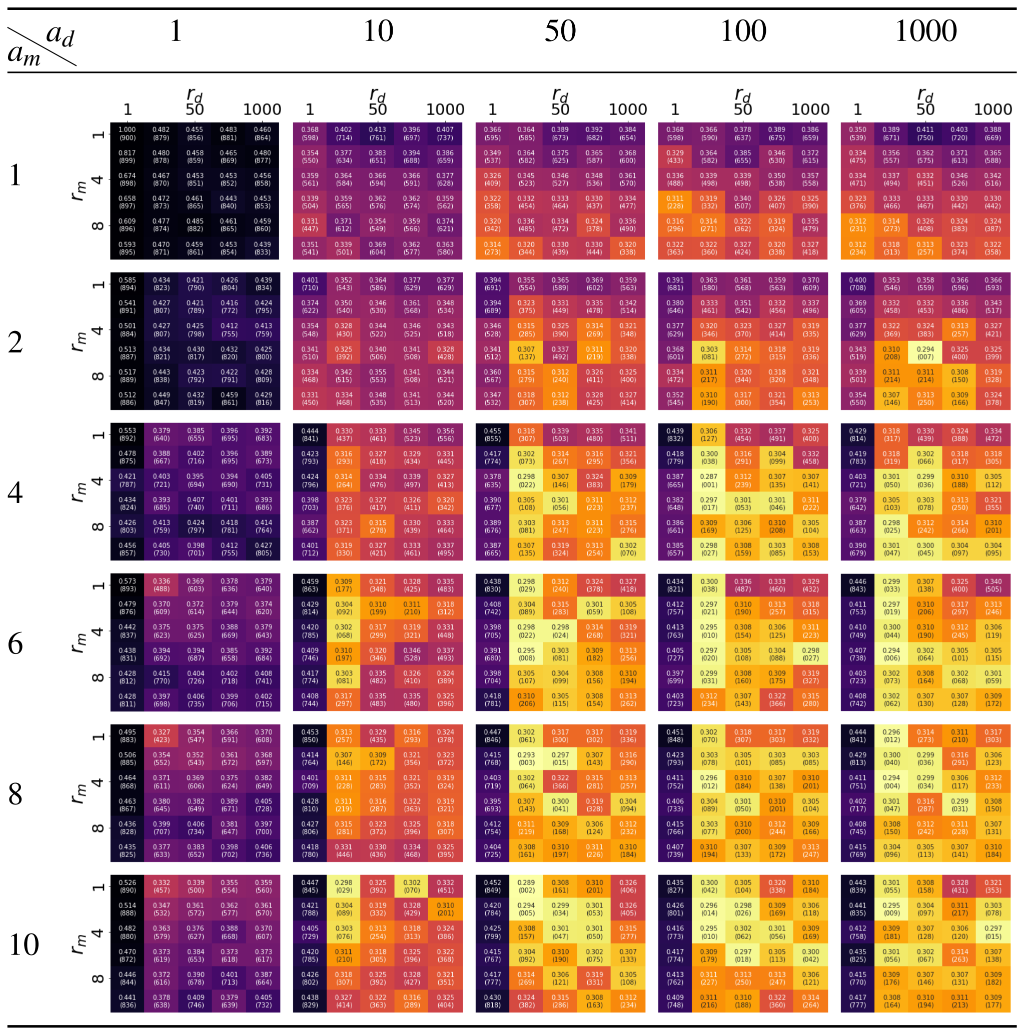


Fig. A.12 Effect of chemotaxis parameters on foraging performance of swarm in the Half50m targets distribution. Best combination for this distribution is $a_m = 4$, $a_d = 100$, $r_m = 4$, $r_d = 10$.

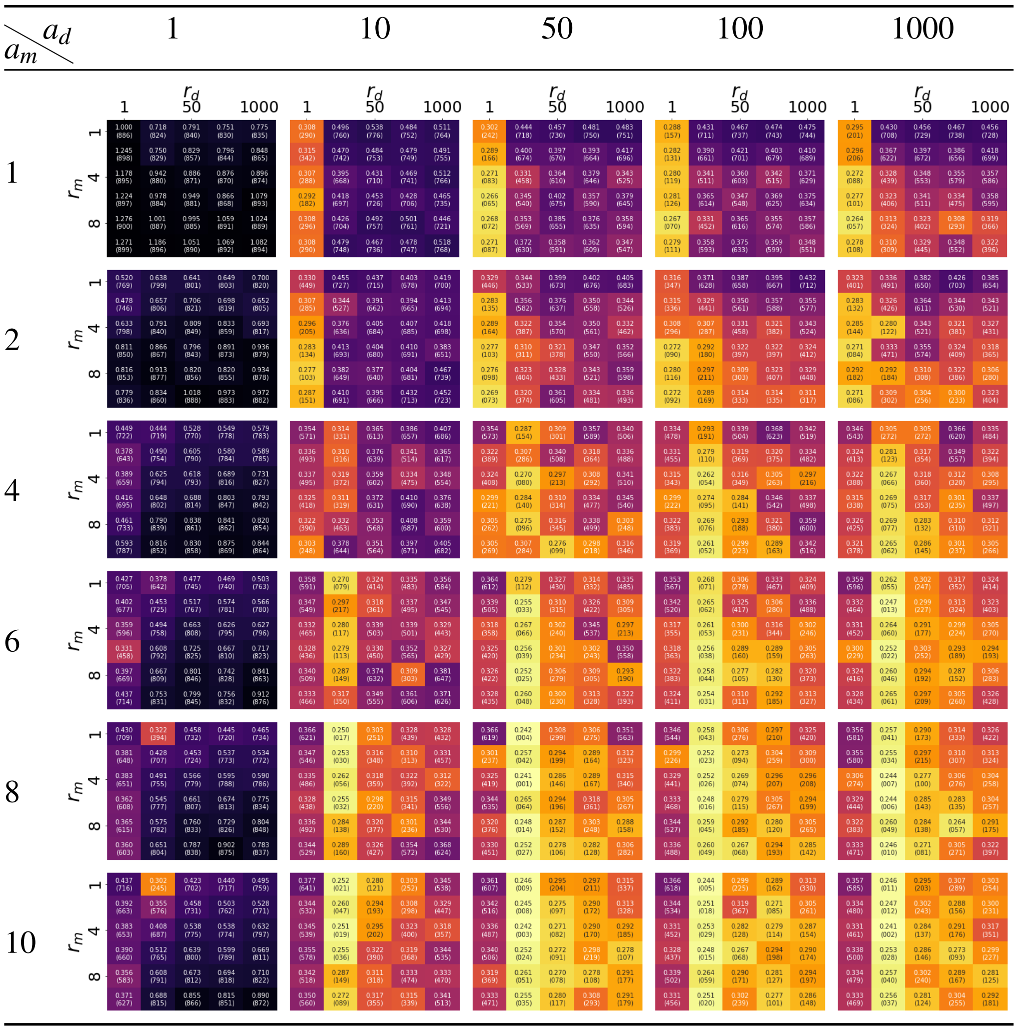


Fig. A.13 Effect of chemotaxis parameters on foraging performance of swarm in the Half100m targets distribution. Best combination for this distribution is $a_m = 8$, $a_d = 50$, $r_m = 4$, $r_d = 10$.

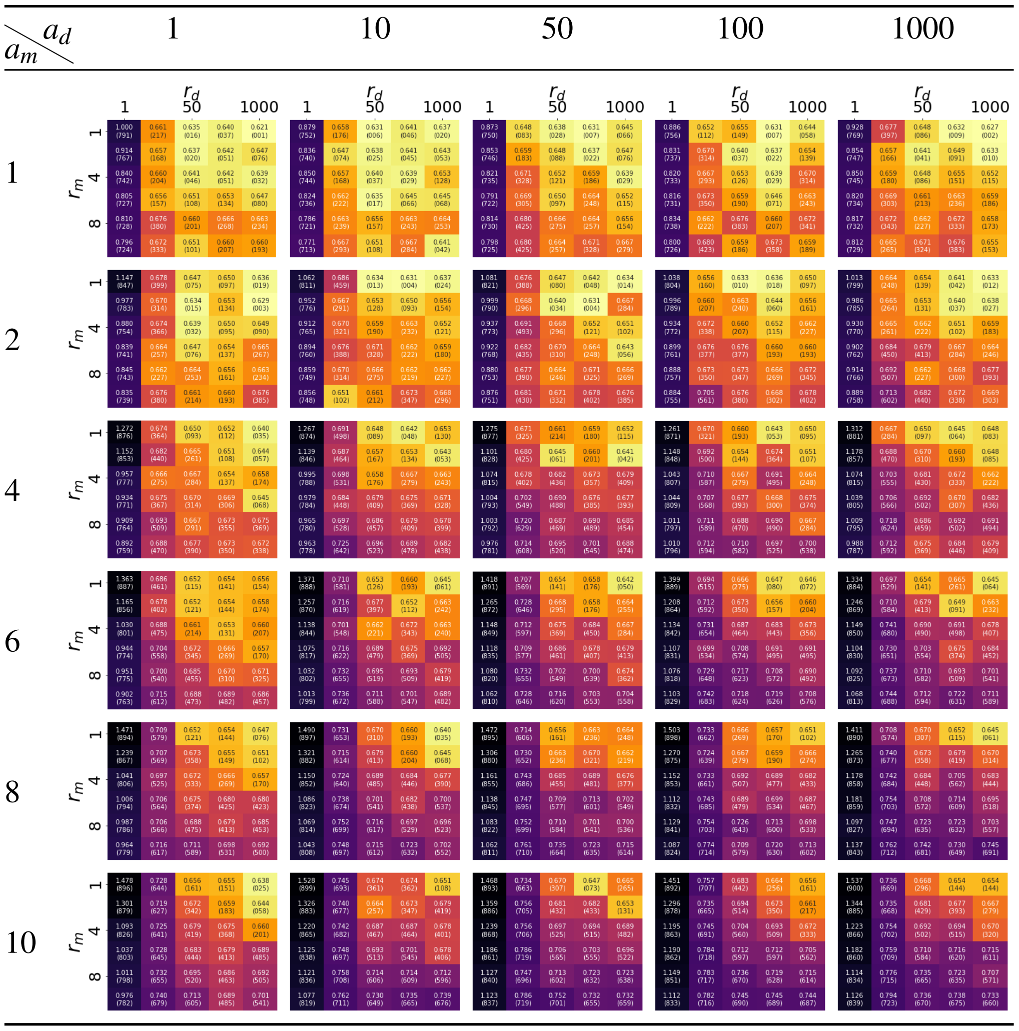


Fig. A.14 Effect of chemotaxis parameters on foraging performance of swarm in the Uniform50m targets distribution. Best combination for this distribution is $a_m = 1$, $a_d = 1$, $r_m = 1$, $r_d = 1000$.

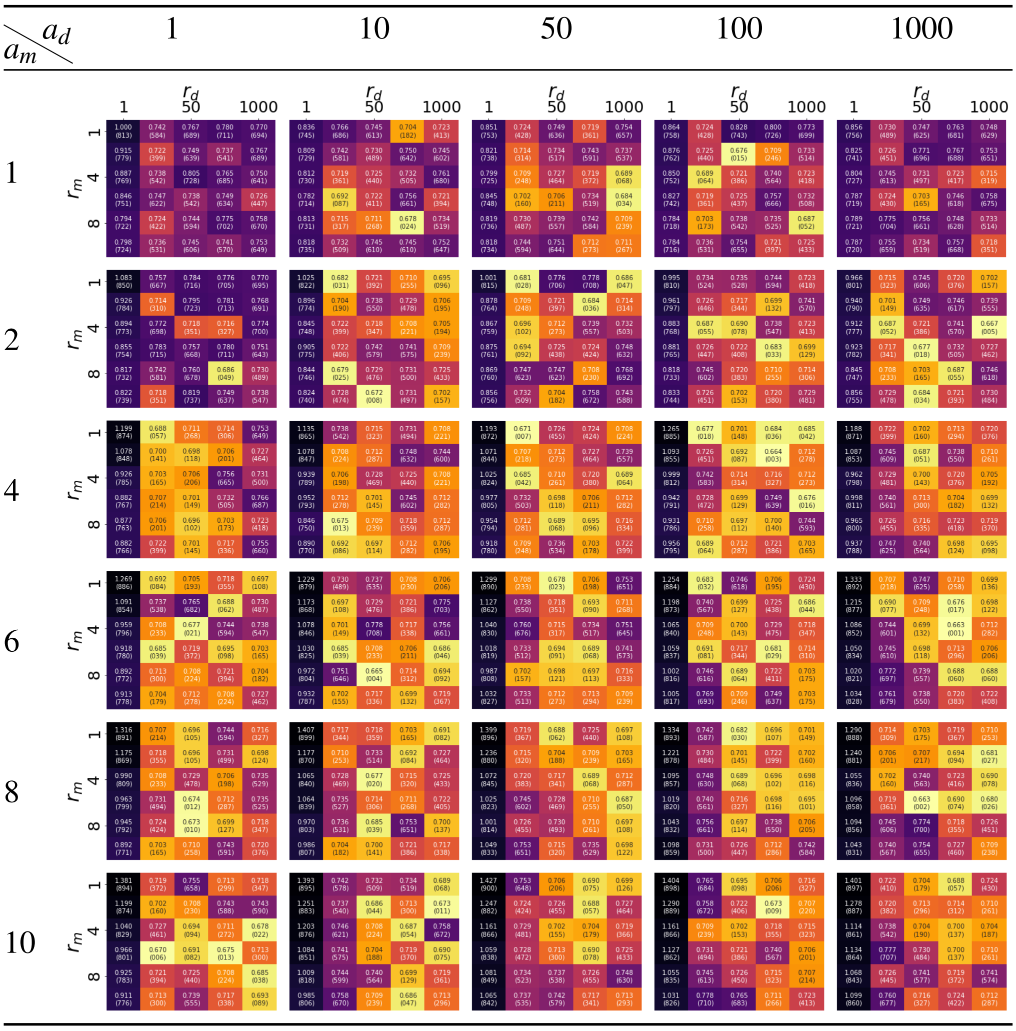


Fig. A.15 Effect of chemotaxis parameters on foraging performance of swarm in the Uniform100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 4$, $r_d = 100$.

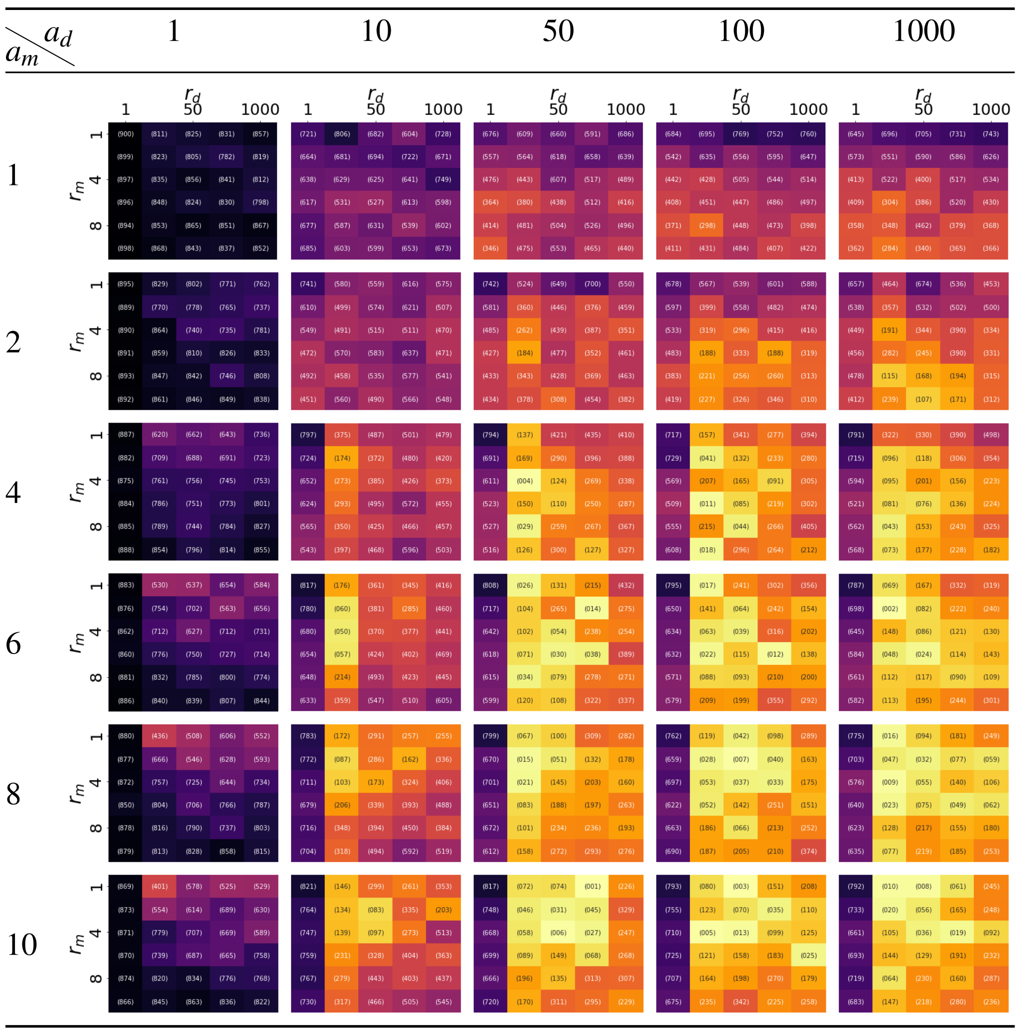
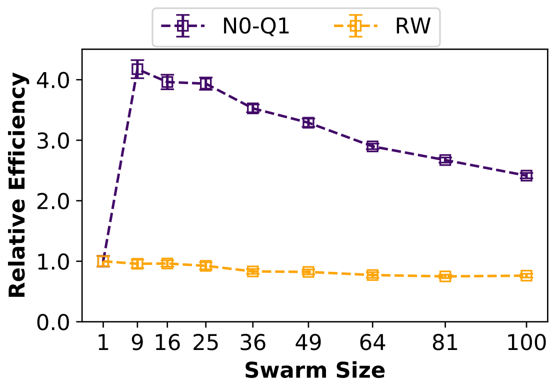
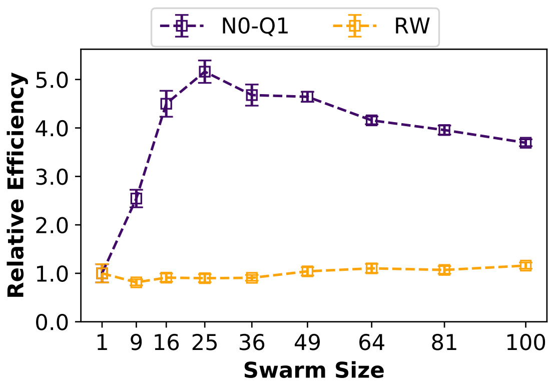


Fig. A.16 Effect of chemotaxis parameters on foraging performance of swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 1$, $r_d = 100$.

A.2 Scalability of RepAtt

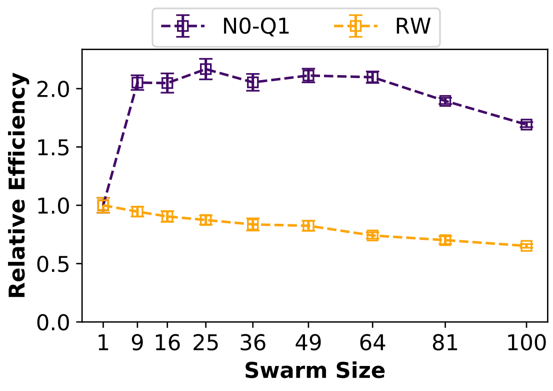


(a) One50m

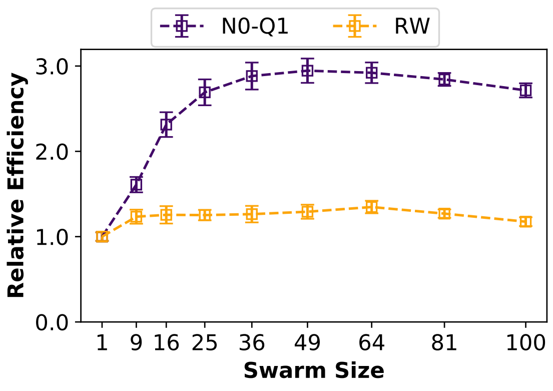


(b) One100m

Fig. A.17 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.

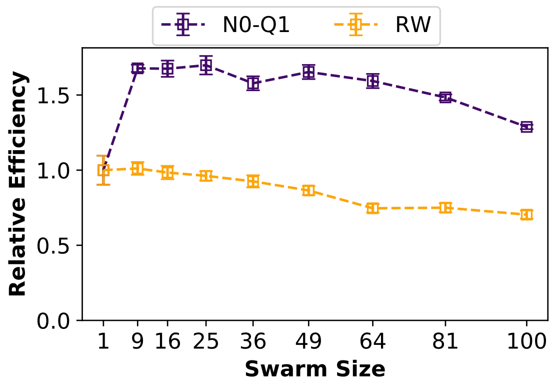


(a) Two50m

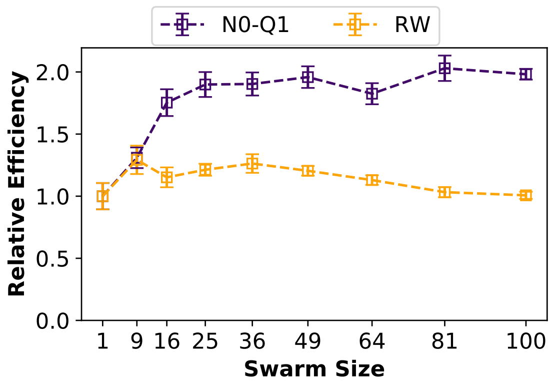


(b) Two100m

Fig. A.18 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.

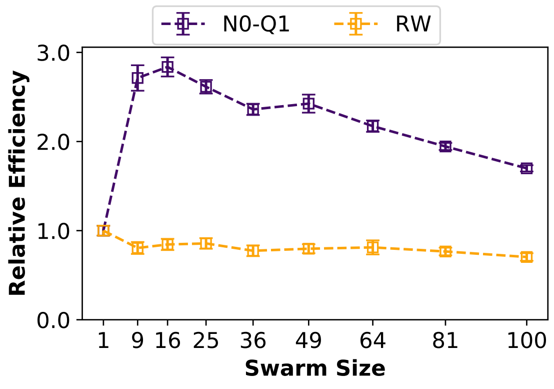


(a) Four50m

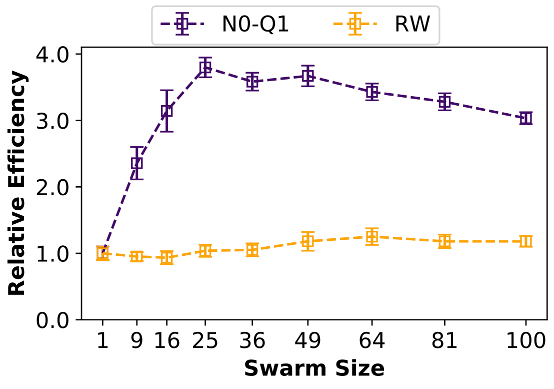


(b) Four100m

Fig. A.19 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.

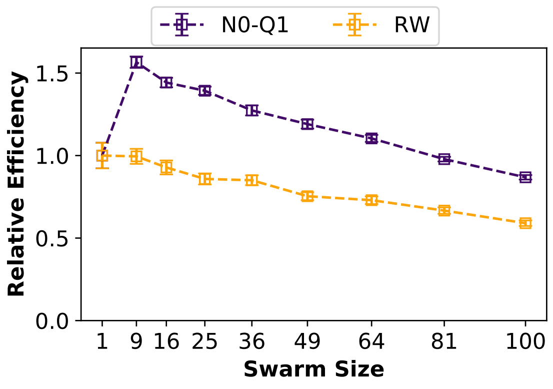


(a) Half50m

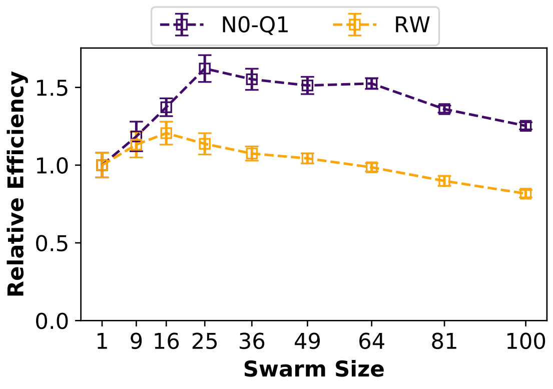


(b) Half100m

Fig. A.20 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.



(a) Uniform50m



(b) Uniform100m

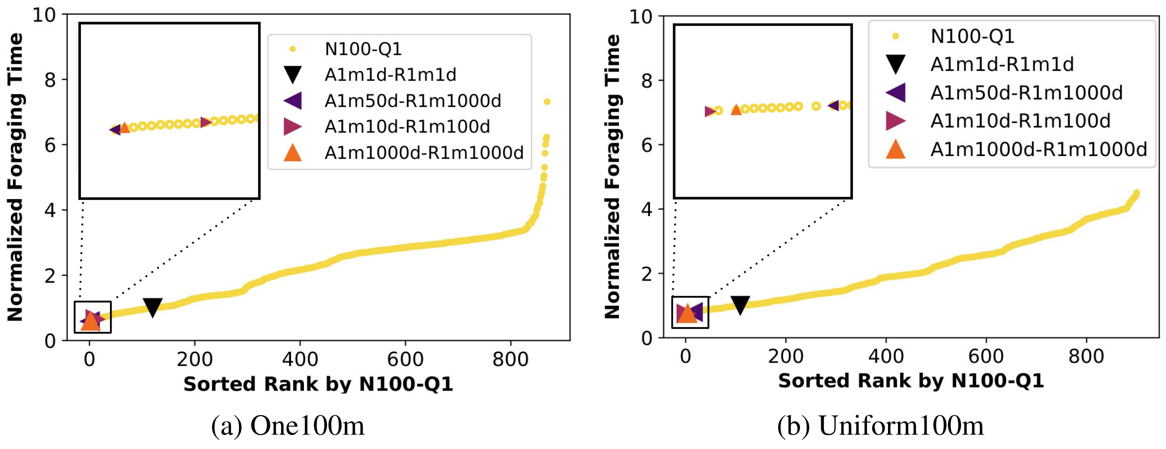
Fig. A.21 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.

Appendix B: Noisy Communication

This appendix contains all results for parameters optimization and scalability simulations in Chapter 4. All the results were not included in the body of the thesis for the sake of readability. All results presented here are based on 30 repetitions of each simulation.

B.1 Parameters Optimisation (N100-Q1)

Task is for swarm of 36 robots to forage 180 out of 200 targets in the specified environments. The chemotaxis parameters optimization involves a parameter sweep experiment where a_m and r_m were selected from 1, 2, 4, 6, 8 and 10, while a_d and r_d were selected from 1, 10, 50, 100, 1000. This resulted in 900 different combinations of the chemotaxis parameters. All the foraging time results were normalised using Random Walk foraging time.



(a) One100m (b) Uniform100m

Fig. B.1 Sorted foraging times normalized using the mean time for Random Walk for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination. The robots foraged using noisy communication without using any noise filtering system (i.e. N100-Q1)

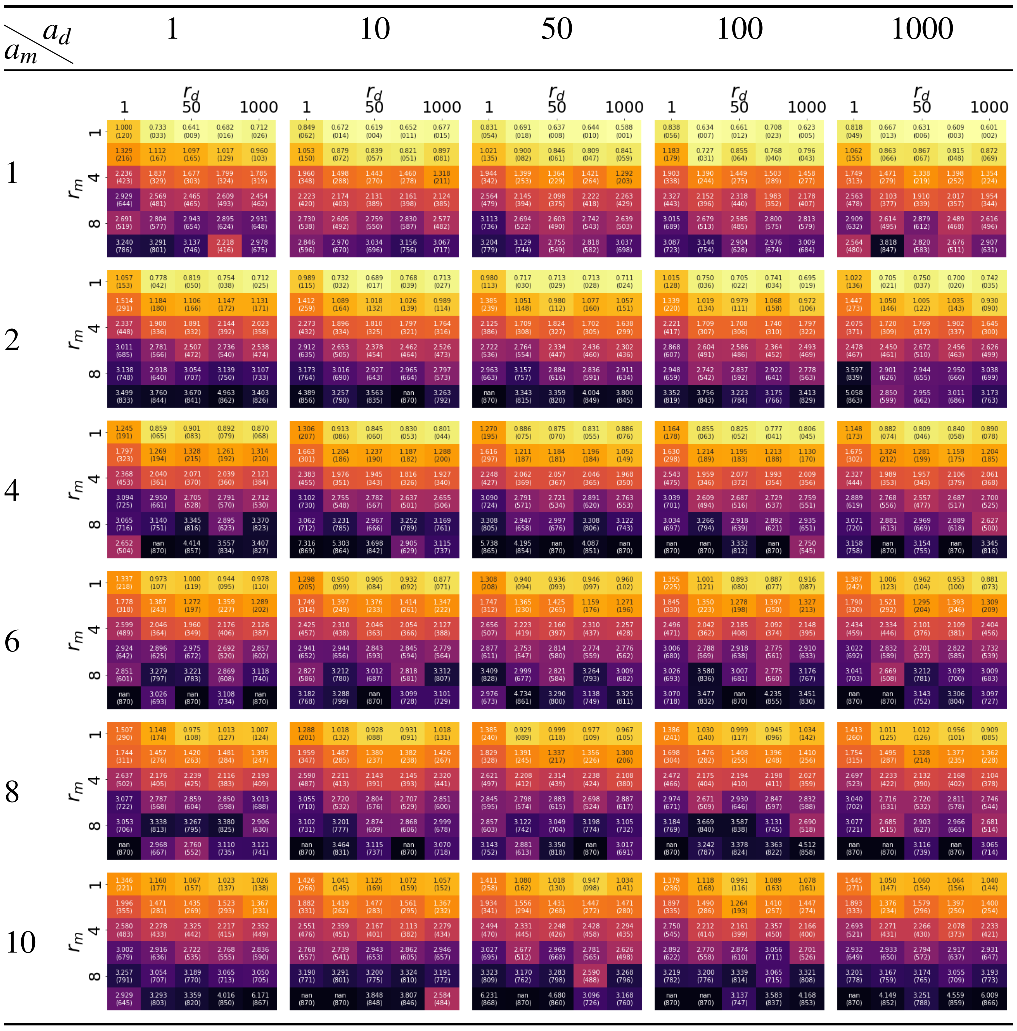


Fig. B.2 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution and using N100-Q1 version of RepAtt. Best combination from the scoring system is $a_m = 1$, $a_d = 1000$, $r_m = 1$, $r_d = 1000$.

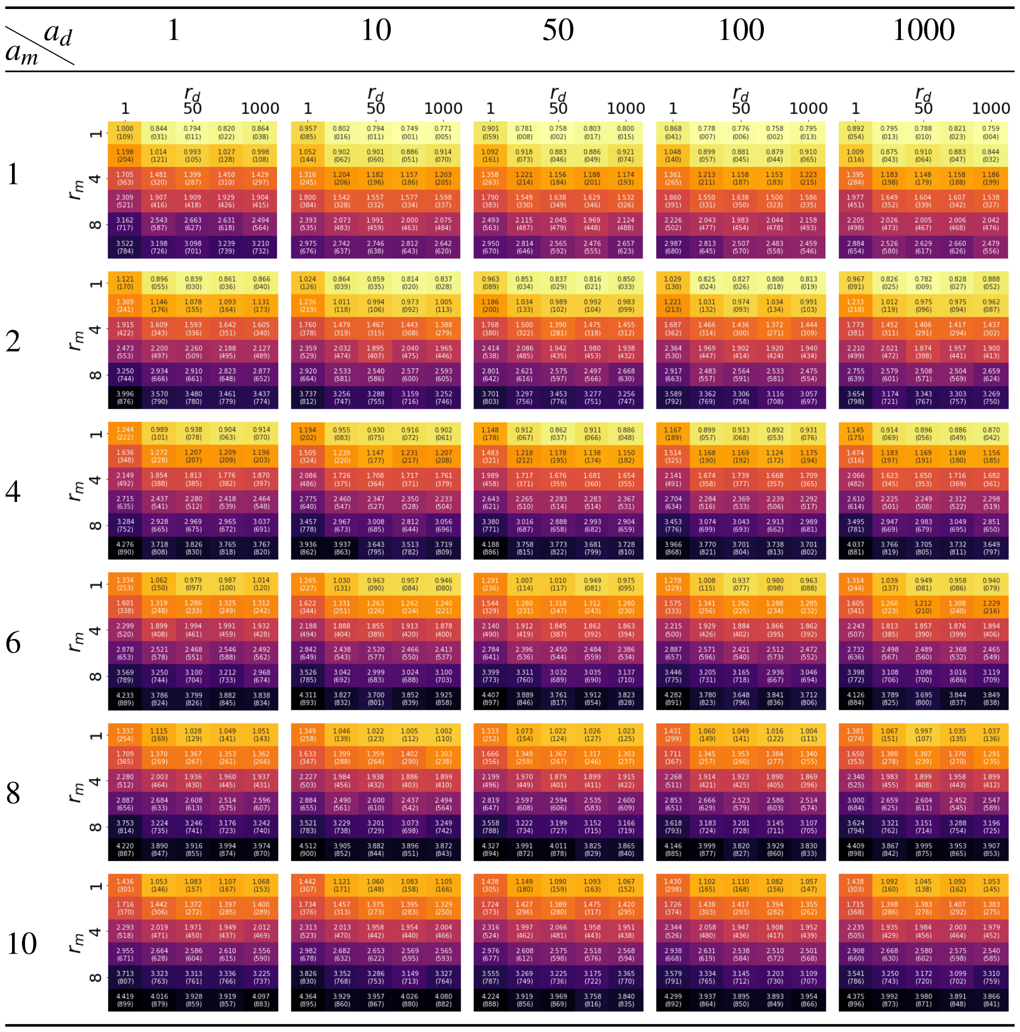


Fig. B.3 Effect of the chemotaxis parameters on the foraging performance of a swarm in the Uniform100m targets distribution and using N100-Q1 version of RepAtt. Best combination from the scoring system is $a_m = 1$, $a_d = 1000$, $r_m = 1$, $r_d = 1000$.

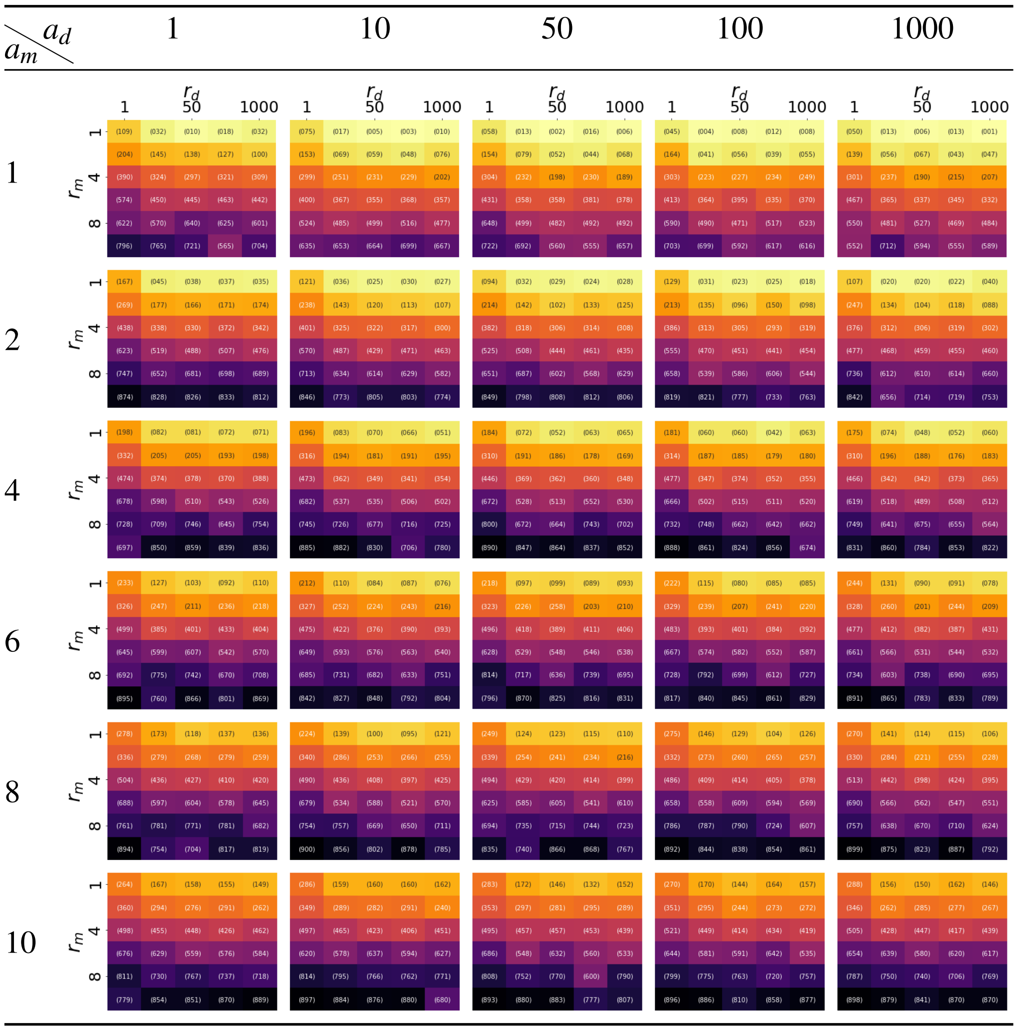
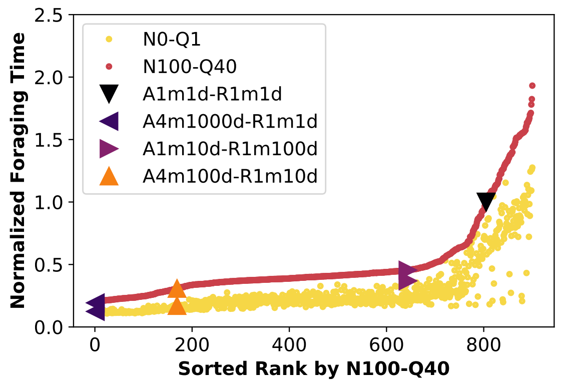
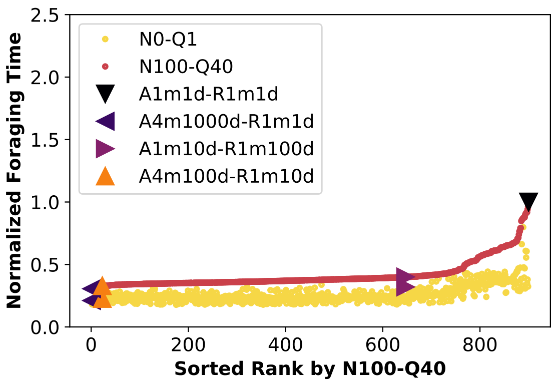


Fig. B.4 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m and Uniform100m targets distributions and using N100-Q1 version of RepAtt. Best combination from the scoring system is $a_m = 1$, $a_d = 1000$, $r_m = 1$, $r_d = 1000$.

B.2 Parameters Optimisation (N100-Q40)

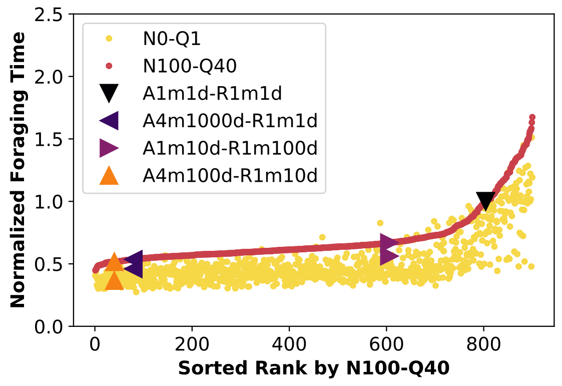


(a) One100m

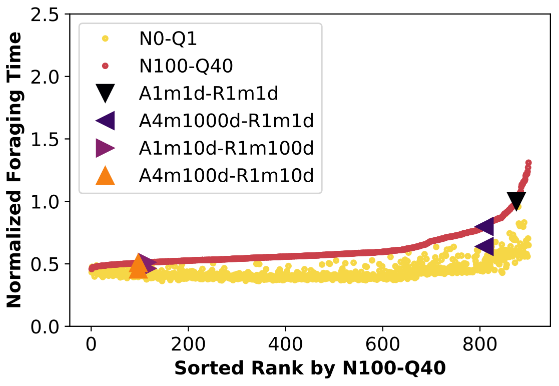


(b) One50m

Fig. B.5 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.

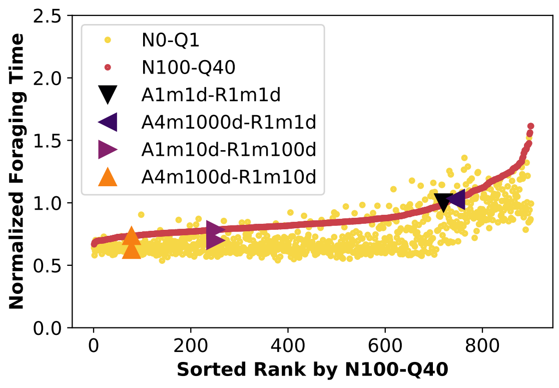


(a) Two100m

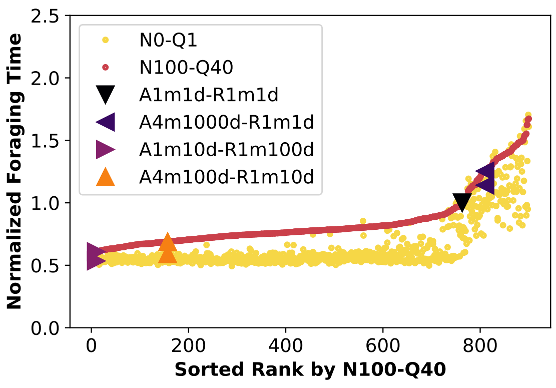


(b) Two50m

Fig. B.6 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.

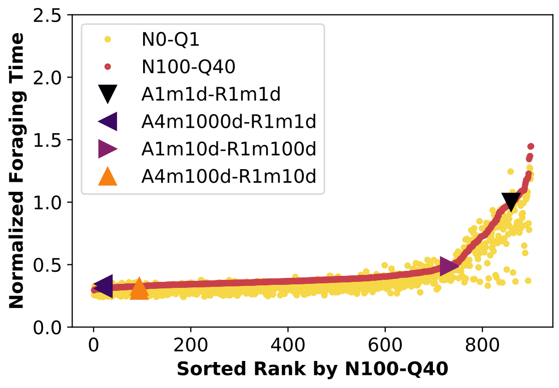


(a) Four100m

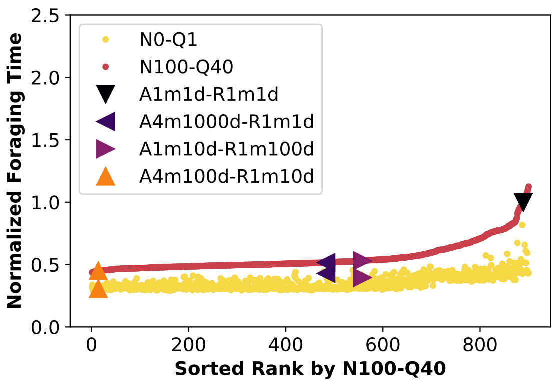


(b) Four50m

Fig. B.7 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.

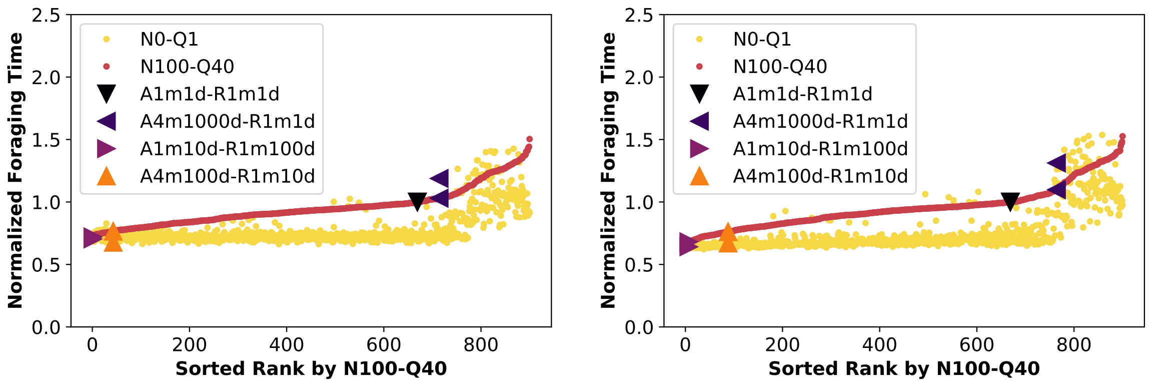


(a) Half100m



(b) Half50m

Fig. B.8 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.



(a) Uniform100m (b) Uniform50m

Fig. B.9 Sorted foraging times normalized using the mean time for A1m1d-R1m1d for the corresponding world setup. Swarm size of 36 robots and 900 different combinations of chemotaxis parameters were used. Each marker represents the mean of 30 simulation repetitions of each chemotaxis parameters combination.

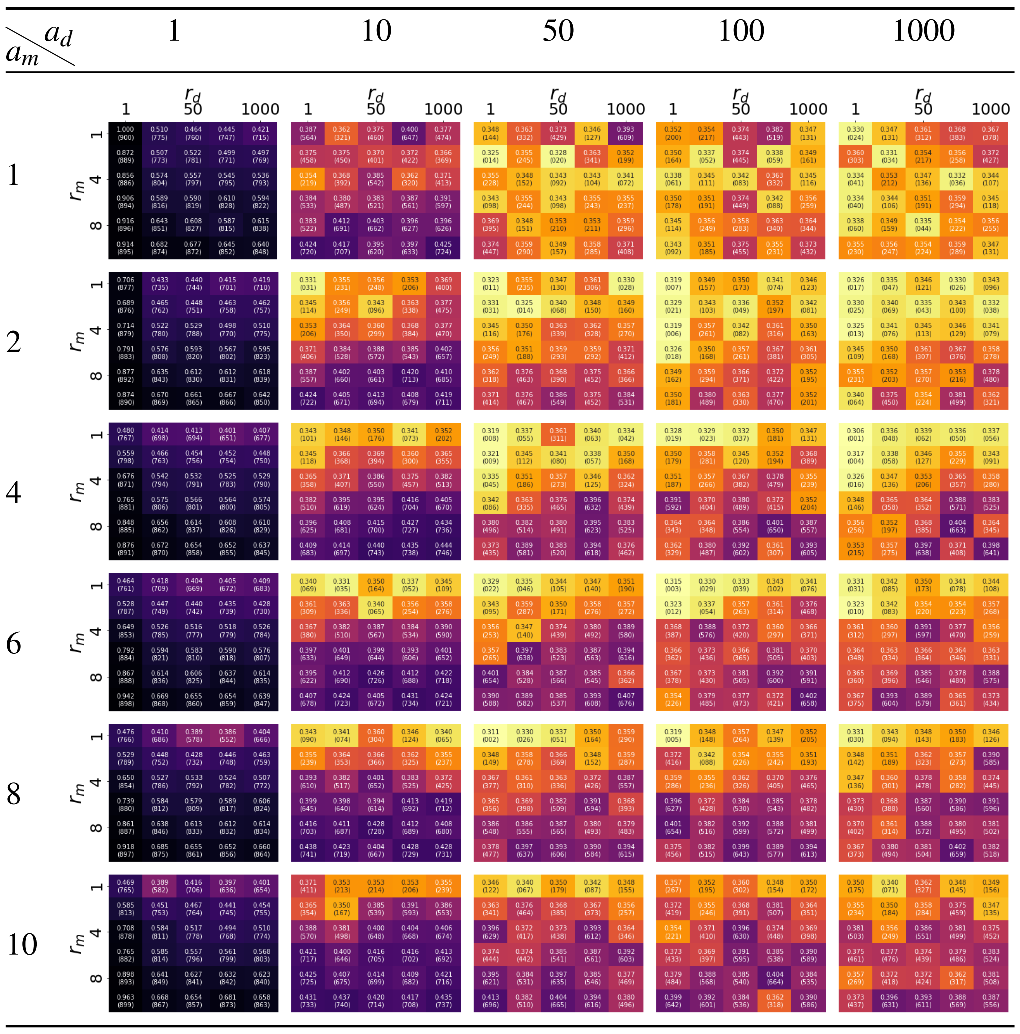


Fig. B.10 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 1000$, $r_m = 1$, $r_d = 1$.

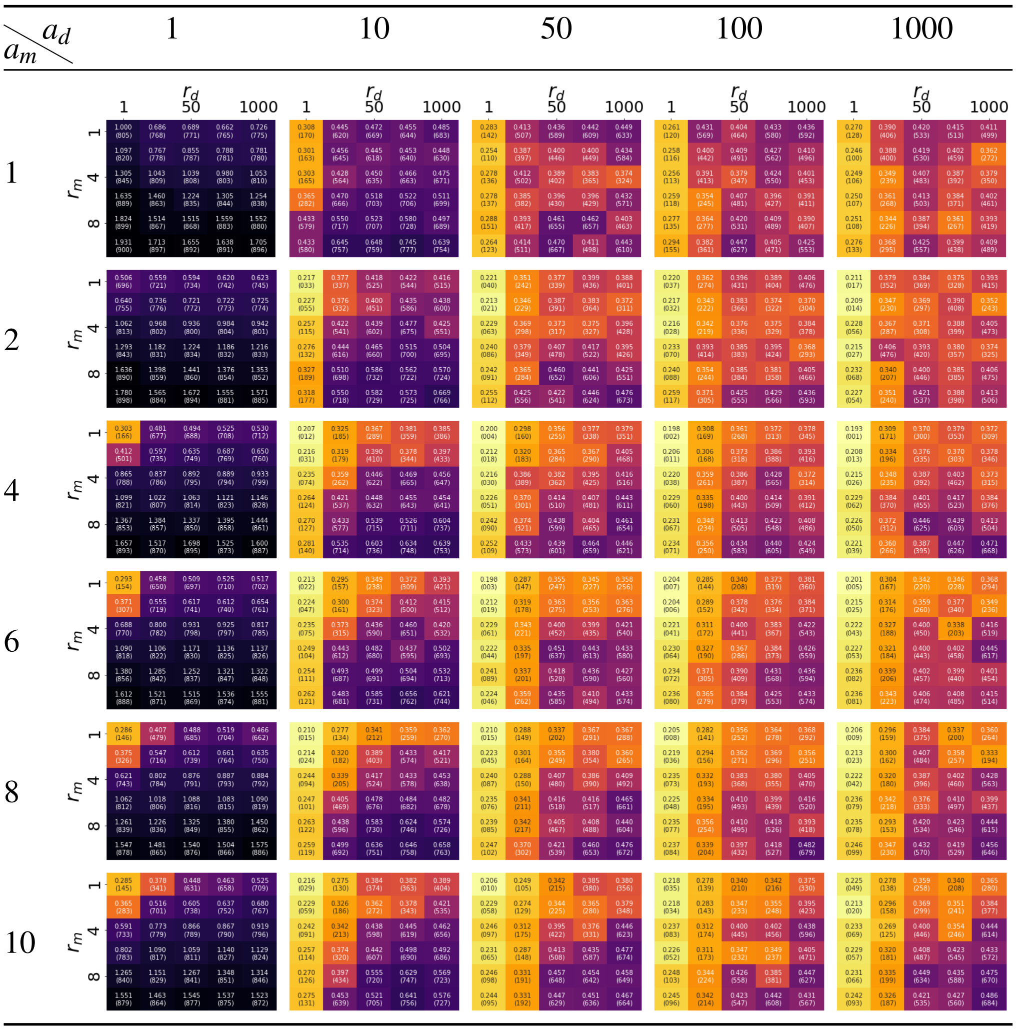


Fig. B.11 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 2$, $r_d = 1$.

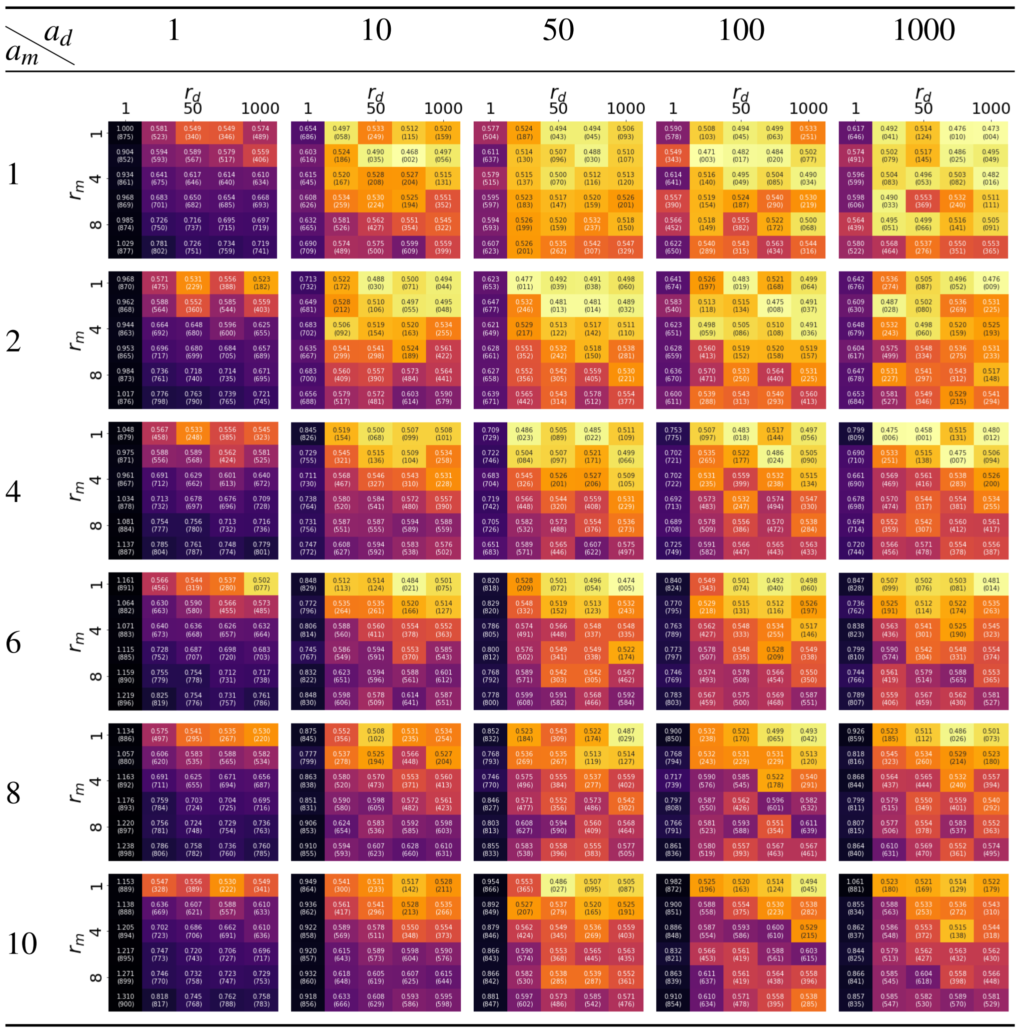


Fig. B.12 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 4$, $r_d = 50$.

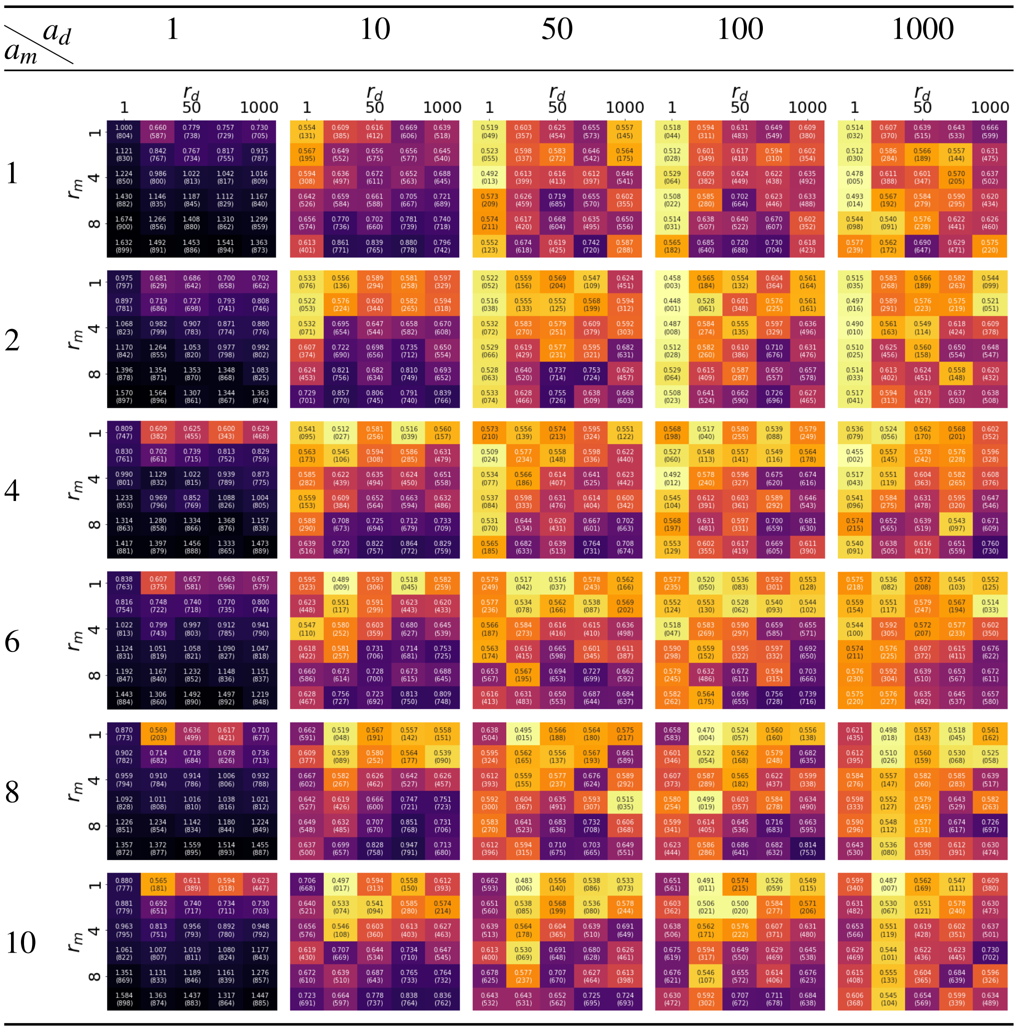


Fig. B.13 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 2$, $r_d = 10$.

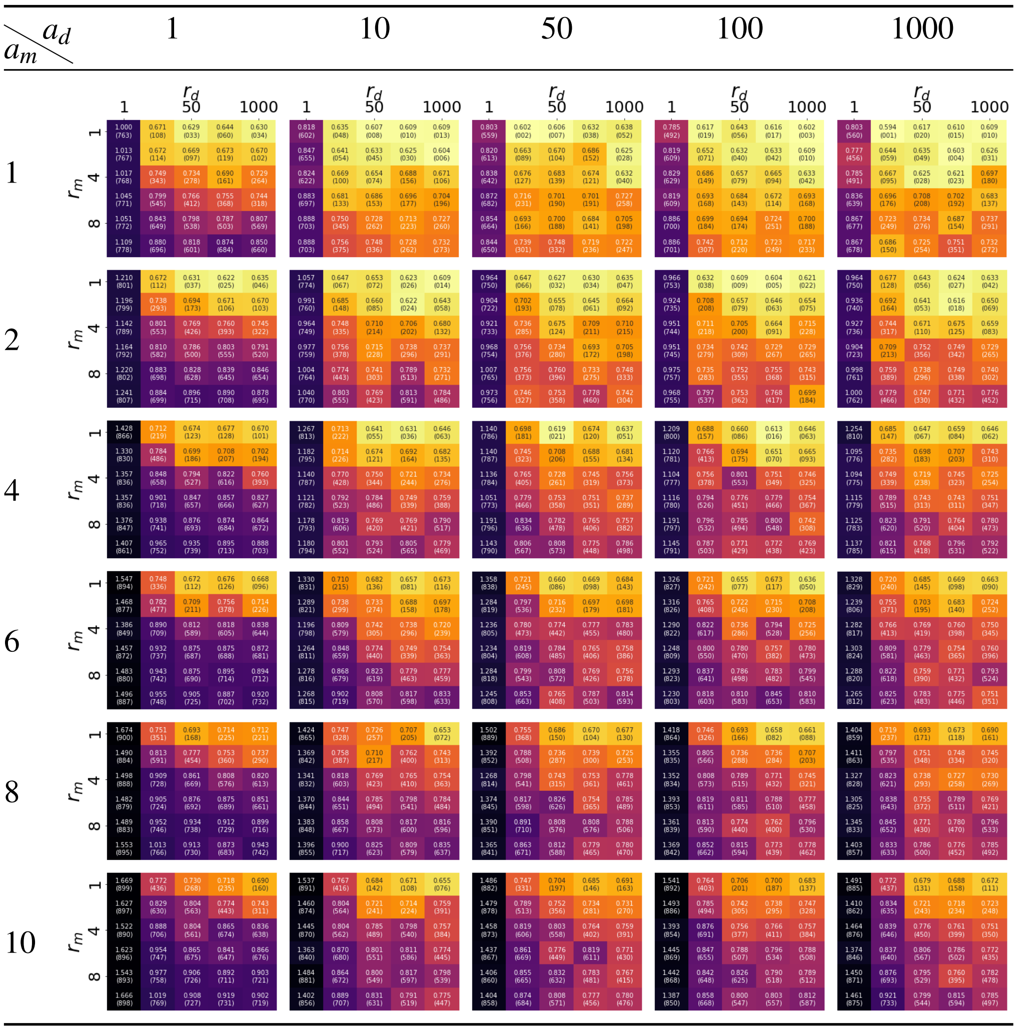


Fig. B.14 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 4$, $a_d = 50$, $r_m = 6$, $r_d = 1000$.

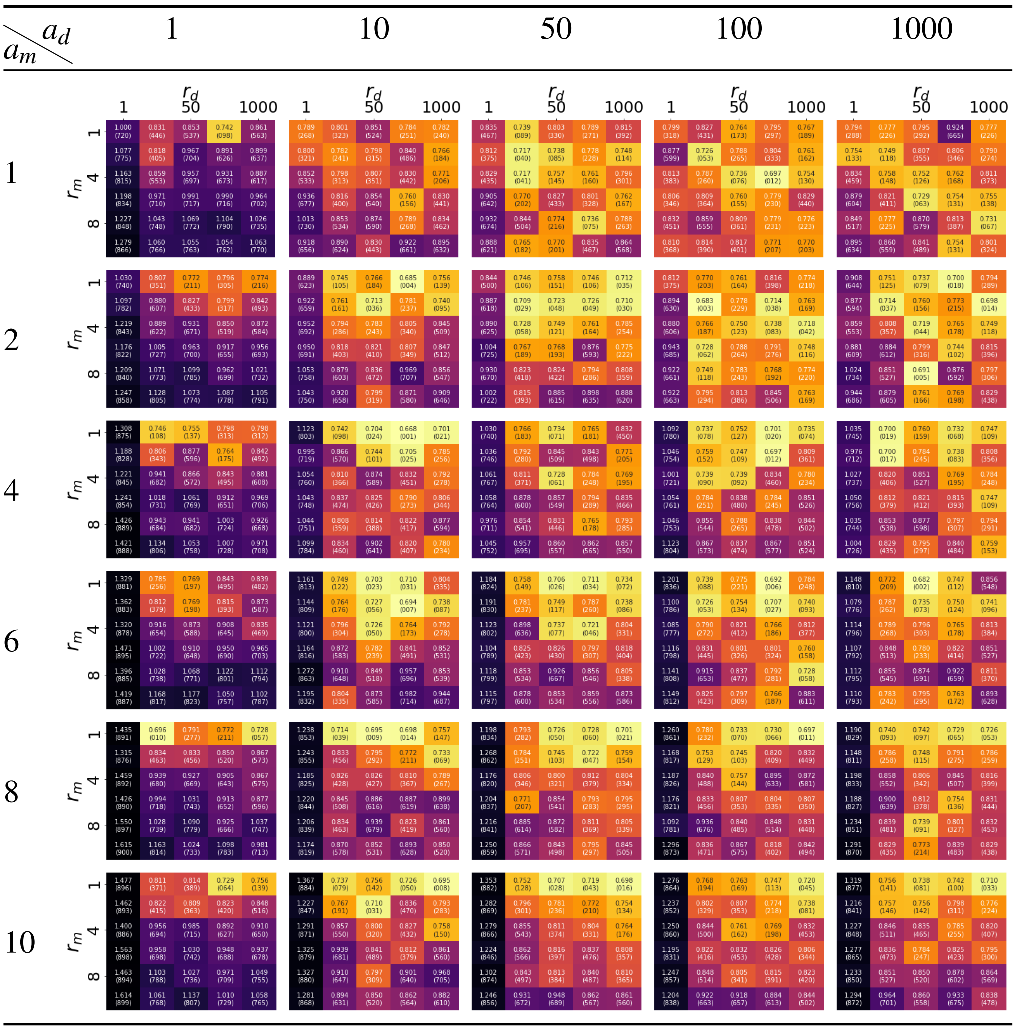


Fig. B.15 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 1000$, $r_m = 1$, $r_d = 10$.

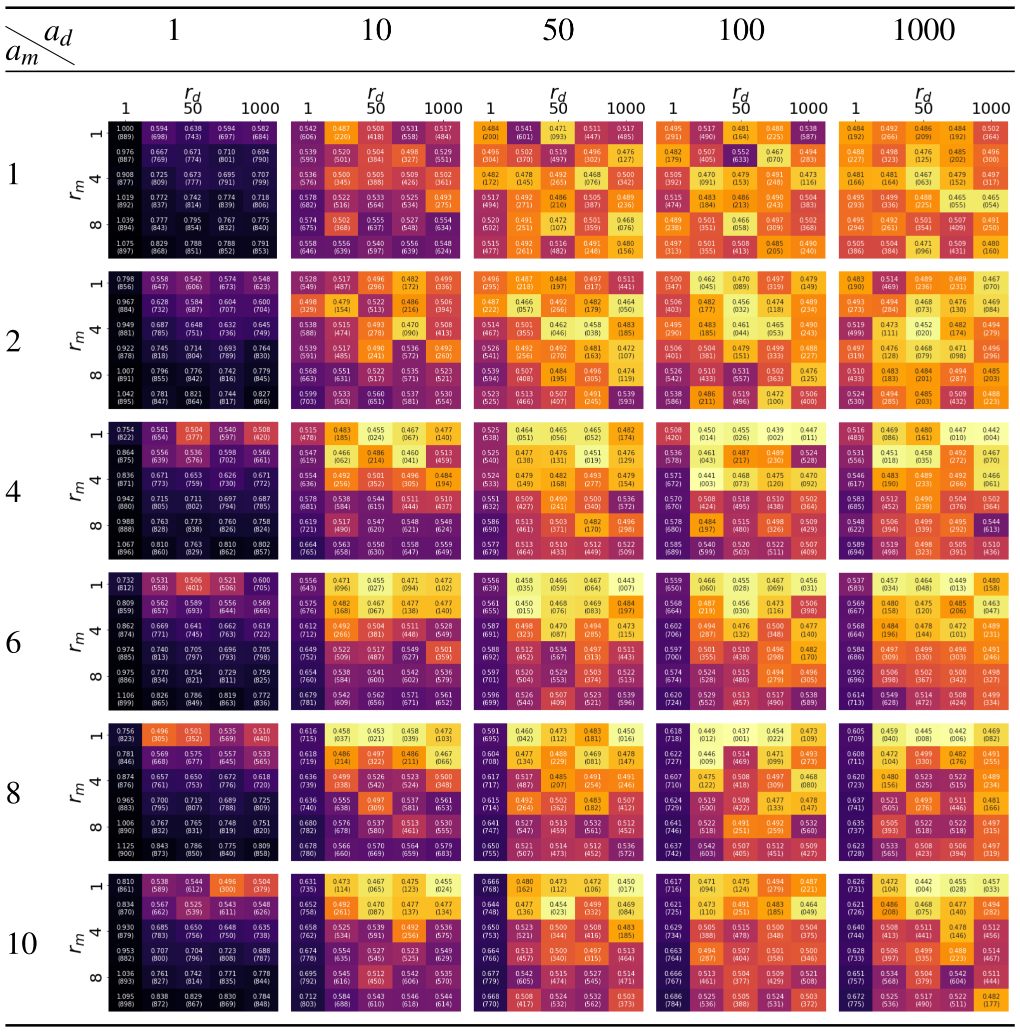


Fig. B.16 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 4$, $a_d = 100$, $r_m = 4$, $r_d = 10$.

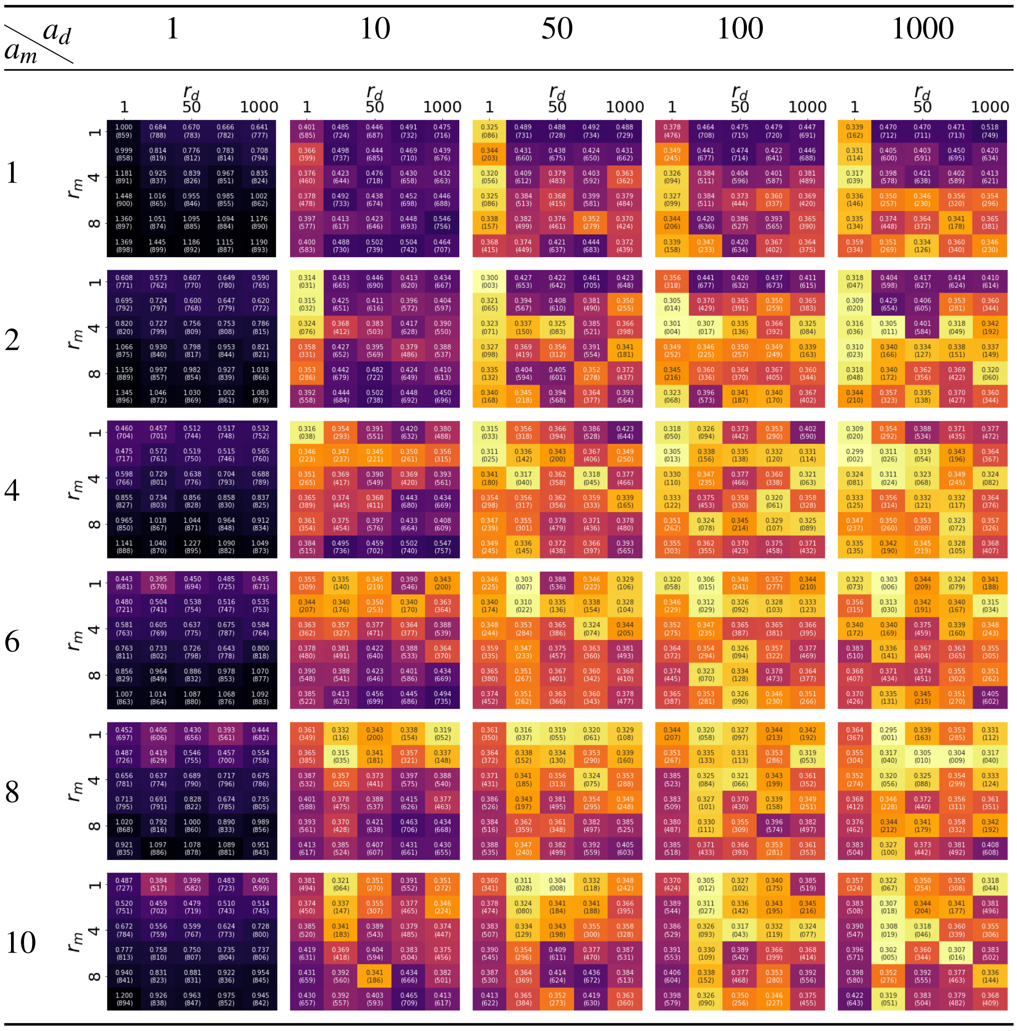


Fig. B.17 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 8$, $a_d = 50$, $r_m = 4$, $r_d = 10$.

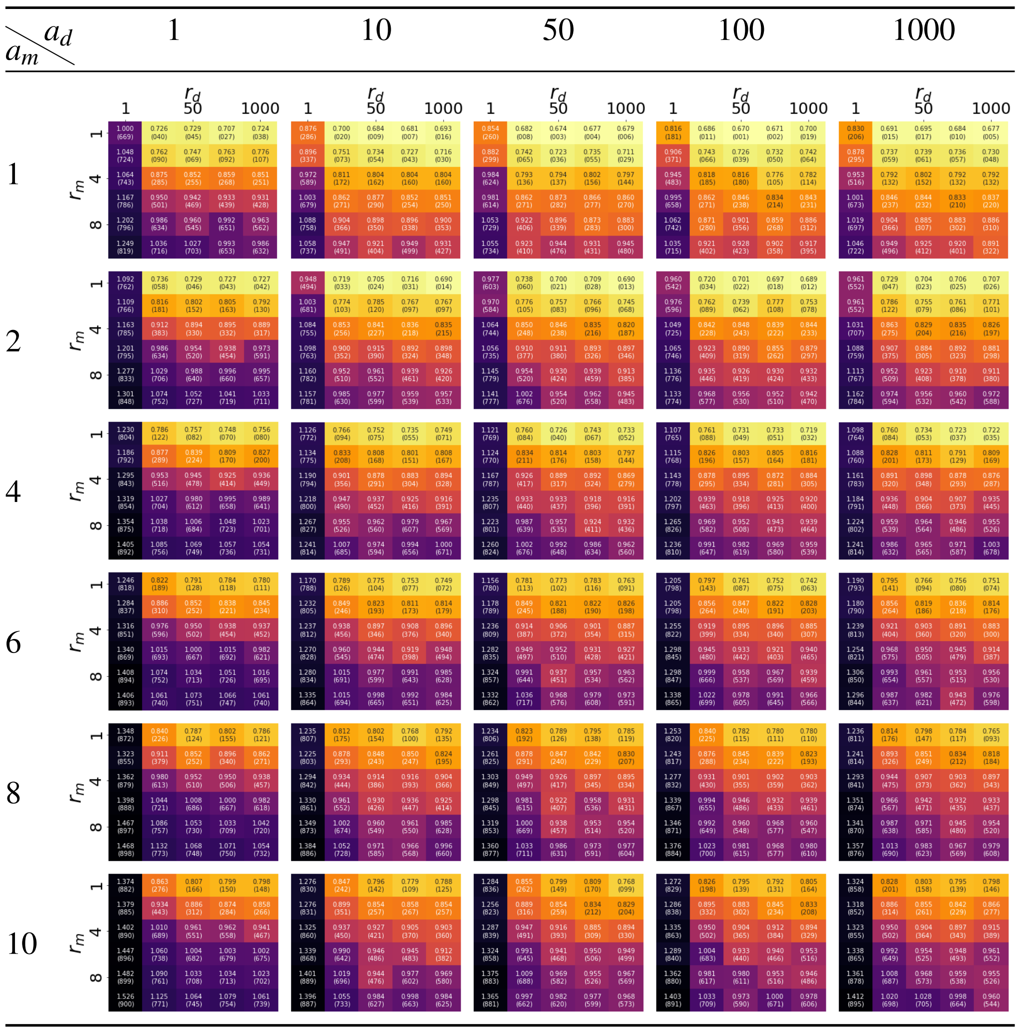


Fig. B.18 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 1$, $a_d = 1$, $r_m = 1$, $r_d = 1000$.

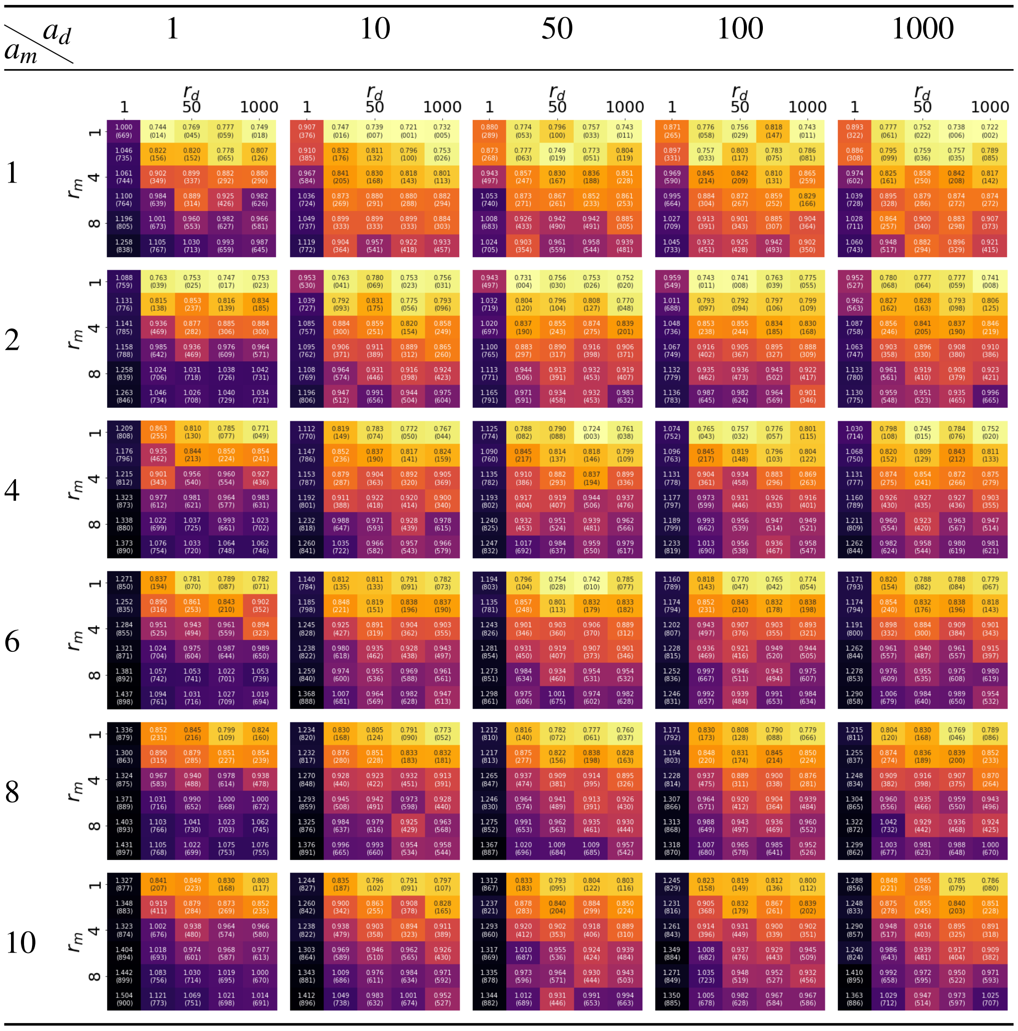


Fig. B.19 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 6$, $a_d = 1000$, $r_m = 4$, $r_d = 100$.

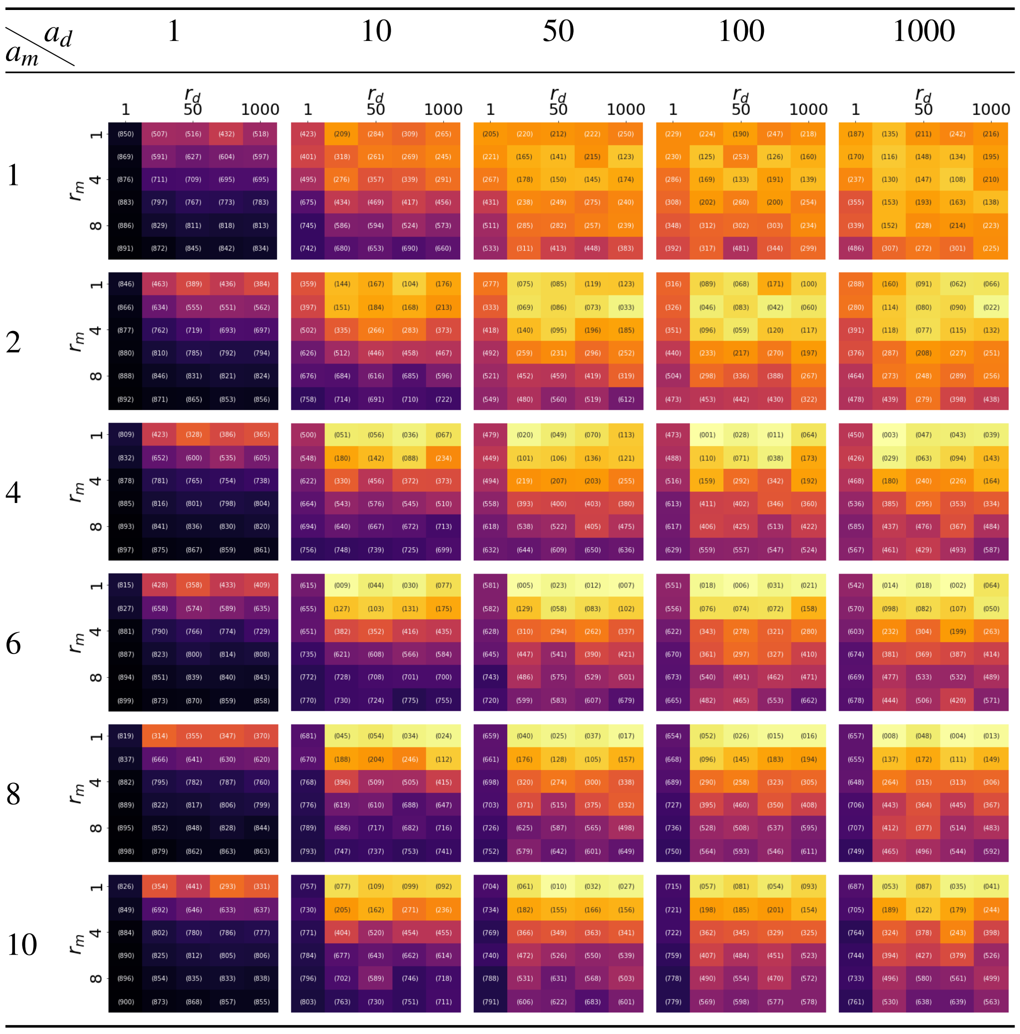
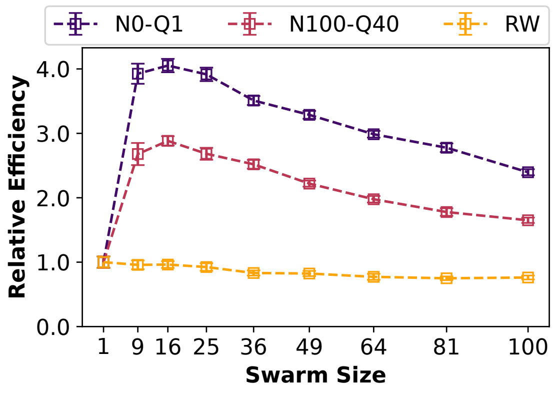
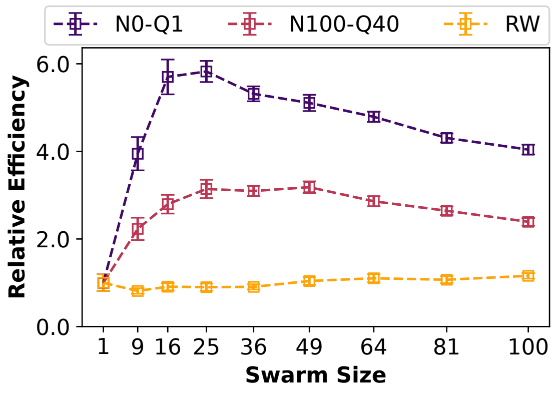


Fig. B.20 Effect of the chemotaxis parameters on the foraging performance of a swarm in the One100m targets distribution. Best combination for this distribution is $a_m = 10$, $a_d = 50$, $r_m = 1$, $r_d = 100$.

B.3 Scalability of RepAtt

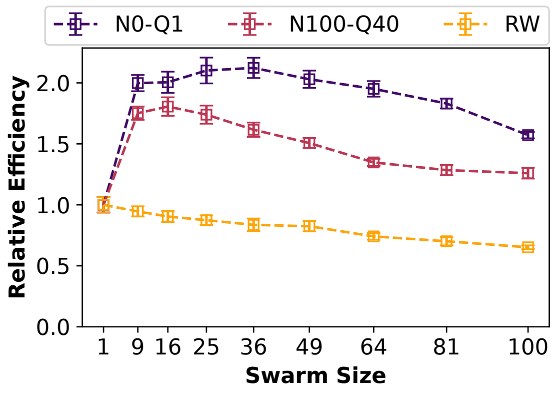


(a) One50m

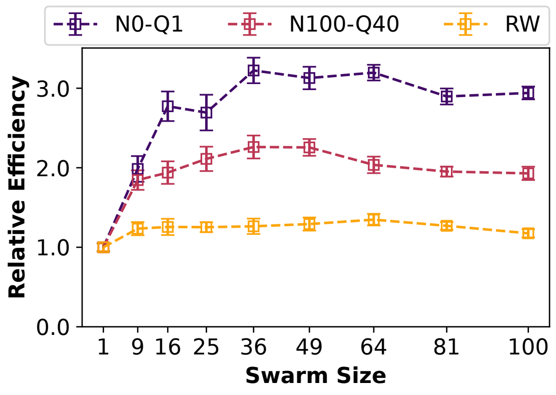


(b) One100m

Fig. B.21 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.

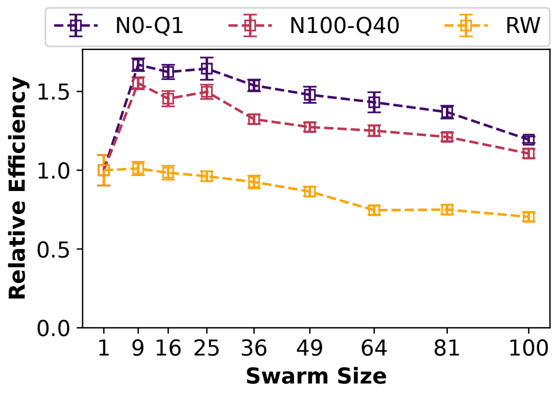


(a) Two50m

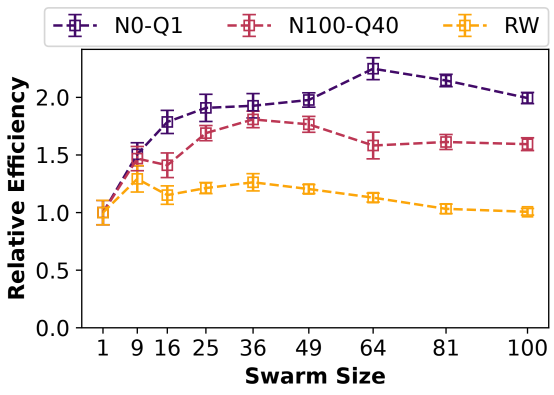


(b) Two100m

Fig. B.22 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.

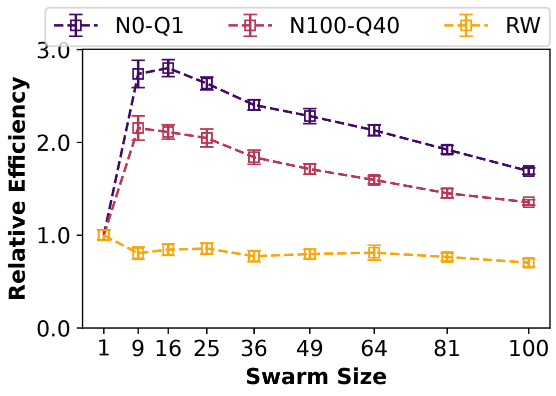


(a) Four50m

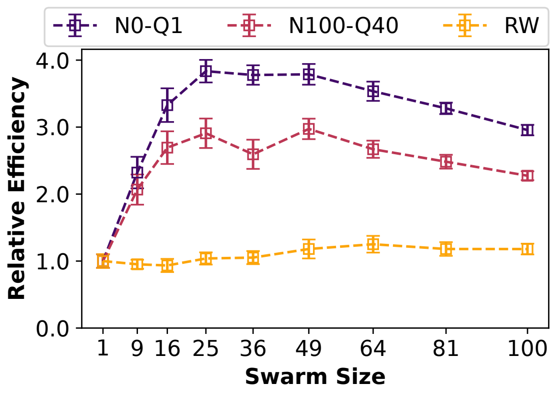


(b) Four100m

Fig. B.23 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.

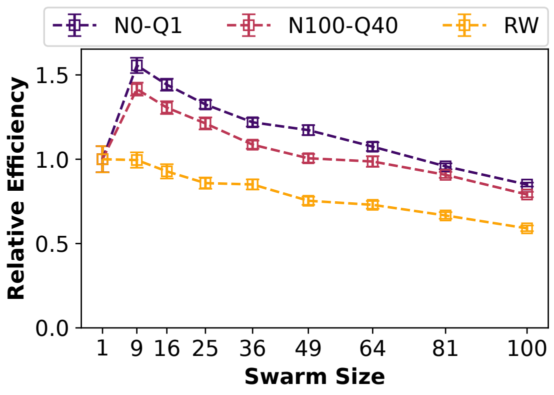


(a) Half50m

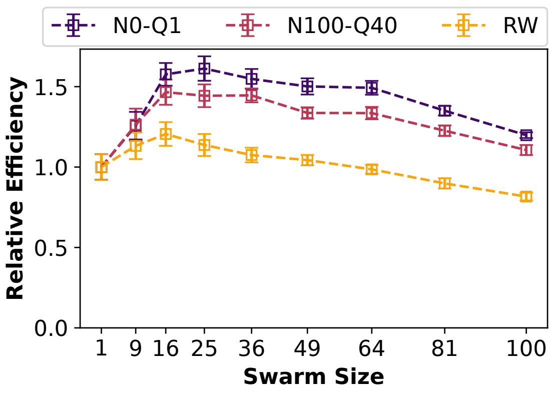


(b) Half100m

Fig. B.24 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.



(a) Uniform50m



(b) Uniform100m

Fig. B.25 Relative efficiency computed based on $p = 180$ out of 200 total targets. Each simulation was repeated 30 times and error bars represent 95% confidence interval. RepAtt parameters of $a_m = 10$, $a_d = 50$, $r_m = 1$ and $r_d = 100$ were used.

Appendix C: Imperfect Vision

This appendix contains extra results for the vision modelling process in Chapter 5. All the results were not included in the body of the thesis for the sake of readability.

C.1 False Positive Data

Table C.1 Analysis of the MobileNet-SSD and tiny-YOLOv3 performances and metrics computation by analysing all frames of the test dataset and extracting data based on the false positive detections on the test dataset. These were not used in the vision model presented in the thesis in order to simplify the model and focus on scenarios where robots vision system fails to detect targets in its immediate surrounding.

	MobileNet-SSD		Tiny-YOLOv3	
	124×124	220×220	128×128	224×224
seen	17287	22545	361	1003
unseen	276393	300077	3830	16101
seen2seen	9087	11785	288	629
seen2unseen	8198	10739	73	374
unseen2seen	5494	6616	34	147
unseen2unseen	268195	289338	3757	15727
P_s	0.0566 ± 0.0193	0.0710 ± 0.0185	0.0794 ± 0.0948	0.1202 ± 0.1631
P_{s2s}	0.4730 ± 0.1019	0.5039 ± 0.0699	0.4066 ± 0.3738	0.3895 ± 0.2959
P_{u2s}	0.0200 ± 0.0057	0.0227 ± 0.0057	0.0128 ± 0.0205	0.0202 ± 0.0545

C.2 Effects of P_{u2s}, P_{s2s} and Vision Update Rate on Swarm Foraging

Varying the P_{s2s} and P_{u2s} between 0.001 – 1 and visualising their effects on swarm of 36 foraging robots tasked with picking up 90% of 200 targets distributed in 10 test environments. Each robot had 120° field of view and a detection distance of 5 metres. For the N0-Q1 and N100-Q40 RepAtt algorithms, chemotaxis parameters used were $a_m = 4$, $a_d = 100$, $r_m = 1$ and $r_d = 10$. In all the figures, the y-axis is the average foraging time, while x-axis is the corresponding P_s value computed from the simulations. Each data point represents mean of 30 simulation repetitions.

C.2.1 Algorithms foraging performance

This section compares Random Walk, N0-Q1 and N100-Q40 for varying vision update rates. The foraging times shown on the y-axis have been normalised using performance of Random Walk algorithm with perfect vision for the corresponding world.

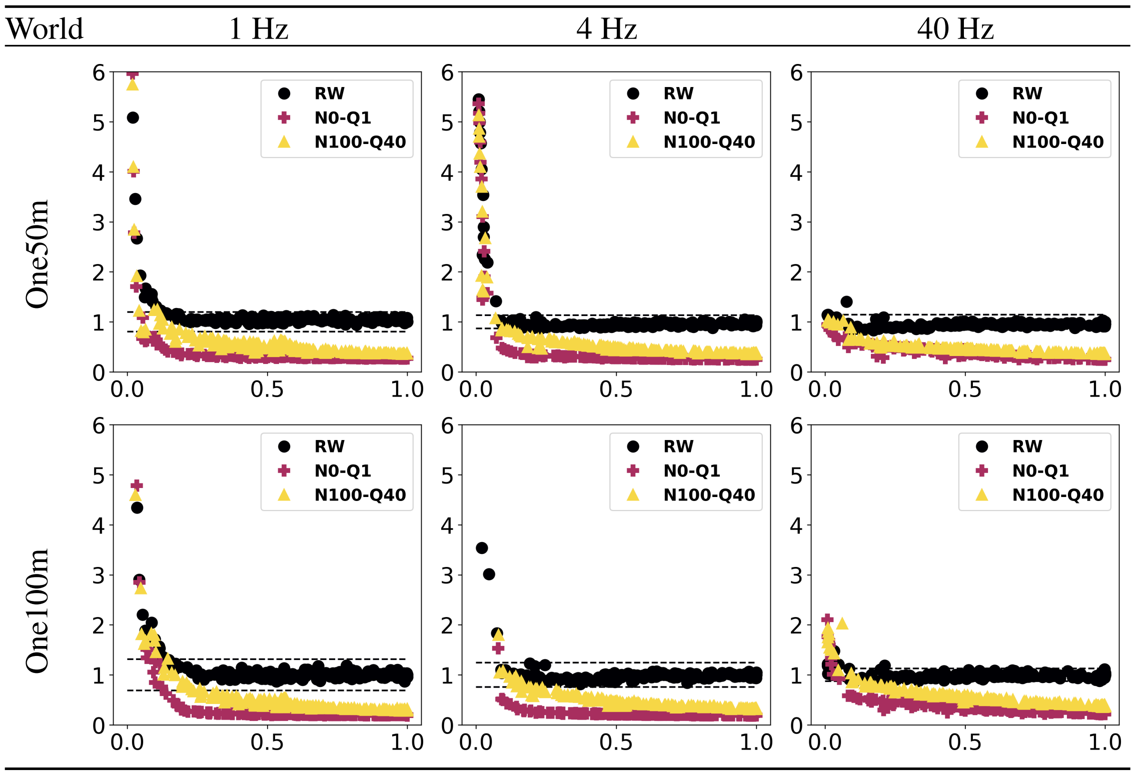


Fig. C.1 One cluster environments

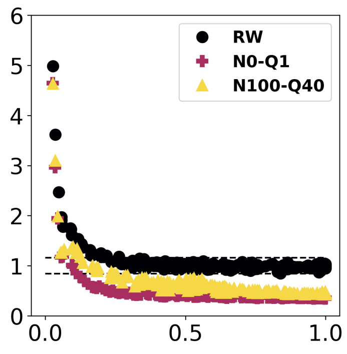
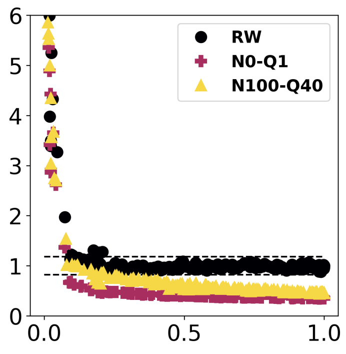
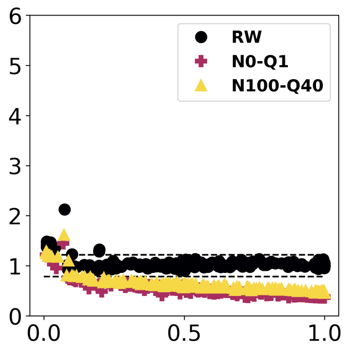
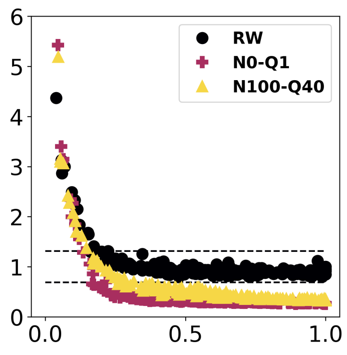
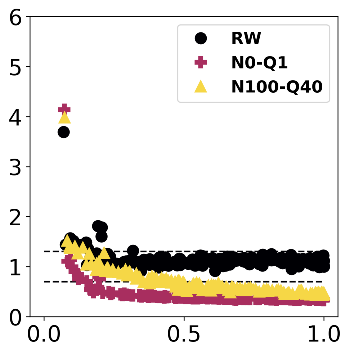
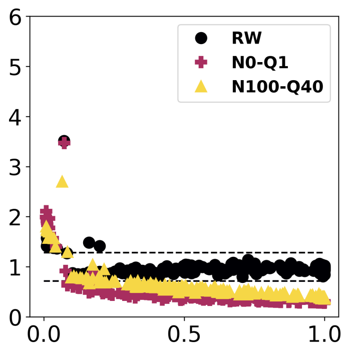
World	1 Hz	4 Hz	40 Hz
Half50m			
Half100m			

Fig. C.2 Half cluster environments

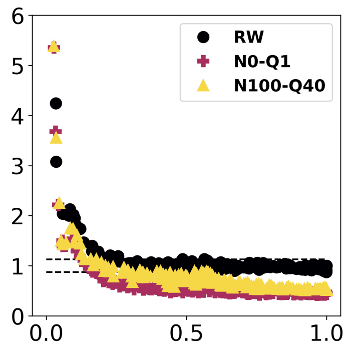
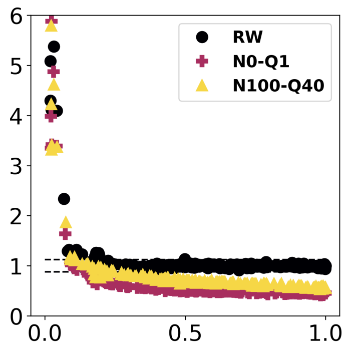
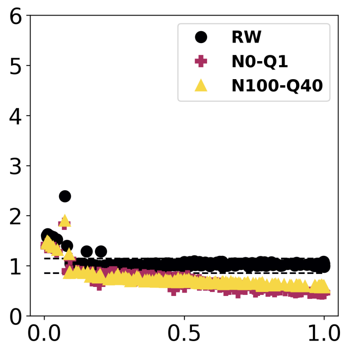
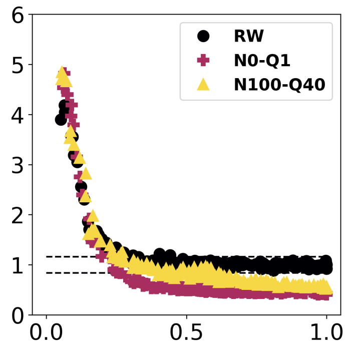
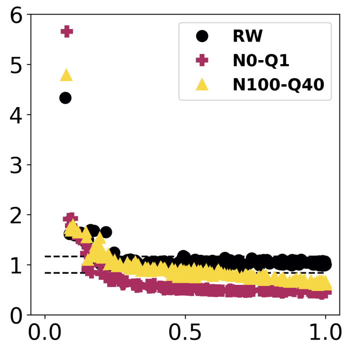
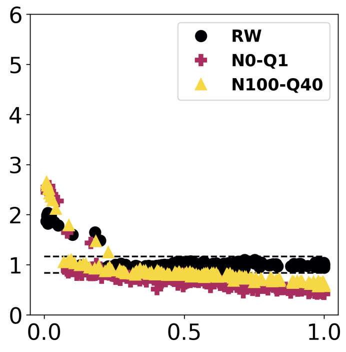
World	1 Hz	4 Hz	40 Hz
Two50m			
Two100m			

Fig. C.3 Two clusters environments

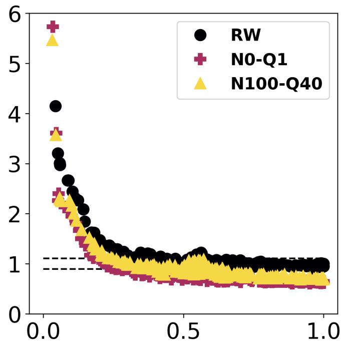
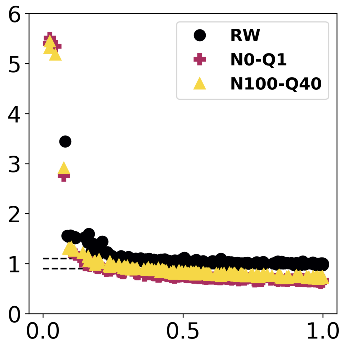
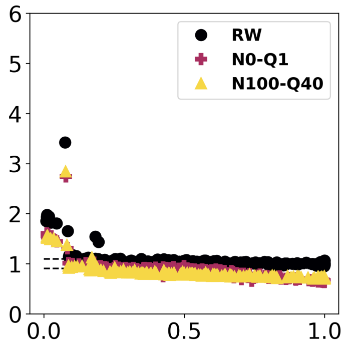
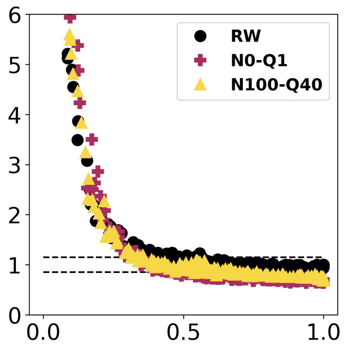
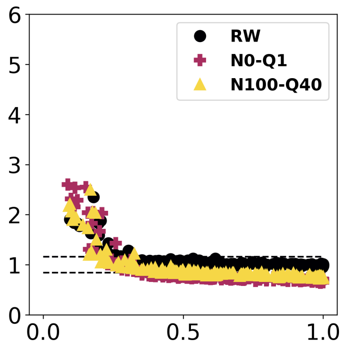
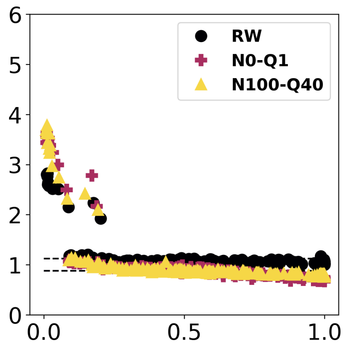
World	1 Hz	4 Hz	40 Hz
Four50m			
Four100m			

Fig. C.4 Four clusters environments

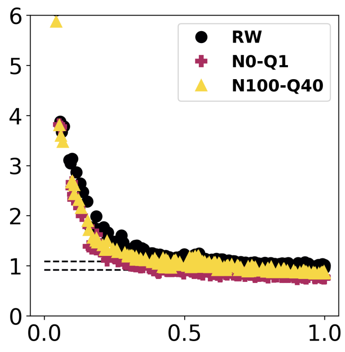
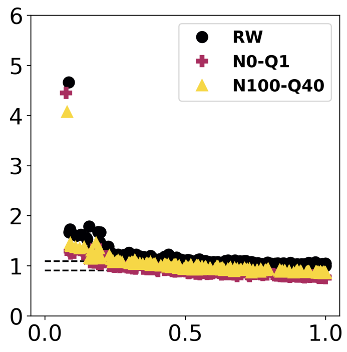
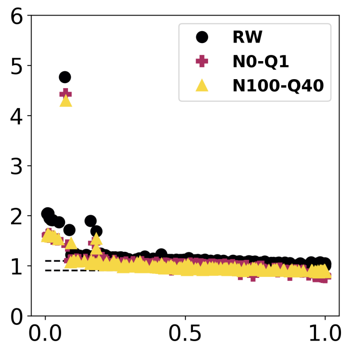
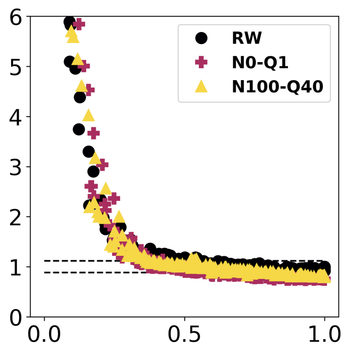
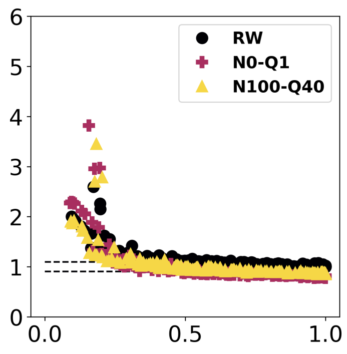
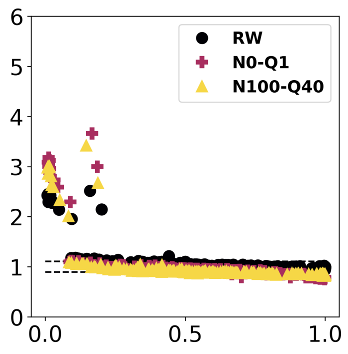
World	1 Hz	4 Hz	40 Hz
Uniform50m			
Uniform100m			

Fig. C.5 Uniform environments

C.2.2 Effects of update rates

In this visualisation (of same simulations), the effect of varying vision update rate on each algorithm is shown separately for Random Walk, N0-Q1 and N100-Q40. Foraging time on y-axis was normalised using corresponding algorithm with perfect vision and 40 Hz update rate.

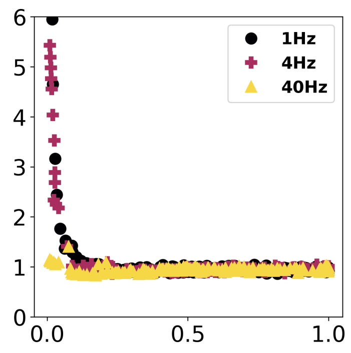
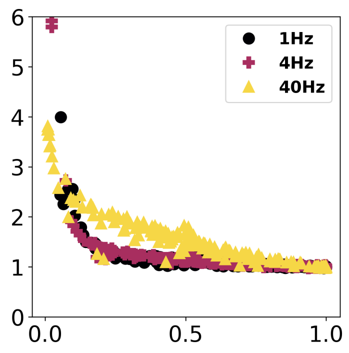
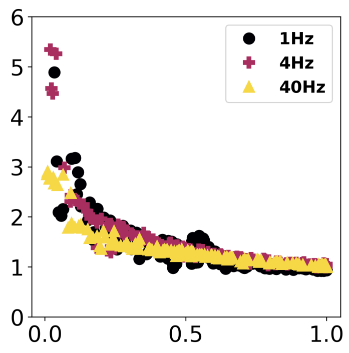
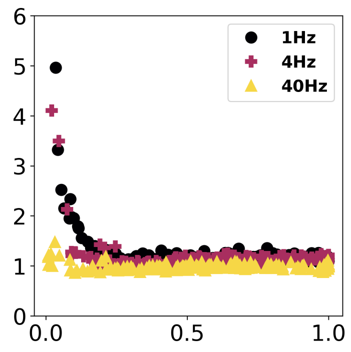
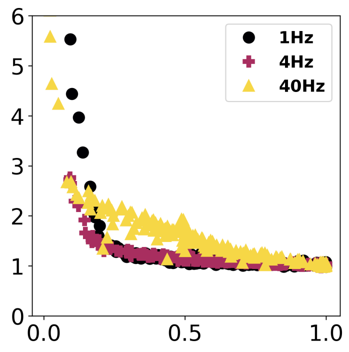
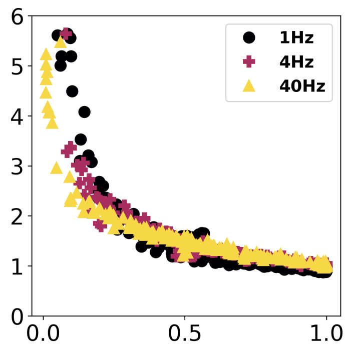
World	RW	N0-Q1	N100-Q40
One50m			
One100m			

Fig. C.6 One cluster environments

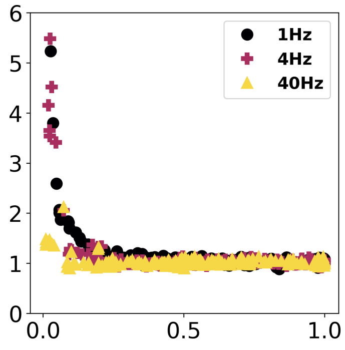
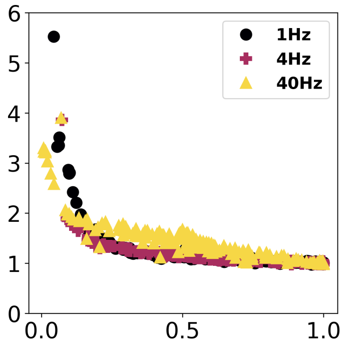
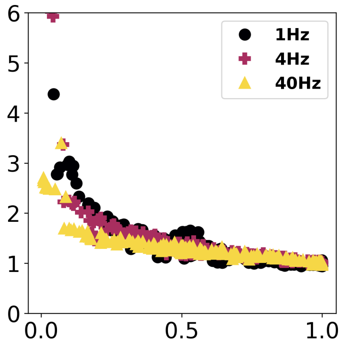
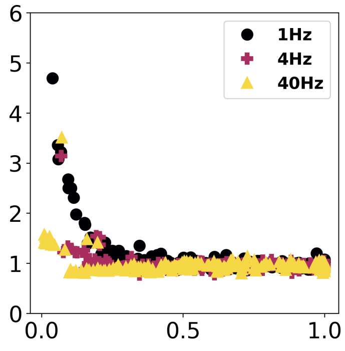
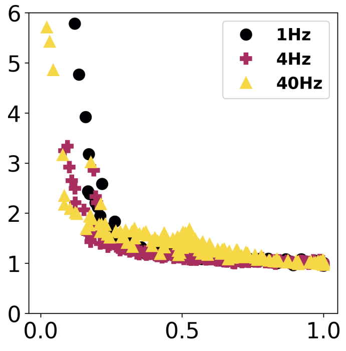
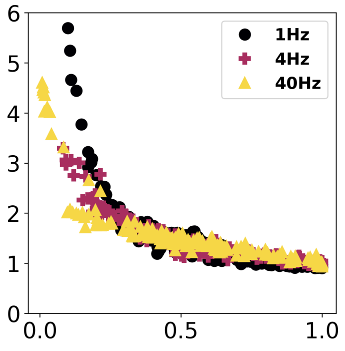
World	RW	N0-Q1	N100-Q40
Half50m			
Half100m			

Fig. C.7 Half cluster environments

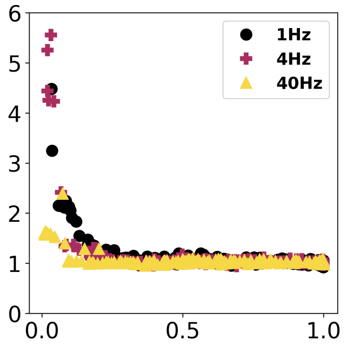
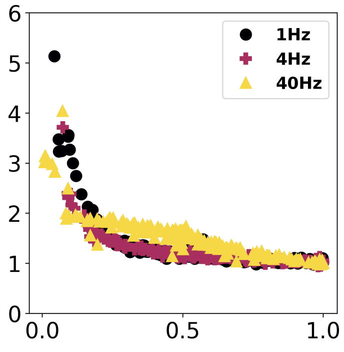
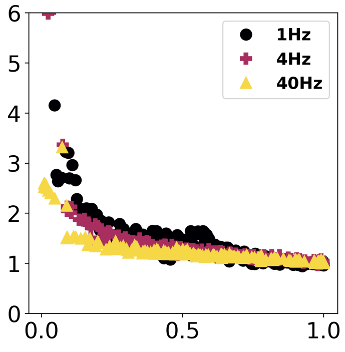
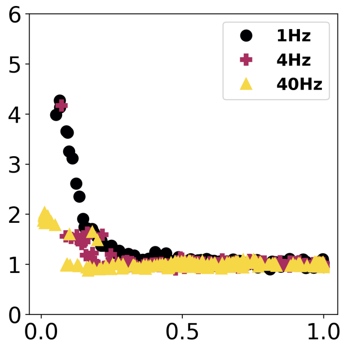
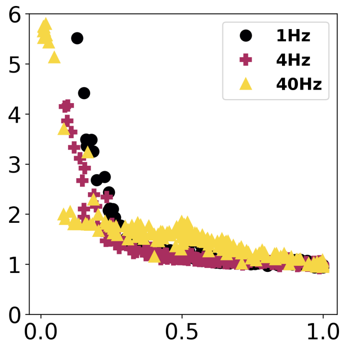
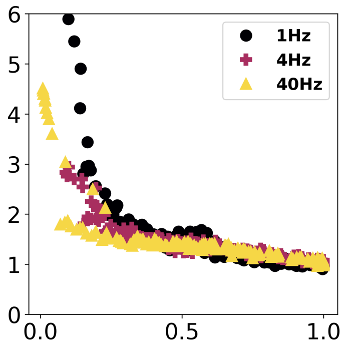
World	RW	N0-Q1	N100-Q40
Two50m			
Two100m			

Fig. C.8 Two clusters environments

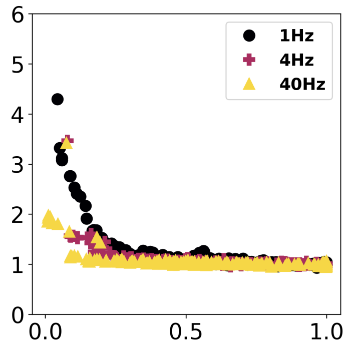
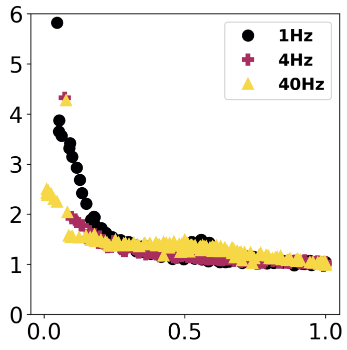
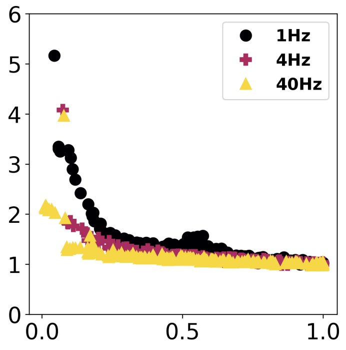
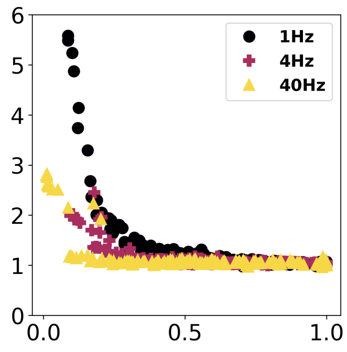
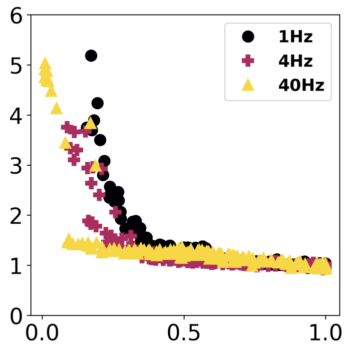
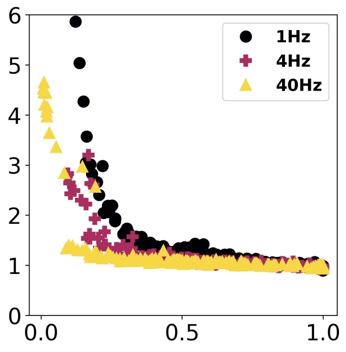
World	RW	N0-Q1	N100-Q40
Four50m			
Four100m			

Fig. C.9 Four clusters environments

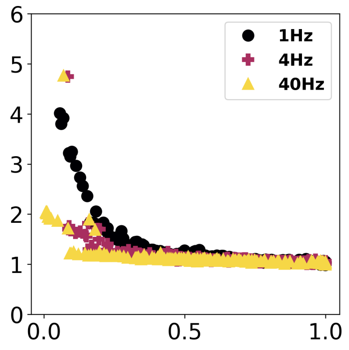
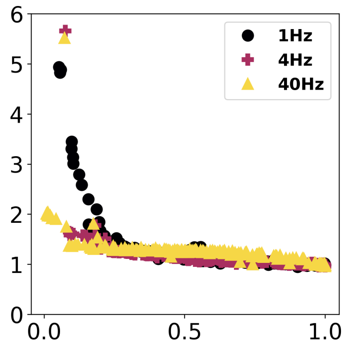
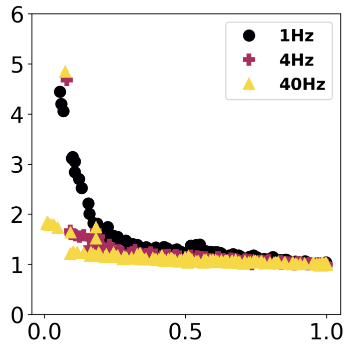
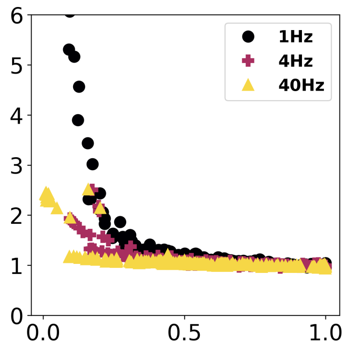
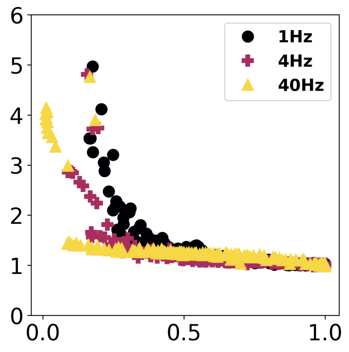
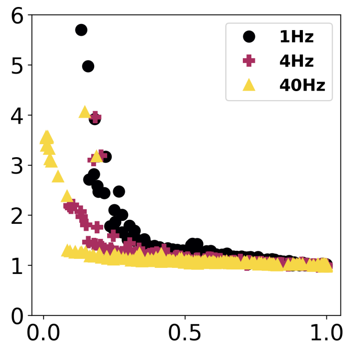
World	RW	N0-Q1	N100-Q40
Uniform50m			
Uniform100m			

Fig. C.10 Uniform environments