# Behavioral Diversity in Learning Robot Teams

A Thesis
Presented to
The Academic Faculty

by

**Tucker Balch**

In Partial Fullfilment
of the Requirements for the Degree
Doctor of Philosophy in Computer Science

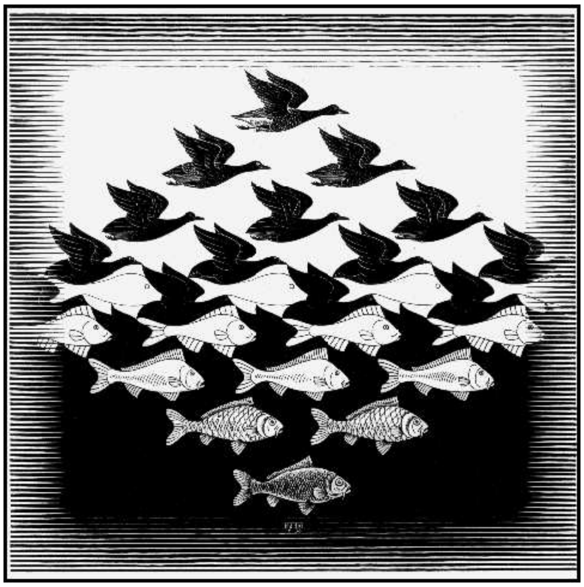
**Georgia Institute of Technology**
December 1998

# Behavioral Diversity in Learning Robot Teams

Thesis Committee:

Ronald C. Arkin, Chairman
Christopher G. Atkeson
Stephen P. DeWeerth
Jessica K. Hodgins
Karsten Schwan

Approved 13 November 1998



To Maria

# Acknowledgements

*"I'd rather be lucky than good."* – Col. Robert Goodman, USAF

A number of fortuitous circumstances conspired to bring me to Atlanta in 1991. Since then I joined Georgia Tech's Mobile Robot Laboratory, slipped the surly bonds of earth in the greatest fighter on the planet and found a Swedish beauty for my wife. I thank God for my good fortune.

Ron Arkin's Mobile Robot Laboratory is an exciting place to "grow up." New hardware comes in the door faster than it can be soldered together. Shelves groan under a load of robot sediment deposited over a decade of research: cables, wires, GPS antennas, miniature video cameras, mirrors, lenses, robotic arms, servos, tools, manuals and schematics. There are always more projects than students and more papers to write than time. In spite of the frenetic pace, Ron got two important messages across: robotics is fun, but discovery is worthless without accuracy and repeatability.

I benefited from the wisdom of four more world-class scientists on my committee; each of them contributed significantly to my research. I enjoyed many hours of debate with Chris Atkeson on the proper application of reinforcement learning and the meaning of diversity. He gave his time and ideas freely. Jessica Hodgins helped me plan a research agenda that both broadened and improved the significance of this work (without taking more time!). She has also been a terrific advisor in my academic, professional and grammatical life. I believe I've answered Steve DeWeerth's persistent question "where's the science?" This dissertation is much improved for his having asked it. Karsten Schwan has a wonderful ability to see through any problem and isolate the interesting parts. He helped me steer a relevant course through these long years of graduate work.

I want to specifically thank my cohorts in the Intelligent Systems *and Robotics* group for helping me build two award-winning multi-robot teams. In 1994 Juan Carlos Santamaría, Doug MacKenzie and Gary Boone breathed life into three vehicles that proved multiple robots can be cheap *and* reliable. Tom Collins and Ray Hsu gave them vision, Harold Forbes built their grippers. In 1997 Darrin Bentivegna and David Huggins handled the vision and Harold came out of retirement to build a new set of grippers. Robin Murphy, Jonathan Cameron and Khaled Ali worked with me on several other robot projects. Thanks also to Brian McNamara, David Johnson, Kent Lyons, Christopher Jurney and Zellyn Hunter for contributing their Java-based simulated soccer teams.

The quality of this dissertation has been significantly improved by those who reviewed it. Maria Hybinette read every chapter twice; some of them three or four times. She contributed many important ideas and suggestions. Three anonymous reviewers gave me excellent suggestions on Chapter 5 when it was submitted for conference publication. Peter Stone and Gary Boone reviewed parts of Chapters 5 and 6. Of course every word passed under Ron Arkin's pen. His reviews, suggestions and advice were invaluable.

Many others helped in important ways. I'm grateful to Gregory Abowd for lending me a PC during the last year of my research. He has been generous to many students with his equipment, time and funding. When I got "stuck" Jonathan Cameron gave me good advice: "do what you think is fun." Doug MacKenzie also provided some common sense: "never call your own research *cheesy*." Manuela Veloso graciously allowed me time at the beginning of my postdoctoral fellowship to complete this dissertation. Jeff Donahoo, Don Allison, Kurt Eiselt, John Stasko, Cathy Dunnahoo, Barbara Durham, Vicky Jackson, Catherine Ghoulson, Cathy Beam, Gunnar, Ingrid, Knut, Rosie, Kip, Big Ben and Little Ben all helped, each in their own way.

Flying fighters was nothing like I expected and neither was earning a Ph.D. It takes many long years and lots of help— both technical and otherwise. Thanks to *my mother*, Anne Cole Balch, for being a rock of moral support over the years. My dad, Clyde R. Balch, inspired me to go after whatever I wanted no matter how impossible it seemed. "There's always room at the top" he said, and he's proven it by reaching the top of his field and remaining there. Finally, thanks to my wife Maria for making sure I got done on time.

Pittsburgh, PA T. B.

# Contents


**1 Introduction** **1**

1.1 Why do agents specialize? . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Research question . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.3 Preview of contributions . . . . . . . . . . . . . . . . . . . . . . . . . 5

**2 Background and Related Work** **7**

2.1 Behavior-based robotics . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.2 Motor schema-based control . . . . . . . . . . . . . . . . . . . . . . . 9

2.2.1 Temporal sequencing of behavioral assemblages . . . . . . . . 10

2.3 Learning in behavior-based systems . . . . . . . . . . . . . . . . . . . 11

2.3.1 Reinforcement-learning . . . . . . . . . . . . . . . . . . . . . . 11

2.3.2 Q-learning . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

2.3.3 Dyna . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

2.3.4 Learning component behaviors . . . . . . . . . . . . . . . . . . 16

2.3.5 Learning a hierarchy of behaviors . . . . . . . . . . . . . . . . 17

2.3.6 Distributed RL . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

2.4 Multi-robot systems . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

2.4.1 Dudek's taxonomy . . . . . . . . . . . . . . . . . . . . . . . . . 18

2.4.2 Learning in behavior-based multi-robot systems . . . . . . . . 19

2.5 Tasks for multi-robot systems . . . . . . . . . . . . . . . . . . . . . . 20

2.5.1 Robotic foraging . . . . . . . . . . . . . . . . . . . . . . . . . . 20

2.5.2 Learning robotic foraging . . . . . . . . . . . . . . . . . . . . . 22

2.5.3 Robotic soccer . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

2.5.4 Learning robotic soccer . . . . . . . . . . . . . . . . . . . . . . 23

2.5.5 Robot formation . . . . . . . . . . . . . . . . . . . . . . . . . . 24

2.6 Social entropy theory . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

2.7 Discussion and summary . . . . . . . . . . . . . . . . . . . . . . . . 27

**3 Methodology** **29**

3.1 Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30

3.2 Task and performance specification . . . . . . . . . . . . . . . . . . 31

3.3 Behavioral design . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32

3.3.1 Example: foraging . . . . . . . . . . . . . . . . . . . . . . . 33

3.3.2 Configuring behavior with Clay . . . . . . . . . . . . . . . . 34

3.4 Reinforcement function specification . . . . . . . . . . . . . . . . . . 40

3.5 Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

3.6 Implementation on mobile robots . . . . . . . . . . . . . . . . . . . 43

3.6.1 Hardware platform . . . . . . . . . . . . . . . . . . . . . . . 44

3.7 Data collection and analysis . . . . . . . . . . . . . . . . . . . . . . 45

3.8 Discussion and summary . . . . . . . . . . . . . . . . . . . . . . . . 46

**4 Task and Reward** **49**

4.1 Characterization of multi-robot tasks . . . . . . . . . . . . . . . . . 50

4.1.1 Time . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52

4.1.2 The subject of action . . . . . . . . . . . . . . . . . . . . . . 52

4.1.3 Limited resources and competition . . . . . . . . . . . . . . . 52

4.1.4 Movement . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

4.1.5 Platform dependencies . . . . . . . . . . . . . . . . . . . . . 54

4.1.6 Classification of example tasks . . . . . . . . . . . . . . . . . 55

4.2 A taxonomy of reinforcement functions . . . . . . . . . . . . . . . . 56

4.2.1 Source of reward . . . . . . . . . . . . . . . . . . . . . . . . 57

4.2.2 Relation to performance . . . . . . . . . . . . . . . . . . . . 58

4.2.3 Time and continuity . . . . . . . . . . . . . . . . . . . . . . 59

4.2.4 Locality . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60

4.3 Discussion and summary . . . . . . . . . . . . . . . . . . . . . . . . 61

**5 Behavioral Difference and Diversity** **63**

5.1 Diversity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64

5.1.1 Introducing social entropy . . . . . . . . . . . . . . . . . . . 66

5.1.2 Why entropy is a useful measure of diversity . . . . . . . . . . 69

5.1.3 Limitations and alternative measures . . . . . . . . . . . . . . 71

5.2 Classification and hierarchic entropy . . . . . . . . . . . . . . . . . . . 74
5.2.1 Tools from numerical taxonomy . . . . . . . . . . . . . . . . . . 75
5.2.2 Clustering algorithms . . . . . . . . . . . . . . . . . . . . . . . 77
5.2.3 Hierarchic social entropy . . . . . . . . . . . . . . . . . . . . . 83
5.2.4 Why hierarchic social entropy is a useful metric . . . . . . . . 85
5.3 Behavioral difference . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
5.3.1 Example: multi-robot foraging . . . . . . . . . . . . . . . . . . . 93
5.3.2 Definition of behavioral difference . . . . . . . . . . . . . . . . 96
5.3.3 Limitations of the approach . . . . . . . . . . . . . . . . . . . 100
5.4 Discussion and summary . . . . . . . . . . . . . . . . . . . . . . . . . 100

**6 Diversity in Robot Foraging 103**
6.1 Task and performance metric . . . . . . . . . . . . . . . . . . . . . . 104
6.1.1 Example foraging robot systems . . . . . . . . . . . . . . . . . 105
6.1.2 Performance metric and task classification . . . . . . . . . . . 106
6.2 Behavioral design of hand-coded strategies . . . . . . . . . . . . . . . 107
6.2.1 Behavioral repertoire . . . . . . . . . . . . . . . . . . . . . . . . 108
6.2.2 Perceptual features . . . . . . . . . . . . . . . . . . . . . . . . . 110
6.2.3 Hand-coded sequencing strategies . . . . . . . . . . . . . . . . . 112
6.3 Performance of hand-coded strategies in simulation . . . . . . . . . . 115
6.3.1 Task performance . . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.3.2 Factors impacting performance . . . . . . . . . . . . . . . . . . 117
6.3.3 Diversity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 118
6.4 Design of learning strategies . . . . . . . . . . . . . . . . . . . . . . . 120
6.4.1 Reinforcement functions for foraging . . . . . . . . . . . . . . . 120
6.5 Performance of learning strategies in simulation . . . . . . . . . . . . 123
6.5.1 Task performance . . . . . . . . . . . . . . . . . . . . . . . . . . 123
6.5.2 Learning rate . . . . . . . . . . . . . . . . . . . . . . . . . . . . 124
6.5.3 Diversity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125
6.6 Implementation on mobile robots . . . . . . . . . . . . . . . . . . . . 128
6.6.1 Experiments in the AAAI "Find Life on Mars" task . . . . . . 130
6.7 Discussion and summary . . . . . . . . . . . . . . . . . . . . . . . . . 136

**7 Diversity in Robot Soccer** **139**
7.1 Task and performance metric . . . . . . . . . . . . . . . . . . . . . . . 140
7.2 Behavioral design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142
7.3 Design of learning strategies . . . . . . . . . . . . . . . . . . . . . . . 145
7.3.1 Reinforcement functions for soccer . . . . . . . . . . . . . . . 145
7.4 Performance with local and global rewards . . . . . . . . . . . . . . . 147
7.4.1 Task performance . . . . . . . . . . . . . . . . . . . . . . . . 147
7.4.2 Learning rate . . . . . . . . . . . . . . . . . . . . . . . . . . 147
7.4.3 Diversity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 148
7.5 Performance using $R_{\text{touch}}$ . . . . . . . . . . . . . . . . . . . . . . . . 150
7.5.1 Task performance . . . . . . . . . . . . . . . . . . . . . . . . 150
7.5.2 Learning rate . . . . . . . . . . . . . . . . . . . . . . . . . . 151
7.5.3 Diversity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 152
7.6 Performance versus human-designed teams . . . . . . . . . . . . . . . 153
7.7 Discussion and summary . . . . . . . . . . . . . . . . . . . . . . . . . 155

**8 Diversity in Cooperative Movement** **159**
8.1 Background and related work . . . . . . . . . . . . . . . . . . . . . . 160
8.2 Task and performance metric . . . . . . . . . . . . . . . . . . . . . . . 162
8.3 Behavioral design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 165
8.4 Fixed formation strategies . . . . . . . . . . . . . . . . . . . . . . . . 166
8.5 Learning cooperative movement strategies . . . . . . . . . . . . . . . 168
8.5.1 Reinforcement functions for cooperative movement . . . . . . 168
8.5.2 Task performance . . . . . . . . . . . . . . . . . . . . . . . . 169
8.5.3 Learning rate . . . . . . . . . . . . . . . . . . . . . . . . . . 170
8.5.4 Diversity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171
8.6 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 172
8.7 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 173

**9 Conclusion** **175**
9.1 Methodology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
9.2 Classifications of task and reward . . . . . . . . . . . . . . . . . . . . 177
9.3 New quantitative metrics . . . . . . . . . . . . . . . . . . . . . . . . . 178
9.4 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179

9.5 Discussion of results . . . . . . . . . . . . . . . . . . . . . . . . . . . 182
9.6 Future directions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 183

**A Motor Schema Formulations and Gain Values** **185**
A.1 Foraging behaviors . . . . . . . . . . . . . . . . . . . . . . . . . . . . 186
A.2 Soccer behaviors . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 189
A.3 Formation behaviors . . . . . . . . . . . . . . . . . . . . . . . . . . . 192

**B Spearman's Rank-Order Correlation Test** **195**

**Bibliography** **196**

# List of Tables


4.1 Summary of terms characterizing Multi-robot tasks. . . . . . . . . . . 51

4.2 Summary the Multi-robot reward taxonomy. . . . . . . . . . . . . . . 57

5.1 Difference matrix for the example society. . . . . . . . . . . . . . . . 78

5.2 Perceptual features available to the foraging robots. . . . . . . . . . . 95

5.3 Behaviors the robots select from in accomplishing the foraging task. . 95

5.4 The policies of two foraging robots. . . . . . . . . . . . . . . . . . . . 96

5.5 Sample evaluation of the behavioral difference between the two agents whose policies are listed in Table 4.3. . . . . . . . . . . . . . . . . . . 99

6.1 Behaviors the robots select from in accomplishing the foraging task. . 108

6.2 Perceptual features available to the foraging robots. . . . . . . . . . . 111

6.3 Summary of performance in hand-coded foraging robot trials. . . . . 130

6.4 Summary of performance in learning foraging robot trials. . . . . . . 131

7.1 The control team's policy summarized as look-up tables. . . . . . . . 143

7.2 The nine soccer robot policies possible for the learning agents discussed in the text. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 148

7.3 Performance and diversity results from robot soccer experiments. . . 156

9.1 Summary of the task and reward space explored in robot team experiments. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 181

9.2 Description of each task according to the classification introduced in Chapter 4 and a summary of the key results in each task. . . . . . . . 181

A.1 Motor schema parameter values and gains used in the behavioral assemblages for foraging. . . . . . . . . . . . . . . . . . . . . . . . . . . 190

A.2 Motor schema parameter values and gains used in the behavioral assemblages for soccer. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 192
A.3 Motor schema parameter values and gains used in the behavioral assemblages for cooperative movement. . . . . . . . . . . . . . . . . . . . 194

# List of Figures


2.1 Motor schema example. . . . . . . . . . . . . . . . . . . . . . . . . . . 9
2.2 The forage FSA. . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
2.3 Typical model of robotic reinforcement-learning. . . . . . . . . . . . 12

3.1 The forage FSA. . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
3.2 Clay source code for the declaration of perceptual and motor schemas employed in the foraging task. . . . . . . . . . . . . . . . . . . . . . . 37
3.3 Source code for the declaration of behavioral assemblages used in foraging. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
3.4 A behavioral sequence is configured using the `FSA` operator in Clay. . 39
3.5 A graphical representation of the hierarchically configured `FORAGE` behavior. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
3.6 Nomad 150 robotic platform. . . . . . . . . . . . . . . . . . . . . . . 43
3.7 Mobile robot's eye view of several attractor objects. . . . . . . . . . 44

5.1 Four collections of toy blocks. . . . . . . . . . . . . . . . . . . . . 65
5.2 In both of these groups there are the same number of blocks and the same number of subsets, but the proportion of elements in each subset is different. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
5.3 A new society is generated by combining two others. . . . . . . . . . 68
5.4 A spectrum of diversity. . . . . . . . . . . . . . . . . . . . . . . . 69
5.5 Two very different systems have similar entropy. . . . . . . . . . . . 71
5.6 One difficulty in the analysis of diversity. . . . . . . . . . . . . . . 72
5.7 Maximum taxonomic distance. . . . . . . . . . . . . . . . . . . . . . 73
5.8 Example classification using numerical techniques. . . . . . . . . . . 75
5.9 The branching structure of the dentograms for these two societies is the same. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76

5.10 Example society of elements distributed in a two-dimensional space. . . 78
5.11 Example of hierarchic nonoverlapping clustering. . . . . . . . . . . . . 78
5.12 A dilemma for nonoverlapping algorithms. . . . . . . . . . . . . . . . 79
5.13 Example of hierarchic overlapping clustering. . . . . . . . . . . . . . . 80
5.14 A comparison of clusterings generated by AutoClass (left) and $C_u$ clustering for the example data. . . . . . . . . . . . . . . . . . . . . . 81
5.15 A set of clusterings for a more compact society. . . . . . . . . . . . . 81
5.16 An example of two societies differing only in the degree of difference between elements. . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
5.17 Entropy depends on $h$. . . . . . . . . . . . . . . . . . . . . . . . . 83
5.18 A comparison of entropy versus $h$ for for two societies. . . . . . . . . 84
5.19 These two example systems are used to demonstrate how hierarchic social entropy can distinguish differences between societies regardless of scale. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
5.20 $C_u$ clusterings of $\mathcal{R}_{2x}$ for different values of $h$. . . . . . . . . . . . . . 86
5.21 Simple entropy of $\mathcal{R}_{2x}$ as a function of $h$. . . . . . . . . . . . . . . . 87
5.22 Hierarchic social entropy (bottom) is computed for three societies. . . 90
5.23 Hierarchic social entropy retains the basic properties of simple entropy. 91
5.24 Evaluating behavioral difference using an idealized "evaluation chamber." . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
5.25 An example task for a multiagent robot team. . . . . . . . . . . . . . 94

6.1 Simulation of the multi-forage task. . . . . . . . . . . . . . . . . . . . 104
6.2 Two foraging robot teams developed at Georgia Tech participated in recent AAAI Mobile Robot Competitions. . . . . . . . . . . . . . . . 106
6.3 An FSA representing the homogeneous foraging strategy. . . . . . . . 112
6.4 Inter-robot interference. . . . . . . . . . . . . . . . . . . . . . . . . . 113
6.5 FSAs representing specialized behaviors for foraging. . . . . . . . . . 114
6.6 Territorial behavioral sequences for foraging. . . . . . . . . . . . . . . 115
6.7 Performance of the hand-coded teams versus size of the team. . . . . 116
6.8 Simulations highlighting some of the factors that impact performance in hand-coded teams. . . . . . . . . . . . . . . . . . . . . . . . . . . 117
6.9 Diversity, as measured by social entropy, versus size of the team for hand-coded teams. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 119

6.10 Performance for learning systems versus the number of robots on a team. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123
6.11 Performance of the best learning and non-learning systems combined in one graph. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 124
6.12 Convergence for learning systems, measured as policy changes per trial; lower numbers indicate convergence to a stable policy. . . . . . 125
6.13 Performance versus trial number for learning systems with eight robots.126
6.14 Hierarchic social entropy versus size of the team for learning teams; larger numbers indicate greater diversity. . . . . . . . . . . . . . . . 126
6.15 Agent similarity matrix. . . . . . . . . . . . . . . . . . . . . . . . . . . 127
6.16 Division of the example team into castes based on behavioral difference.127
6.17 Nomad 150 robot equipped with passive gripper demonstrates three foraging behaviors. . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
6.18 Two robots demonstrate learned foraging polices in the Mobile Robot Lab. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
6.19 FSA representing the behavioral configuration used by Lewis and Clark at the AAAI-97 Mobile Robot Competition. . . . . . . . . . . . 133
6.20 Robot with an active gripper demonstrates a delivery sequence in the Mobile Robot Laboratory. . . . . . . . . . . . . . . . . . . . . . . . . 134
6.21 Diagram of laboratory arrangement for mobile robot experiments in the "Find Life on Mars" task. . . . . . . . . . . . . . . . . . . . . . . 135

7.1 Simulated and real robot soccer. . . . . . . . . . . . . . . . . . . . . 140
7.2 Policy convergence measured as average number of policy changes per trial for teams using local and global rewards. . . . . . . . . . . . . 148
7.3 Examples of homo- and heterogeneous learning soccer teams. . . . . 149
7.4 Score differentials for teams using the $R_{\text{touch}}$ reward function as $\gamma$ is swept from 0.1 to 1.0. . . . . . . . . . . . . . . . . . . . . . . . . . 151
7.5 Policy changes versus trial number for teams using the $R_{\text{touch}}$ reward function. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 151
7.6 Score versus trial number for teams trained using $R_{\text{touch}}$. . . . . . . . 152
7.7 Hierarchic social entropy for soccer teams trained using $R_{\text{touch}}$ as $\gamma$ varies from 0.1 to 1.0. . . . . . . . . . . . . . . . . . . . . . . . . . . 152
7.8 Example team trained using $R_{\text{touch}}$ in trials against **DTeam**. . . . . . 153

7.9 Performance versus trial number for games against the **BriSpec** team. 155
7.10 Performance versus trial number for games against two human-designed teams. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 156

8.1 A team of four robotic scout vehicles manufactured for DARPA's Demo II project. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
8.2 Four robots in leader-referenced *diamond, wedge, line* and *column* formations. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161
8.3 Large scale formation. . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
8.4 The simulation environment used in the experiments. Robots are initialized on the left. . . . . . . . . . . . . . . . . . . . . . . . . . . . 163
8.5 Sequence of images from an experimental run with four robots programmed to use the *diamond* behavior. . . . . . . . . . . . . . . . . . 164
8.6 Attachment site geometries for different formations. . . . . . . . . . . 166
8.7 Example four-robot formations resulting from the use of different attachment site geometries. . . . . . . . . . . . . . . . . . . . . . . . . . . 167
8.8 Average performance for fixed homogeneous teams. . . . . . . . . . . 168
8.9 Performance for learning navigating teams. . . . . . . . . . . . . . . . . 170
8.10 Policy convergence measured as average number of policy changes per trial for teams using local and rewards. . . . . . . . . . . . . . . . . . 170
8.11 Hierarchic social entropy of learning navigational teams. . . . . . . . 171
8.12 Agents exploit each other's formation behavior to move more quickly. 173

A.1 Parameters used in the calculation of **avoid** motor schema vectors. . 186
A.2 Parameters used in the calculation of **move_to** motor schema vectors. 187
A.3 Parameters used in the calculation of the **swirl_ball** motor schema vector. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 191

# Figure credits

- *Birds to Fish* on Dedication page and page 171, *Drawing Hands* on page 7, *Man with Cuboid* on page 29 and *Three Spheres* on page 63 by M.C. Escher 

- Photograph of bees on page 1 by the Smithsonian Institution. Used by permission.

- Figure 2.3 on page 12 is based on a drawing by Kaelbling, et al [KLM96].

- Photograph of a rat in a Skinner Box on page 49 by Professor Adrian Smith, Laboratory Animal Unit, Norwegian College of Veterinary Medicine, Oslo. Used by permission.

- Leaf-cutter ants photograph on page 101 by the Smithsonian Institution. Used by permission.

- Robot soccer competition photograph on page 136 by Hiroaki Kitano. Used by permission.

- Photograph of four F-84s on page 155 by the USAF Thunderbirds.

- Photograph of four robotic HUMMERS on page 156 by Lockheed-Martin.

# Chapter 1

# Introduction

## 1.1 Why do agents specialize?

Individuals in nearly all multiagent societies specialize: ant colonies have workers, soldiers and a queen; corporations have machinists, managers and presidents. Why does specialization occur? Are individuals born with skills and physical attributes that suit them for a job or do they just learn to fill a niche? It may be impossible to answer this question for natural systems, especially human groups, but we can investigate the issue in an artificial society: the robot team.

Multi-robot team design is challenging because performance depends significantly on issues that arise solely from interaction between agents. These interactions complicate development since they aren't obvious in the hardware or software design but only emerge in an operating team. Cooperation, robot-robot interference and communication, for instance, are not considerations for a single robot, but are crucial in multi-robot systems. Fortunately, the additional effort involved in deploying several robots is often rewarded by a more robust and efficient solution. Furthermore, individuals on a team are potentially simpler and less expensive than a single robot designed for the entire task. Still, automatic methods for matching multi-robot configuration to task don't yet exist; in most cases multiagent design is ad hoc. This research seeks to address that by applying a principled approach to the analysis and design of learning behavior-based multi-robot teams.

When feedback regarding success in a task is available, *reinforcement learning* can simplify robot systems design by shifting the task of behavioral configuration from the designer to the robots operating in their environment. For some simple tasks, given a sufficiently long trial, agents are even able to develop optimal policies [KLM96]. Rather than attempting to design an optimal system from the start, the designer imbues his robots with adaptability. The robots strive continuously to improve their performance; finding suitable behaviors automatically as they interact with the environment. This approach has the added benefit of allowing agents to adapt to changing environmental conditions. This is important since for non-trivial tasks, a fixed solution could never be optimal in all environments. For these reasons reinforcement learning is becoming pervasive in mobile robotics research. The behavior that arises in multi-robot societies using reinforcement learning is the focus of this research.

To date, reinforcement learning is most often applied in single robot systems; but recent work indicates multi-robot systems should benefit as well [Bal97c, Mat97]. As in other aspects of robot design however, when learning is extended from the individual to a team, new interactions arise. One consideration, for instance, involves the type of reinforcement used in training the agents. When an individual agent succeeds, should it be rewarded alone or should all members of the team share the reinforcement?

In multiagent teams there is also the question of similarity between the agents. Most research in multi-robot teams has centered on homogeneous systems, with work in heterogeneous systems focused primarily on mechanical and sensor differences (e.g. Parker's work [Par94]). Recent investigations indicate that *behaviorally* heterogeneous systems offer advantages in some tasks [Bal97c, FM97]. Teams of mechanically identical robots are especially interesting because they may be homogeneous or heterogeneous depending only on their behavior. Behavior is an extremely flexible dimension of robot heterogeneity since learning teams may *choose* between homo- or heterogeneity. The idea that individuals on a learning team might converge to different behaviors raises fascinating questions like: How and when do robot castes arise? Does the best policy for a robot depend on how many are on the team? When is a heterogeneous team better? This research provides a framework in which these types of questions may be answered.

## 1.2 Research question

Before the research question is presented, a few terms are briefly introduced (these are addressed in depth later in the dissertation). First, what is *behavioral difference* for robots? "Identical" robots from the same assembly line will differ in their performance due to differences in their construction, even if they are driven by identical control systems. This research, however, is concerned with the more significant differences that indicate a team's societal structure. As an example, suppose a team of robots are to learn a foraging task where red and blue objects must be collected. Slight differences in how well individuals navigate towards an object are not nearly as important as the difference between one agent that seeks red objects and another that seeks blue ones.

In this work, we look at the robots' control systems for factors which reveal behavioral difference. Assume for now that the agents follow a fixed policy. Two robots are defined as *behaviorally equivalent* if for every identical perception they select identical actions, otherwise there is a *behavioral difference* between them[1].

*Behavioral diversity* is defined in terms of behavioral differences between a robot team's individuals. The team is *behavioral homogeneous* if all robots are behaviorally equivalent to the others, and *behaviorally heterogeneous* otherwise. Diversity is a sliding scale between the extreme where all robots are identical and the other where all are different.

Finally, the *reinforcement function* is the feedback for agents learning a policy with reinforcement learning. Typically, the function is a measure of how well the agent is performing the task, so the agent should maximize the value of the function over time. In an equivalent but different view, the reinforcement signal represents

[1]This work focuses on comparisons between non-stochastic policies, but it could be extended to address probabilistic methods as well.

cost, and should thus be minimized.

**Research Question**

**How can the relationships between task, reinforcement function and behavioral diversity in learning multi-robot teams be investigated in a principled manner?**

Investigating the research question requires the exploration of a number of subsidiary issues. The research focuses on three in particular:

**Subsidiary Question 1**

**How can behavioral diversity be measured in a learning behavior-based multi-robot team?**

Once a learning team has converged to a stable set of behaviors, how can the structure of the resultant society be evaluated? The approach is to borrow the concept of *information entropy* from information theory as a measure of social diversity. More details on this topic are presented in Chapter 5.

**Subsidiary Question 2**

**How can reinforcement learning be implemented in a motor schema-based multi-robot team?**

Although many aspects of this work are useful to the multi-robot research field in general, the research focuses specifically on learning in motor schema-based multi-robot systems (the motor schema approach is behavior-based; see Chapter 2 for more detail). Several efforts have succeeded in using learning to adjust parameters in a motor schema systems [PAR92, SSR98, CAR92]. For this work, learning takes place at a higher level; larger-grained behavioral choices are considered than those explored previously. This research is more interested in whether an agent chooses to search for red or blue objects than the gain value on one of the many activated schemas.

Several types of reinforcement learning are appropriate for this investigation (e.g. [Sut91, Tes92, BSW90, WD92]). While the utility of the various types of learning in multiagent systems is an important topic, it is not the focus of this research. Q-learning was selected arbitrarily for use here, but it has the advantages of a strong mathematical basis and it is well established in the reinforcement learning literature.

**Subsidiary Question 3**

**How can reinforcement functions for learning multi-robot teams be characterized?**

Several types of reinforcement function have been proposed for individual robots and for learning multiagent systems. As of yet, however, there is no work organizing the approaches taxonomically or showing how they impact learning multi-robot teams. As an example of one dimension along which reinforcement functions vary, consider *global* versus *local* reinforcement. Global reinforcement is the type where a single reinforcement signal is simultaneously delivered to all agents, while in *local* reinforcement each agent is rewarded individually. Identifying important dimensions along which multiagent reinforcement functions vary will help establish an investigation space for the experimental portion of the research. This issue is addressed by a taxonomy of multiagent reinforcement functions.

## 1.3 Preview of contributions

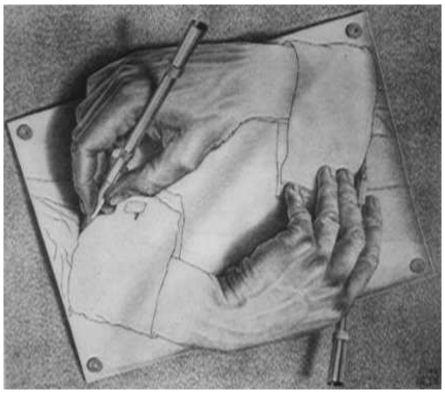
Results of the research will be of interest to several communities including autonomous robotics, machine learning, organizational science and ethology. Specifically, this research provides:

- A methodology for building and evaluating learning behavior-based robot teams.
- A classification of multi-robot tasks and a taxonomy of multi-robot reinforcement functions.
- Quantitative measures of behavioral diversity in multi-robot teams.
- Evaluation procedures for multi-robot task performance.
- A significant body of experimental results generated using the methodology in foraging, robot soccer and cooperative movement tasks.

The contributions are reviewed in detail in Chapter 9.

# Chapter 2

# Background and Related Work



This research concerns *reinforcement-learning* in *behavior-based multiagent* robot teams. Separately, each of these areas is backed by a significant body of research. Reinforcement-learning and behavior-based systems are especially well established, but the intersection of these two with multiagent systems is an emerging field. This chapter reviews the core research in robotic reinforcement learning, behavior-based robotics and learning multiagent robotic systems. The goal is to establish a base from which the research proceeds and to differentiate it from similar work.

## 2.1 Behavior-based robotics

In general, a robot is to satisfy a goal by selecting and executing a sequence of actions to achieve it. The goal may be a location to reach or a state of the world to be obtained (e.g. "put the soda cans in the wastebasket"). Sensors guide the selection of actions along the way. The "classic AI" approach to this problem is for the robot to process and store sensor readings with the aim of developing and maintaining an internal model of the external world. Deliberative strategies reason over the model and develop a plan (sequence of actions) for achieving the goal. Usually, intermediate failures trigger a complete re-planning. A famous example of the classic approach is Shakey the Robot [Nil84].

In contrast, over the last decade a "new wave" of robotics researchers have advanced a behavior-based view. Their central theme is to directly couple a robot's sensors and actuators so as to avoid the trouble of maintaining a model or deliberating over it. Rodney Brooks' subsumption architecture and Arkin's motor schemas are early examples [Bro86, Ark89]. Brooks asserts that world models are useless; the best model is the world itself. Space precludes a detailed review of the many behavior-based systems implemented since 1985, but some important qualities they all share include

**Tight sensor to motor coupling:** Sensor input is minimally processed before motor actions are selected.

**Minimal representation:** Many behavior-based systems do not maintain any internal state, or representation of the world at all. Some use just a few bits of memory.

**Speed:** Since the computational costs of reasoning over and maintaining a world model are avoided, behavior-based systems usually interact with their environment more quickly than "classic AI" systems.

**Composition:** The behavioral system is composed of several separate primitive behaviors. They may be arranged in layers [Bro86], or run in parallel [Ark89].

**Emergent properties:** The overall behavior of the system emerges through the interaction of of the several primitive behaviors.

**Robust performance in dynamic environments:** Since behavior-based systems are primarily sensor-oriented, they quickly respond to changes in the environment.

Two frequently raised concerns regarding behavior-based approaches are

**Behaviors are hand-coded:** Many existing behavior-based robotic systems *are* comprised of hand-coded behaviors. To address this, researchers are investigating automatic approaches including genetic algorithms [PAR92] and reinforcement-learning [MC92].

**"The fly at the window":** A fly buzzing at a window is a classic example of a behavior-based system caught in a local minimum. Behavior-based systems are often unable to recognize failure and select alternative strategies. This is one reason some argue for the integration of deliberative systems (able to detect a failure) and reactive systems (able to act quickly in the dynamic environment). Some approaches avoid the local minimum problem altogether [CG93], or use short term memory-based mechanisms to move out of them [BA93].

## 2.2 Motor schema-based control

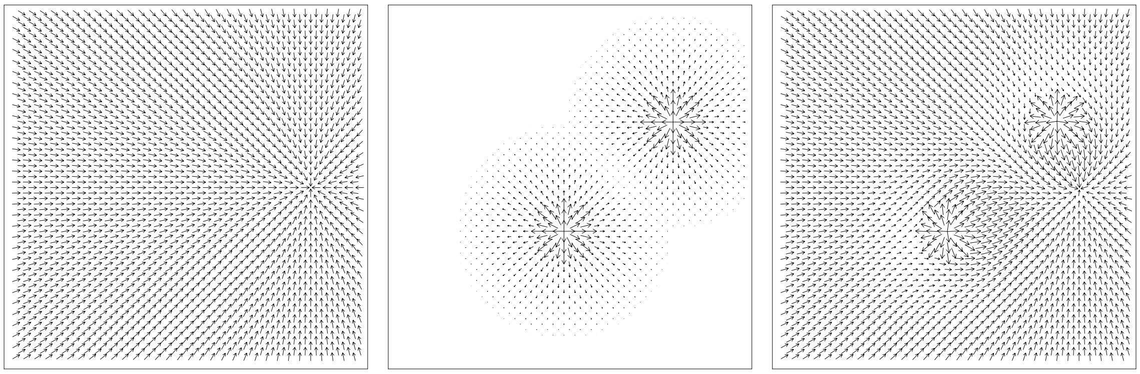


**Figure 2.1**: Motor schema example. The diagram on the left shows a vector field corresponding to a **move-to-goal** schema, pulling the robot to a location on the right. The center diagram shows an **avoid-obstacles** field, repelling the robot from two sensed obstacles. On the right, the two schemas are summed, resulting in a complete behavior for reaching the goal. It is important to note that the entire field is never computed, only the vectors for the robot's current location.

Motor schemas are an important example of behavior-based robot control. The motor schema paradigm is the central method in use at the Georgia Tech Mobile Robot Laboratory and is platform for this research.

Motor schemas are the reactive component of Arkin's Autonomous Robot Architecture (AuRA) [AB97]. AuRA's design integrates deliberative planning at a top level with behavior-based motor control at the bottom. The lower levels, concerned with executing the reactive behaviors are incorporated in this research.

Individual motor schemas, or primitive behaviors, express separate goals or constraints for a task. As an example, important schemas for a navigational task would include **avoid_obstacles** and **move_to_goal**. Since schemas are independent, they can run concurrently, providing parallelism and speed. Sensor input is processed by perceptual schemas embedded in the motor behaviors. Perceptual processing is minimal and provides just the information pertinent to the motor schema. For instance, a **find_obstacles** perceptual schema which provides a list of sensed obstacles is embedded in the **avoid_obstacles** motor schema.

The concurrently running motor schemas are integrated as follows. First, each produces a vector indicating the direction the robot should move to satisfy that schema's goal or constraint. The magnitude of the vector indicates the importance of achieving it. It is not so critical, for instance, to avoid an obstacle if it is distant, but crucial if close by. The magnitude of the **avoid_obstacle** vector is correspondingly small for distant obstacles and large for close ones. The importance of motor schemas relative to each other is indicated by a gain value for each one. The gain is usually set by a human designer, but may also be determined through automatic means, including on-line learning [CAR92], case-based reasoning [RS93] or genetic algorithms [PAR92]. Each motor vector is multiplied by the associated gain value and the results are summed and normalized. The resultant vector is sent to the robot hardware for execution. An example of this process is illustrated in Figure 2.1.

The approach bears a strong resemblance to potential field methods [CG93], but with an important difference: the entire field is never computed. In the example figure an entire field is shown, but this is only for visualization purposes. The robot only computes the vectors that apply to its present location and perceptual state.

### 2.2.1 Temporal sequencing of behavioral assemblages

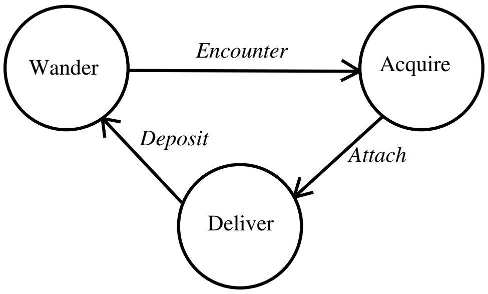



**Figure 2.2**: The forage FSA.

As illustrated above for navigation, motor schemas may be grouped to form more complex, emergent behaviors. Groups of behaviors are referred to as *behavioral assemblages* . One way behavioral assemblages may be used in solving complex tasks is to develop an assemblage for each sub-task and to execute the assemblages in an appropriate sequence. The steps in the sequence are separate *behavioral*

*states*. Perceptual events that cause transitions from one behavioral state to another are called *perceptual triggers*. A resulting task solving strategy can be represented as a Finite State Automaton (FSA). This technique is referred to as *temporal sequencing* [AM94].

As an example task where temporal sequencing is useful, consider foraging. Robot foraging behaviors have been examined in detail at Georgia Tech (see [BA95a] and [ABN93]). The forage task for a robot is to wander about the environment looking for items of interest (attractors). Upon encountering one of these attractors, the robot moves towards it, finally attaching itself. After attachment the robot navigates to the homebase where it deposits the attractor.

In this approach to solving the forage task, a robot can be in one of three behavioral states: *wander*, *acquire* and *deliver*. The robot begins in the *wander* state. If there are no attractors within the robot's field of view, the robot remains in *wander* until one is encountered. When an attractor is encountered, a transition to the *acquire* state is triggered. While in the *acquire* state, the robot moves towards the attractor and when it is sufficiently close, attaches to it. The last state, *deliver*, is triggered when the robot attaches to the attractor. While in the *deliver* state the robot carries the attractor back to home base. Upon reaching home base, the robot deposits the attractor there and reverts back to the *wander* state. An FSA for the forage task is illustrated in Figure 2.2.

## 2.3 Learning in behavior-based systems

Most of the behavior-based approaches presented so far in this chapter focus on static behavioral configurations. Even though behavior-based approaches are robust for many tasks and environments, they are not necessarily adaptive. We now consider some of the ways learning can be integrated into a behavior-based system.

### 2.3.1 Reinforcement-learning

Reinforcement learning offers a powerful set of techniques that allow a robot to learn a task without requiring its designer to fully specify how it should be carried out. If the task is feasible and feedback regarding how well the agent is doing is provided, several reinforcement learning techniques are *guaranteed* to converge (within an arbitrary $\epsilon$) to the *optimal* solution [WD92, TVR96]. The guarantees are tempered by rather strong conditions for convergence; Q-learning for example, requires all actions to be repeatedly sampled in all states.

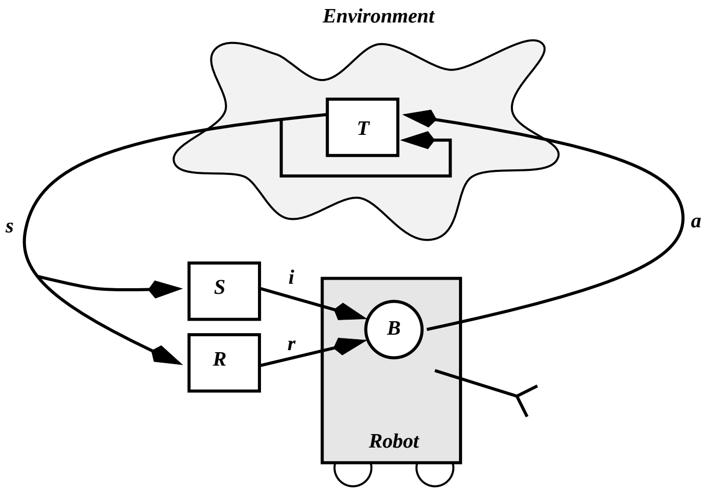



**Figure 2.3**: Typical model of robotic reinforcement-learning (adapted from Kaelbling).

A standard model of robotic reinforcement-learning is illustrated in Figure 2.3 [KLM96]. At each step of interaction, the robot receives input from the environment, $i$, which may be filtered by the robot's sensors $S$. It also receives a reinforcement signal $r$, which depends on conditions in the environment and $R$, the reinforcement function. $r$ is an indication to the robot of how well it is performing. The agent's behavior $B$ should choose actions that maximize the sum of reinforcement signals: The robot's action, $a$, effects a change in the environment modeled by $T$. Depending on the type of learning, the agent may or may not be provided $T$ and $R$.

If $T$ and $R$ are known, an optimal $B$ may be found using dynamic programming [Bel57]. Once the optimal behavior has been determined, the reinforcement signal $r$, is unnecessary and the behavior depends only on the current perceived state of the environment $i$. A function that selects an action depending on $i$ is referred to as a *policy* and is denoted by $\pi(i)$. Optimal policies are labeled $\pi^*(i)$. Even though these approaches may generate optimal policies it often takes too long for practical applications [LDK95].

For many robot tasks it is not reasonable to expect that perfect models of interaction with the environment can be known a priori. When $T$ and $R$ are not provided, the robot must learn the best action for each situation through trial and error. Research in reinforcement learning centers on developing algorithms to compute a policy that converges to optimal as quickly as possible. Example algorithms include Dyna [Sut91], $TD(\lambda)$ [Tes92], Adaptive Heuristic Critic (AHC) [BSW90], Model-Based [AAMR88] and Q-learning [WD92].

Although reinforcement-learning bears a resemblance to supervised learning, the two are distinct. In supervised learning, the agent is presented with input/output pairs, where the given output is presumably the best choice of action. The input/output pairs correspond to $s$ and $a$ of the reinforcement-learning model. In the case of reinforcement-learning, the agent is provided with the present state, a reward and the next state $(s_t, r_t, s_{t+1})$, but the robot is not told which action would be in its best interest. It can only discover this by testing actions and evaluating rewards.

Learning agents strive for optimal performance, but what is "optimal?" It was mentioned earlier that the robot should attempt to maximize $r$ over time. The length of time however, is important. If we know the length of an agent's life for instance, the best course is to maximize the sum of $r$ over that entire period (this corresponds to a finite-horizon). Reinforcement-learning methods differ in their definitions of optimality; three views predominate:

- **Finite-horizon** where the the agent takes actions that maximize the sum of rewards over a finite number of steps.
- **Average-reward** where the the agent takes actions that maximize the average reward over all steps in the future.
- **Infinite-horizon discounted** where the the agent takes actions that maximize the reward over all future steps, but the future rewards are discounted geometrically.

Another important distinction between the various reinforcement-learning methods concerns their use of models. In *model-based* algorithms, the agent learns approximations to $T$ and $R$, then utilizes the models to develop a policy. In *model-free* methods the agent learns the policy directly without a model. In this research we will focus on Watkins' model-free Q-learning.

### 2.3.2 Q-learning

Q-learning is a type of reinforcement-learning in which the value of taking each possible action in each situation is represented as a utility function, $Q(s, a)$. Where $s$ the state or situation and $a$ is a possible action. For the purposes of discussing Q-learning, assume the robot's sensors pass the world state on to the agent unmodified (thus $s$ is used instead of $i$ to signify the state). If the function is properly computed, an agent can act optimally simply by looking up the best action for any situation. The problem is to compute the $Q(s, a)$ that provides an optimal policy. Watkins [WD92] has developed an algorithm for determining $Q(s, a)$ that converges to optimal. Watkins' prefers to represent $Q(s, a)$ as a table, $Q[s, a]$ and asserts in [WD92] that the algorithm is not guaranteed to converge otherwise.

Q-learning agents seek to maximize the infinite-horizon discounted sum of $r$. $Q[s, a]$ is the value of choosing action $a$ in situation $s$. The policy for a robot using Q-values is to chose the action that maximizes $Q[s, a]$. So

$$\pi(s) = \text{argmax}_a Q[s, a]; \tag{2.1}$$

The problem then is to compute and update Q-values based on interaction with the environment. In discussing Q-values, it is useful to consider the value of reaching a particular state. For a state $s_t$ the value of $s_t$ is defined as:

$$V(s_t) \quad = \quad R(s_t, a_t) + \gamma R(s_{t+1}, a_{t+1}) + \gamma^2 R(s_{t+2}, a_{t+2}) + \gamma^3 R(s_{t+3}, a_{t+3}) \ldots$$

$R(s_t, a_t)$ is the reward for being in state $s_t$ and executing action $a_t$; $s_{t+1}$ is the succeeding state. $\gamma$ is the discount factor with $\gamma < 1$. $V(s_t)$ therefore, is the sum of all future rewards discounted at the rate $\gamma$. $V = V^*$ if the states are reached following an optimal policy. $V(s_t)$ may also be written recursively as

$$V(s_t) \quad = \quad R(s_t, a_t) + \gamma V(s_{t+1})$$

If each $Q[s, a]$ were set as follows, the policy of selecting actions based on them would be optimal:

$$Q[s_t, a_t] \quad = \quad R(s_t, a_t) + \gamma V^*(s_{t+1})$$

where $s_{t+1}$ is the state reached by applying action $a_t$ in state $s_t$ and subsequent actions are selected optimally. Unfortunately $Q[s, a]$ cannot be computed directly because $R(s)$ and $V(s)$ aren't known initially. Watkins [WD92] introduced the following scheme for updating Q-values as an agent interacts with the environment and receives rewards:

$$Q[s_t, a_t] \quad = \quad (1-\alpha)\ \underbrace{Q[s_t, a_t]}_{\text{old value}} + \alpha \underbrace{(R(s_{t+1}, a_{t+1}) + \gamma \max_u Q[s_{t+1}, u])}_{\text{improved estimate}} \tag{2.2}$$

This update is applied each time action $a_t$ "fires" in state $s_t$. The first term is the old Q-value, while the second is an improved estimate based on an actual reward and the estimated value of the subsequent state. $\alpha$ is a learning rate that indicates how much "trust" should be given the improved estimate. In the second term, $\max_u Q(s_{t+1}, u)$ is an approximation of $V(s_{t+1})$ (if $Q = Q^*$ it is easy to show that $V^*(s) = \max_u Q[s, u]$). Watkins proved this iteration converges to $Q^*$ under the condition that the learning set includes an infinite number of episodes for each state and action. This is a strong condition, but under the stochastic conditions of his theorem, no method could be guaranteed to find the optimal policy under weaker conditions.

### 2.3.3 Dyna

Model-free systems like Q-learning are computationally simple, but require many experience steps to converge. Model-based systems seek to reduce the cost of experience in the real-world (as in risk of damage to the robot) by using experience to model interaction with the world, then developing a policy based on the model.

Dyna [Sut91], like Q-learning, represents the utility of executing a particular action $a$ in a particular state $s$ as $Q(s, a)$, but it uses models of $T$ and $R$ to compute $Q$. The models of $T$ and $R$ are referred to as $\hat{T}$ and $\hat{R}$ respectively. Carrying forward the notation introduced earlier for Q-learning, we will consider $Q, \hat{T}$, and $\hat{R}$ as tables.

At each step of interaction with the environment, Dyna records an experience-tuple, $(s_t, a_t, s_{t+1}, r_t)$. Where action $a_t$ applied in $s_t$ results in a new state $s_{t+1}$ and reward $r_t$. Next $\hat{T}$ and $\hat{R}$ are updated based on the observation using a simple statistical model. $\hat{T}[s_t, a_t, s_{t+1}]$ is the probability that state $s_{t+1}$ results from applying action $a_t$ in state $s_t$. Similarly $\hat{R}[s_t, a_t]$ is the estimated reward for executing $a_t$ in state $s_t$. Also, at each step, $Q[s_t, a_t]$ is updated as follows:

$$Q[s_t, a_t] \quad = \quad \hat{R}[s_t, a_t] + \gamma \sum_{s_{t+1}} \hat{T}[s_t, a_t, s_{t+1}] \max u Q[s_{t+1}, u] \tag{2.3}$$

Dyna also performs a fixed number of additional updates to random state-action pairs, as follows:

$$Q[s_k, a_k] \quad = \quad \hat{R}[s_k, a_k] + \gamma \sum_{s_{t+1}} \hat{T}[s_k, a_k, s_{t+1}] \max uQ[s_{t+1}, u] \tag{2.4}$$

Finally, as in Q-learning, Dyna uses the Q-values to select actions.

Kaelbling evaluated Dyna and Q-learning in a simulated navigational task [KLM96]. She found Dyna to require an order of magnitude fewer steps of experience than Q-learning to converge to an optimal policy, but Dyna uses about six times more compute cycles.

### 2.3.4 Learning component behaviors

Reinforcement-learning is one way for a robot to learn appropriate sequences of action to attain a goal. Mahadevan and Connell [MC92] have applied Q-learning in a slightly different manner: to learn the component behaviors within a pre-defined sequence. The particular task they investigate is for a robot to find, then push a box across a room. They pre-define three behavioral states **F, P** and **U** for find-box, push-box and unwedge-box respectively; they also define conditions under which the robot transitions from one state to another. Separate reinforcement functions and tables of Q-values apply for each state.

The state vector $s$ is composed of local sonar occupancy information, infra-red bump sensors and a "stuck" sensor. The possible actions are: go forward, turn left, turn hard left, turn right and turn hard right. Since the state space is rather large, Mahadevan sought ways to reduce it, including weighted Hamming distance and statistical clustering to group similar states. Using this approach, their robot, OBELIX was able to learn to perform better than hand-coded behaviors for box-pushing.

Mahdevan's sequence of behaviors is similar to the temporal-sequencing approach outlined earlier. An important difference is that learning takes place in the behavioral states. The significance of Mahadevan's result is that Q-learning is useful in learning sequences within sequences of behaviors; it may be applied at several levels.

### 2.3.5 Learning a hierarchy of behaviors

In research at Carnegie Mellon University [Lin93], Lin developed a method for Q-learning to be applied hierarchically, so that complex tasks are learned at several levels. He argues that by by decomposing the task into sub-tasks and learning at the sub-task and task level, the overall rate of learning is increased compared to monolithic learners. The approach follows these steps (from [Lin93]):

1. **Task decomposition.** A complex task is decomposed into multiple elementary tasks. The original complex task is thus reduced to the task of integrating the solutions to the elementary tasks to form the solution to the original task. Task decomposition involves designing a reward function for each elementary task.

2. **Learning elementary skills.** An elementary skill needs to be learned to solve each elementary task. Here Q-learning can be used and each elementary skill corresponds to a Q-function: $Q(s, action)$.

3. **Learning a high-level skill.** A high-level skill for coordinating the elementary skills needs to be learned in order to solve the original task. Learning a high-level skill is conceptually similar to learning and elementary skill. Again, Q-learning can be used and the high-level skill corresponds to a Q-function: $Q(s, skill)$.

In Lin's work, the job of task decomposition and assigning reward functions to sub-tasks is carried out by humans, the rest is learned by the robot. Lin's results show significantly faster convergence and better performance for agents that use this technique, compared to those learning an entire task at once.

Similarities between Lin's decomposition and temporal-sequencing for assemblages of motor schemas (Section 2.2.1) are readily apparent. Lin's sub-tasks or elementary skills correspond to behavioral assemblages, while a high-level skill is a sequence of assemblages. Learning at the high-level is equivalent to learning the state-transitions of an FSA (as in Figure 2.2) and learning the elementary skills corresponds to tuning individual states or behavioral assemblages.

A difficulty with reinforcement learning in complex tasks is that performance may converge slowly, or not at all. The problem is aggravated when only occasional (delayed) reinforcement is provided; after the task is completed for instance. Some other learning speedups Lin examined to address this include experience replay and teaching [Lin91]. Experience replay involves presenting sequences of previous experiences to the Q-learning algorithm. Presumably this serves to reduce the problems of having to gather costly or rare experiences more than once. For teaching, a human leads the robot through a series of actions to achieve the goal and the sequence of experiences thus gathered are used to train the system.

### 2.3.6 Distributed RL

The reinforcement learning approaches outlined so far use a centralized scheme for learning when particular sub-behaviors should be activated. Maes and Brooks [MB90] propose an alternative, distributed mechanism. In their scheme, each behavior learns for itself when it ought to be applied. They pre-define a set of behaviors and a set of binary perceptual conditions. Each behavior learns when it should be "on" or "off" based on the perceptual conditions. Positive and negative feedback are provided to guide the learning.

The behaviors learn, for each perceptual condition, *relevance* and *reliability* of the behavior to the condition. A behavior is relevant in the presence of a particular condition if it is positively correlated to positive feedback, i.e. positive feedback is likely to be received if the behavior is activated in that condition. A behavior is reliable if the probability of receiving the feedback is close to 1. The behaviors learn both negative relevance (when they should be turned off) and positive relevance for each condition. Conditions that are neither positively or negatively relevant are eventually dropped from consideration.

Maes and Brooks tested their approach on a robotic hexapod. Negative feedback is provided if either the front or rear of the robot touches the ground. Positive feedback is based on the rotation of a trailing wheel that measures forward motion. The robot was able to learn to walk, using a tripod gait in two to ten minutes. This is a significant success, but mathematical properties (rate of convergence for instance) of the technique have not yet been established rigorously.

## 2.4 Multi-robot systems

### 2.4.1 Dudek's taxonomy

The taxonomy of multiagent systems introduced by Dudek [DJW93] is becoming an important reference in multiagent literature. It provides a useful set of axes for discriminating between the many types of multiagent robot systems. The following is a synopsis of his taxonomy by dimension:

**SIZE** The number of robots in the environment. Types include ALONE, PAIR, LIM (a limited number of robots) and INF (unlimited).

**COM** Communication range; NONE, NEAR, INF.

**TOP** Communication topology; BROAD (broadcast), ADD (address), TREE and GRAPH.

**BAND** Bandwidth of the communication; ZERO, LOW, HIGH and MOTION. BAND-MOTION is a special case where the communication cost is equal to the cost of moving the robot between two locations.

**ARR** The rate at which the collective can spatially re-organize itself; STATIC, COMM (the members coordinate rearrangement using communication) and DYN (dynamic).

**PROC** The processing ability of individual units in the collective; SUM (non-linear summation), FSA (finite state automaton), PDA (push-down-automaton), TME (Turing machine equivalent).

**CMP** Composition; HOM (homogeneous), HET (heterogeneous).

The taxonomy provides an important context for this research. In particular, the CMP (composition) axis will be explored in terms of agent behavior.

### 2.4.2 Learning in behavior-based multi-robot systems

To date, only a few researchers have investigated learning in multi-robot systems, most notably Matarić [Mat92, Mat94] and Parker [Par94]. Parker developed the ALLIANCE architecture [Par94] for controlling teams of physically heterogeneous robots. The system is built on the behavior-based subsumption architecture [Bro86]. In a manner similar to temporal sequencing (Section 2.2.1), tasks are broken into sub-tasks, with groups of behaviors addressing each sub-task. At the highest level, mutually inhibitory *motivational behaviors* direct the overall behavior of the robot, activating in turn lower-level behaviors that combine to solve the sub-task.

Along with the typical sensor-based conditions that might trigger motivational behaviors Parker adds *impatience* and *acquiescence*. Impatience increases if no other robot is attempting to solve the sub-task associated with a motivational behavior, while acquiescence inhibits a behavior if the robot is not meeting with success. The combined result of the ordinary conditions, impatience and acquiescence in a group is that the group cooperates in striving to solve the overall task.

ALLIANCE was extended to L-ALLIANCE which provides for learning. Agents in L-ALLIANCE are able to learn the abilities of other robots to complete sub-task

This information, coupled with a strategy whereby the robot most suited for each task executes it, enables robot teams to significantly improve performance over other techniques.

Matarić's work is more closely related to this research because it involves the use of reinforcement learning. Her work in multi-robot learning systems is examined in Section 2.5.2.

## 2.5 Tasks for multi-robot systems

This research investigates the relationships between reinforcement function, performance and diversity in three multi-robot tasks: robotic foraging, soccer and formation maintenance. This section introduces each task and provides some background on the related research in each domain.

### 2.5.1 Robotic foraging

The forage task involves the collection of objects of interest (attractors) scattered about the environment. In a typical strategy, an agent begins by wandering about the environment looking for attractors. Upon encountering an attractor, the robot moves towards it and grasps it. After attachment, the robot returns the object to a home base. In some foraging strategies attractors may be handed off to another agent for final delivery.

Foraging has a strong biological basis. Many ant species, for instance, perform the forage task as they gather food. Foraging is also an important subject of research in the mobile robotics community; it relates to many real-world problems [Ark92, ABN93, BA95a, GM97, FM97]. Among other things, foraging robots may find potential use in mining operations, explosive ordnance disposal and waste or specimen collection in hazardous environments (e.g. the Mars Pathfinder rover).

At Georgia Tech, Arkin and Balch have investigated several homogeneous strategies for robot foraging. [Ark92, ABN93, BBC+95]. Their work specifically investigates the impact of of communication on performance in foraging teams. A motor schema approach with temporal sequencing is utilized (Figure 2.2 illustrates an example sequence from this work). The results show that foraging agents can cooperate without communicating. The investigation also found that simple communication provides an important performance advantage over no communication at all, but complex communication does not provide an additional improvement. The research is extended in this dissertation to include a more complex foraging task, several new strategies (including heterogeneous approaches) and learning.

In related research, Goldberg and Matarić have proposed a framework for investigating the relative merits of heterogeneous and homogeneous behavior in foraging tasks [GM97]. Like the research reported in this paper, their work focuses on mechanically identical, but behaviorally different agents. They propose *interference* as a metric for evaluating a foraging robot team. Interference refers to the situation where two robots attempt to occupy the same place at the same time; it is measured as the amount of time agents spend avoiding one another. Since interference may reduce the efficiency of a robot team, Goldberg suggests pack and caste arbitration as mechanisms for generating efficient behavior and reducing interference. In the pack scheme, each agent is arbitrarily assigned a place in the "pack hierarchy." Agents higher in the hierarchy are permitted to deliver attractors before the others. In the caste approach, only one agent completes the final delivery; the other robots leave their attractors on the boundary of a designated "home zone." The researcher's results indicate that interference per unit time is maximized in homogeneous foraging and minimized in pack foraging. In spite of the fact that interference is minimized in the heterogeneous pack systems, homogeneous systems perform best in terms of the number of pucks collected.

In separate research, Fontan and Matarić have investigated a territorial heterogeneous foraging strategy where the search area is equally divided between agents [FM97]. Robots hand off collected attractors from area to area, with the last agent completing delivery to the homebase. Their work indicates that performance degrades if the number of robots is increased beyond a certain maximum.

Drogoul investigates several homogeneous foraging strategies in simulation [DF94]. His research investigates the utility of laying "crumbs" as path markers for other agents. The idea was inspired by the technique of laying chemical trails to food sources utilized by many ant species [HW90]. Interestingly, the issue of agent-agent interference arises in Drogoul's work as well. In the most efficient "crumb-laying" foraging strategy, performance is reduced when the number of agents exceeds a particular mark. To address this, a "docker" behavioral strategy is explored. The docker robots are able to pass attractors from one to another while remaining in a fixed position. In robot simulations using this behavior, spontaneous chains of agents arise.

Instead of carrying attractors back to the base individually, they hand them from one to another in the chain. When resource-rich areas are discovered, performance is maximized in the docker strategy. The key drawback to this approach is the mechanical challenge of building agents able to accomplish such hand offs.

### 2.5.2 Learning robotic foraging

Matarić has investigated learning for multi-robot behavior-based teams in foraging tasks. Her work has focused on developing heuristic reinforcement functions for social learning [Mat94]. In one approach, the overall reinforcement, $R(t)$, for each robot is composed of separate components, $D, O$ and $V$. $D$ indicates progress towards the agent's present goal. $O$ provides a reinforcement if the present action is a repetition of another agent's behavior. $V$ is a measure of vicarious reinforcement; it follows the reinforcement provided to other agents. She tested this approach in a foraging task with a group of three robots. Results indicate that performance is best when the reinforcement function includes all three components. In fact the robots' behavior did not converge otherwise.

In another multi-agent learning investigation Matarić compares Q-learning with a heuristic learning strategy for foraging. The new strategy utilizes a "shaped" reinforcement function where agents are rewarded as they accomplish parts of the task. The heuristic approach is shown to perform significantly better than Q-learning and Matarić concludes that Q-learning is not appropriate for multiagent learning tasks.

The results reported in this dissertation contradict Matarić's conclusion regarding the suitability of Q-learning for multi-robot learning. In this research, multiagent teams using Q-learning converge to behaviors that perform as well as or better than human-coded approaches. This result holds in foraging, as well as soccer and cooperative movement tasks. Also, shaped reinforcement is shown to provide little or no advantage over the standard performance-based rewards used in most other reinforcement learning studies.

### 2.5.3 Robotic soccer

Robotic soccer is one of several task domains this research investigates. Soccer is a particularly good task for multiagent research because it includes cooperation between teammates, competition versus an opponent and unpredictable dynamic play.

In early robot soccer research, Sahota developed a system called Dynamite [Sah94]. The Dynamite test-bed utilized remotely driven cars controlled by an off-board computer. The computer was able to monitor the game through an overhead camera. He proposed *reactive deliberation* as a control scheme. In this architecture, a high level module (the Deliberator) selectively activates "action schemas" to be run at the lower level. The system did not include learning, but it was demonstrated to play soccer well. Reactive deliberation bears some resemblance to AuRA in that a higher level deliberation module selects schemas for execution by the lower level, but AuRA offers the possibility of activating and integrating multiple schemas simultaneously.

Recent interest has sparked more research in robot soccer. Kitano and Asada promote the Robot World Cup as a vehicle for multiagent research [KAK+97]. They have developed an internationally agreed upon set of rules for a game involving mobile robots and a separate simulation system using the same rules. Asada has additionally investigated learning individual skills (e.g. shooting) for robot soccer players.

### 2.5.4 Learning robotic soccer

Stone and Veloso have developed a multi-layered learning system for soccer [SV98]. In their approach, individual agents are taught lower-level skills first, using a neural-net technique. Higher-level behaviors are developed using decision trees. Although the mechanism is different (decision trees) the approach to training is similar to Lin's in that the lower level skills are developed first with higher-levels trained afterwards [Lin92].

Salustowicz, *et al* have investigated reinforcement learning in a simulated soccer task [SWS98]. Their research is focused on a comparison of PIPE and TD-Q learning. PIPE is genetic programming variant [Koz92], while TD-Q is based on the neural network approach introduced by Lin [Lin93] (note: TD-Q is distinct from Q-learning). The results indicate PIPE generates teams with better performance than those trained using TD-Q. The work is similar to the approach used in this research for training soccer agents, but with several important distinctions. In Salustowicz's approach the agents are implicitly homogeneous. All agents share the same policy, so it is impossible for heterogeneity to emerge. In contrast, in this work, each agent develops an individual policy that may or may not correspond to that of the other agents. Also, this research evaluates the impact of several competing reinforcement strategies (local and global) while Salustowicz's work utilizes global performance-based rewards for all training.

### 2.5.5 Robot formation

Formation behaviors in nature, like flocking and schooling, benefit the animals that use them in various ways. Each animal in a herd, for instance, benefits by minimizing its encounters with predators [Veh87]. By grouping, animals also combine their sensors to maximize the chance of detecting predators or to more efficiently forage for food. Studies of flocking and schooling show that these behaviors emerge as a combination of a desire to stay in the group and yet simultaneously keep a separation distance from other members of the group [CSB65]. Since groups of artificial agents could similarly benefit from formation tactics, robotics researchers and those in the artificial life community have drawn from these biological studies to develop formation behaviors for both simulated agents and robots.

Formation is important in mobile multiagent applications where sensor assets are limited. Formations allow individual team members to concentrate their sensors across a portion of the environment, while their partners cover the rest. Air Force fighter pilots for instance, direct their visual and radar search responsibilities depending on their position in a formation [For92]. Robotic scouts also benefit by directing their sensors in different areas to ensure full coverage [CGH96]. Formation is potentially applicable in many other domains such as search and rescue, agricultural coverage tasks, security patrols and so on.

In the behavior-based approach utilized in this research, formation maintenance is accomplished in two steps: first, a perceptual process, **detect-formation-position**, determines the robot's proper position in formation based on current environmental data; second, the motor process **maintain-formation**, generates motor commands to direct the robot toward the correct location. Each robot computes its proper position in the formation based on the locations of the other robots. Several motor schemas, **move-to-goal**, **avoid-static-obstacle**, **avoid-robot** and **maintain-formation** implement the overall behavior for a robot to move to a goal location while avoiding obstacles, collisions with other robots and remaining in formation. An additional background schema, **noise**, serves as a form of reactive "grease", dealing with some of the problems endemic to purely reactive navigational methods [Ark89].

In the most closely related approach, Parker simulates robots in a line-abreast formation navigating past waypoints to a final destination [Par93]. The agents are programmed using the layered subsumption architecture [Bro86]. Parker evaluates the benefits of varying degrees of global knowledge in terms of cumulative position error and time to complete the task. The approach includes a provision for obstacle avoidance, but performance in the presence of obstacles is not reported. Parker's results suggest that performance is improved when agents combine local control with information about the leader's path and the team's goal.

This research extends this earlier work by providing agents with the ability to learn formation behaviors. At this writing, the author knows of no other multi-robot formation research involving learning agents.

## 2.6 Social entropy theory

A precise definition of *diversity* in robot societies is important for this research. *social entropy* is proposed as an appropriate metric of diversity in robot systems. Details of social entropy in robot groups are provided in Chapter 5 and in [Bal97c, Bal97b]. Interestingly, sociologists have developed a similar (and eponymous) *social entropy theory* as a means of explaining and evaluating social structure in human groups [Bai90].[1]

Briefly, both human and robotic social entropy are based on information entropy, a measure of randomness in communication systems. Greater entropy indicates more randomness and disorder. Entropy in communication depends on the number of distinct symbols to be transmitted and the frequency of each symbol in a typical message. Similarly, social entropy depends on the number of distinct types of individuals in a society and their frequency of occurrence in the society. Both robotic and human measures of social entropy depend on a categorization of agents into groups based on differences between them.

The selection of an appropriate set of features or attributes on which to com-

[1] Although entropy was introduced as a tool in sociology as early as 1958, the application of entropy for the evaluation of robot systems is new and was developed independently.

pare individuals is a heatedly debated topic in the sociological entropy literature. The most frequently cited framework is [Bai90] by Bailey. Bailey employs a five-dimensional system of mutable variables that describe each person in a society. For each person, each variable has a particular value. People with similar attributes may be grouped together. Bailey's mutable variables are

- **I**, information: education, religious beliefs, political ideology.
- **L**, level of living: quality of life, income.
- **S**, space: location of residence.
- **T**, technology: level of technological skill.
- **O**, organization: position in organizational hierarchy.

The variables are referred to as "mutable" because an individual is able, and even likely, to change them through their life. For instance **S** is changed when someone relocates, **L** changes when a person get a raise and so on. People are also ascribed immutable characteristics like gender, time of birth (age), skin color, etc. Since in this research, the focus is on *behavioral* diversity in robots, the features for categorization are somewhat different. Agents are categorized on the basis of differences in behavior; the idea is compare their learned policies and group them according to similarities in their strategies.

As an example of how entropy might be employed for social analysis, consider how the spatial (**S**) distribution of Americans has shifted from rural areas to the cities over the last 50-100 years. When we were primarily a rural society, the value of **S** was likely to be different for nearly all citizens. As people moved to the cities, however, there was a greater and greater likelihood for many people to share the same or similar **S**. This shift has served to *decrease* the entropy of our country's spatial distribution, indicating that we have become more ordered, at least with respect to geography.

The use of entropy for similar purposes in sociology supports its use in robotics. It is important to note that sociologists hardly ever calculate a numerical value for the entropy of, say, the United States. Rather the idea is a framework for analysis. It provides a way for researchers to analyze social change and structure.

## 2.7 Discussion and summary

This chapter reviews the important existing work related to the dissertation. It also provides the reader with a background on the algorithms and techniques drawn from others and employed in this work. Key points:

- **Motor schema-based control** is employed as the robot behavioral programming platform [Ark89]. Motor schemas are grouped together to form behavioral assemblages. Assemblages are activated in an appropriate sequence to accomplish a task.
- **Q-learning**, a reinforcement learning technique, is used to train robots when to activate particular behaviors to accomplish a task [WD92].
- **Social entropy** is utilized as a quantitative measure of diversity in robot teams. The technique is also used in sociology for evaluating the structure of human society [Bai90].
- **Robotic tasks** including foraging, soccer and formation maintenance are explored in this research. The significant work of other researchers in these tasks is cited and reviewed.

The research in this dissertation differs from other work in several important respects. First, while other researchers are investigating performance in homogeneous and heterogeneous robot systems, here we are primarily concerned with the *origins* of heterogeneous and homogeneous behavior. The work is further distinguished by the fact that *learning* agents are the central investigative tool. No commitment is made in advance to any particular societal structure or arbitration mechanism. Instead, the robots develop their own societal solutions. This opens up the possibility that new forms of arbitration and cooperation may be discovered by the robots themselves. Finally, we are interested in measuring the diversity of the resulting society and utilize the metric of social entropy for that purpose [Bal97b].

# Chapter 3

# Methodology

A key contribution of this work is the idea that diversity should be evaluated as a *result* rather than an initial condition of multi-robot experiments. In earlier investigations researchers configured homogeneous and heterogeneous teams *a priori*, then evaluated their performance [FM97, GM97, Par94]. The latter approach is useful when investigating the impact of diversity on performance, but it does not provide for the study of diversity as an emergent property of agents interacting with their environment. Defining heterogeneity as an independent rather than dependent variable enables the examination of diversity from an ecological point of view. We can now ask questions like "how does the number of agents impact diversity?" or "how does reward impact diversity in learning teams." These issues are the core of this work.

Along with the opportunity for new kinds of research, the paradigm of diversity as a measured outcome presents new challenges. First, a quantitative measure of diversity is necessary. Second, a methodology for employing the metric in the experimental exploration of diversity in multi-robot systems must be developed. The issue of a quantitative metric is dealt with in Chapter 5. The purpose of this chapter is to introduce a methodology for investigating diversity in multi-robot systems and to explain how it was applied experimentally.

## 3.1 Overview

Principled research in any field requires adherence to a methodological framework. Over the last decade the Mobile Robot Laboratory at Georgia Tech has evolved and refined a successful approach to behavior-based robot design and implementation. Key components of the method are the use of simulation for experimentation and behavioral prototyping along with verification of the results on mobile robots.

The framework was extended significantly in this research. First, a formal view of multi-robot task is adopted; multi-robot tasks are classified according to how performance is measured in them. This enables a principled description and exploration of the multi-robot task space. Second, a classification of reward functions is employed. This classification defines an experimental space for investigating the impact of reward on multi-robot systems. To support these experiments, motor schema control and reinforcement learning are integrated using a new object-oriented system for behavioral specification. Finally, new evaluation metrics necessary for the measurement of diversity in multi-robot teams were developed and employed in the analysis of experimental data.

The design and implementation of multi-robot systems and their experimental evaluation is carried out in the following steps:

1. **Task and performance metric specification:** This step defines performance, one of the dependent variables of experimentation.

2. **Behavioral design:** In this phase, a library of behaviors are developed for solving the task. Both hand-coded and learning systems are built using the behavioral components.

3. **Reinforcement function specification:** A goal of the research is to explore how different reinforcement functions impact performance and diversity in learning systems. In each task domain, several reinforcement functions are employed, with primary focus on the comparison of local and global rewards.

4. **Simulation:** The behaviors and learning systems are prototyped and tested in simulation.

5. **Implementation on mobile robots:** Performance of the simulated system is validated on mobile robots. If inconsistencies are discovered the simulation environment is refined to more closely approximate mobile robot performance.

6. **Data collection:** Multiple runs (thousands, usually) are conducted in simulation, and when possible, on mobile robots. The experimental space is explored by varying the independent variables (e.g. number of robots and/or the reward function).

7. **Analysis:** The data are analyzed using the performance metric the measures of diversity presented in Chapter 5.

The remainder of this chapter describes the methodology in more detail and provides a high-level synopsis of the thesis experiments.

## 3.2 Task and performance specification

The research was conducted using the same methodology in each of three multi-robot task domains. Within each domain, quantitative performance data were gathered to determine the utility of various reinforcement functions and their impact on the robots' societal structure.

In systems using reinforcement learning the best possible reward function is the performance metric itself. Following this philosophy, the task for a robot team is to maximize the performance metric. The tasks investigated in this work and their performance metrics include

- **multi-foraging:** earlier multi-robot research defined the foraging task for a robot as "find all attractor objects and deliver them to a location (or homebase)" [BBC+95, ABN93, Mat92]. In multi-foraging, however, the task is extended to include coded attractors and bases. Individual attractors must be delivered to a base of the same color: red attractors to red bases, blue attractors to blue bases and so on. Do agents diversify by specializing in the collection of one type of attractor or the other? Performance in this task is defined as the number of attractors collected and properly delivered in a set time period.

- **robotic soccer:** the task is to propel a ball by bumping and kicking to a goal. The problem is complicated by an opposing team trying to do the same thing in the opposite direction. The game is interesting for multiagent robotics research because it is familiar, reasonable for implementation on mobile robots, and offers opportunities for specialization. Do learning robotic agents specialize as humans do

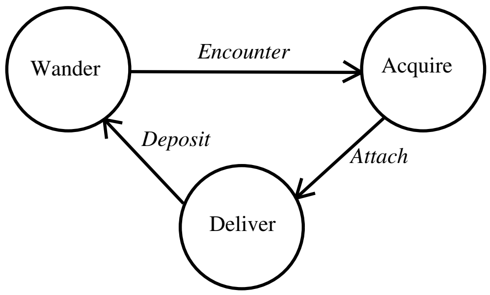



**Figure 3.1**: The forage FSA.

(goalie, forward, etc.)? The task has been explored by several researchers [KAK+97, NMH96] and has been the focus of international competitions in 1997 and 1998. Performance in soccer is defined as the difference in score at the end of a game.

- **cooperative movement:** The task is for a robot team to negotiate cluttered terrain using the most efficient formation behavior (or none at all). Is there an advantage to traveling in a group? Earlier research by the author and Ron Arkin has shown that some types of formation have an advantage over the others [BA95b]. The primary performance metric in this task is time to navigate across a specified distance.

The relationship between task and performance is covered in depth in Chapter 4.

## 3.3 Behavioral design

A schema-based reactive control system is used for robot programming. In this approach the agent is provided several pre-programmed behavioral assemblages that correspond to steps in achieving the task. As an example, for a foraging robot we might develop *wander*, *acquire* and *deliver* behaviors for steps in the task [AM94]. Binary perceptual features (also referred to as perceptual triggers) are used to sequence the robot through the behaviors to complete the task. Selection of the appropriate behavior(s) given the situation may be hand-coded or discovered by the robot through reinforcement learning. In this work, both hand-coded and learning systems are developed and evaluated side-by-side.

To ensure a fair comparison between the various hand-coded and learning systems, a fixed repertoire of behaviors are developed for each task domain. In each case, the repertoire is suitable for building behaviorally homogeneous teams as well as heterogeneous strategies. Hand-coded agents proceed deliberately from behavior to behavior as they accomplish the task while the learning agents must discover which behavior to activate when.

### 3.3.1 Example: foraging

To illustrate how behaviors are developed and coded, consider the programming of a foraging robot. In this example, a robot can be in one of three behavioral states: *wander*, *acquire* and *deliver*. The robot begins in the *wander* state. If there are no attractors within the robot's field of view, the robot remains in *wander* until one is encountered. When an attractor is encountered, a transition to the *acquire* state is triggered. While in the *acquire* state, the robot moves towards the attractor and when it is sufficiently close, grasps it. The last state, *deliver*, is triggered when the robot grasps the attractor. While in the *deliver* state the robot carries the attractor back to home base. Upon reaching home base, the robot deposits the attractor and reverts back to the *wander* state. An FSA summarizing this behavioral configuration is presented in Figure 3.1. The behavioral states are composed of more primitive behaviors (motor schemas) as follows:

- *wander:* move randomly about the environment in search of attractors. Upon encountering an attractor, the pre-programmed agents automatically transition to an appropriate *acquire* behavior. Learning systems, in contrast, discover an appropriate follow-on behavior on their own. Motor schemas active in the *wander* assemblage are
  - **noise:** high gain, moderate persistence to cover a wide area of the environment.
  - **avoid_obstacles:** gain sufficiently high to avoid collisions.
  - **avoid_robots:** high gain to encourage the robots to distribute about the search area.
- *acquire:* move towards the closest visible attractor. When close enough to grasp the attractor, hand-coded agents close their gripper and transition to the *deliver* behavior. Learning agents must learn which follow-on behavior to activate. Motor schemas activated in this assemblage include
  - **noise:** low gain, to deal with local minima endemic to potential field approaches.
  - **avoid_obstacles:** gain sufficiently high to avoid collisions.
  - **avoid_robots:** gain sufficiently high to avoid collisions.
  - **move_to_attractor:** high gain to move the agent to the attractor.
- *deliver:* move towards the delivery area. When close enough to deposit the attractor in the delivery area, hand-coded agents open their gripper and transition to the *wander* assemblage.

- **noise:** low gain, to deal with local minima endemic to potential fields approaches.
- **avoid_obstacles:** gain sufficiently high to avoid collisions.
- **avoid_robots:** gain sufficiently high to avoid collisions.
- **move_to_homebase:** high gain to move the agent to the delivery area.

In this simplified example the robots select from only three behavioral assemblages, in actual foraging experiments the robots are provided six (see Chapter 6). The next section shows how these behaviors can be implemented in Clay, the behavioral coding used in this research.

### 3.3.2 Configuring behavior with Clay

Robot behaviors are implemented using Clay, a library of primitive behaviors and coordination mechanisms coded in Java [Bal97a]. Clay is a component of JavaBots, a new system for simulation and control of multi-robot teams. Clay is the first behavioral configuration tool that integrates motor schema-based control and reinforcement learning. The name "Clay" was chosen for its connotations of reconfigurability and ease of use. Robots utilizing Clay benefit from the real-time performance of motor schemas in continuous and dynamic environments and adaptive reinforcement learning. Clay coordinates assemblages (groups of motor schemas) using embedded reinforcement learning modules. The coordination modules activate specific assemblages based on the presently perceived situation. Learning occurs as the robot selects assemblages and samples a reinforcement signal over time.

Clay is similar to earlier approaches integrating reinforcement learning and behavior-based control [MC92, Lin93, Mat94], but it differs in several important aspects: First, behavioral expression in Clay is fully recursive: there is no limit to the number of levels in a behavioral hierarchy. Second, Clay's primitive, the motor schema, provides a rich repertoire for behavioral design [AB97]. Motor schemas take full advantage of continuous sensor values and can generate an infinite range of actuator output; most of the other approaches only select from a discrete list of actions [SSR98]. Finally, while experiments with Clay have so far only explored learning at one level the designer is free to introduce learning at any level in the behavioral hierarchy. In previous research Georgia Tech's Mobile Robot Laboratory developed a system called *MissionLab* to support the design and test of sequenced behaviors on robots and in simulation [MCA95]. *MissionLab* includes a set of tools for recursively expressing sequenced behaviors. Like *Missionlab*, Clay provides for recursive expression of behavior, but it adds learning coordination operators and an object-oriented syntax.

The basic building block in Clay is a **node**. There are two important phases in a node's life: intialization and runtime. Nodes have only two methods, corresponding to these phases: the constructor, used for initialization; and `Value()`, called repeatedly at runtime. The object-oriented approach provides for a direct expression of schema instantiation and the embedding of perceptual schemas in motor schemas. The embedding is specified at initialization time using the node's constructor. Here is an example of how one node is embedded in another:

```
PS_OBS = new Obstacles(abstract_robot);
MS_AVOID_OBSTACLES  = new Avoid(2.0, 1.0, PS_OBS);
```

In this case, a perceptual schema for detecting obstacles (`PS_OBS`) is embedded in an `Avoid` node. The resulting motor process `MS_AVOID_OBSTACLES`, will draw the robot away from the perceived obstacles. The `PS` and `MS` prefixes are used to help readers and programmers distinguish between perceptual and motor schemas in the code.

Note that the embedding provides for code re-use. We could, for instance, avoid robots instead of obstacles by embedding a `PS_ROBOTS` versus `PS_OBS` in the `Avoid` node. It is also possible to re-use instantiated nodes by embedding them in several other nodes. In this next example, `PS_OBS` is imbedded in an `MS_AVOID_OBSTACLE` node and a `MS_SWIRL_OBSTACLE` node:

```
PS_OBS = new Obstacles(abstract_robot);
MS_AVOID_OBSTACLES  = new Avoid(2.0, 1.0, PS_OBS);
MS_SWIRL_OBSTACLES  = new Swirl(2.0, 1.0, PS_OBS, heading);
```

Nodes are combined by embedding them in a blending node. `StaticWeightedSum` is an example blending node class. This type node is used for the "sum and normalize" step of schema and assemblage combining. It takes an array of nodes and an array of weights as input at configuration time. At runtime, it multiplies the output of each embedded node by the associated weight or gain, then sums them. The following statements generate a new node, `AS_AVOID_N_SWIRL`, that is the average of its two embedded nodes:

```
AS_AVOID_N_SWIRL = new StaticWeightedSum();
AS_AVOID_N_SWIRL.embedded[0]   = MS_AVOID_OBSTACLES;
AS_AVOID_N_SWIRL.weights[0]    = 0.5;
AS_AVOID_N_SWIRL.embedded[1]   = MS_SWIRL_OBSTACLES;
AS_AVOID_N_SWIRL.weights[1]    = 0.5;
```

Figures 3.2 through 3.4 show how foraging behaviors may be coded in Clay. In this example a sequential (non-learning) system is configured. First, perceptual schemas and the motor schemas they are embedded within are declared (Figure 3.2). Next (Figure 3.3) the behaviors are grouped into assemblages using a weighted-sum operator. Finally, a high-level behavioral sequence is defined by coding it as an FSA (Figure 3.4; also shown graphically in Figure 3.1). Transitions between the behavioral states are triggered when appropriate perceptual features become true.

The code in Figures 3.2 through 3.4 specifies a multi-level hierarchical behavioral system. The hierarchy is presented graphically in Figure 3.5. The lowest level perceptual features are declared first (e.g. `PS_HOMEBASE`), then embedded in motor schemas (e.g. `MS_MOVE_TO_HOMEBASE`) that are in turn embedded in behavioral assemblages (e.g. `AS_DELIVER`). At the assemblage level, behaviors are combined cooperatively using a weighted sum operator. At the next level up, assemblages are selected in sequence using an FSA. The highest level behavior, `FORAGE`, is at the top of a four-layer recursive tree. If a designer was interested in building a more complicated agent with foraging as one of its several capabilities, `FORAGE` could be included as just another assemblage for integration at the next level up.

Control systems coded in Clay follow a perception/action cycle where each cycle is referred to as a timestep or movement cycle. For each cycle, computation begins at the top (e.g. `FORAGE`) and continues recursively downward through the configured assemblages and schemas. A potential difficulty for hierarchically specified behavioral systems is that as a behavioral configuration grows more complex, run time computational demands increase dramatically. Clay avoids the problem by only executing activated assemblages and schemas. Computational demands are also reduced when the designer re-uses schemas in a configuration (as `PS_ATTRACTORS` is re-used above). A synchronization technique ensures a schema's output is only computed once per movement cycle.

Reinforcement learning can be incorporated into a behavioral configuration using an additional coordination operator, `CoordinateLearner`. `CoordinateLearner` is

```
//=== Perceptual schemas
	// type of object in gripper (-1 if nothing)
	PS_IN_GRIPPER = new InGripper(abstract_robot);

	// sonar readings
	PS_OBS = new Obstacles(abstract_robot);

	// other robots
	PS_ROBOTS = new Teammates(abstract_robot);

	// location of the homebase
	PS_HOMEBASE = new GlobalToEgo(abstract_robot, PS_HOMEBASE_GLOBAL);

	// the list of visible attractors
	PS_ATTRACTORS = new VisualObjects(0,abstract_robot);

	// the closest attractor
	PS_CLOSEST_ATTRACTOR = new Closest(PS_ATTRACTORS);

//=== Motor schemas
	// sphere of influence 1.5m, safety radius 0.1m, objects to avoid
	MS_AVOID_OBSTACLES = new Avoid(1.5, RADIUS + 0.1, PS_OBS);

	// sphere of influence 2.5m, safety radius 0.1m, objects to avoid
	MS_AVOID_ROBOTS = new Avoid(2.5, RADIUS + 0.1, PS_ROBOTS);

	// generate a new random direction every five seconds
	MS_NOISE_VECTOR = new Noise(5,seed);

	// attraction decreases linearly inside 0.4m constant beyond that
	MS_MOVE_TO_HOMEBASE = new LinearAttraction(0.4, 0.0, PS_HOMEBASE);

	// attraction decreases linearly inside 0.4m constant beyond that
	MS_MOVE_TO_ATTRACTOR = new LinearAttraction(0.4, 0.0, PS_CLOSEST_ATTRACTOR);
```

**Figure 3.2**: Clay source code for the declaration of perceptual and motor schemas employed in the foraging task. For clarity, a `PS` prefix is used in the declaration of a perceptual schema, while `MS` is used for motor schemas. Note how the syntax supports the embedding of perceptual schemas in motor schema declarations. For example, the `PS_HOMEBASE` perception is embedded in the `MS_MOVE_TO_HOMEBASE` motor schema.

```
//=== Wander assemblage
       AS_WANDER = new StaticWeightedSum();
       AS_WANDER.weights[0]  = 1.0;
       AS_WANDER.embedded[0] = MS_NOISE_VECTOR;
       AS_WANDER.weights[1]  = 0.8;
       AS_WANDER.embedded[1] = MS_AVOID_OBSTACLES;
       AS_WANDER.weights[2]  = 0.8;
       AS_WANDER.embedded[2] = MS_AVOID_ROBOTS;

//=== Acquire assemblage
       AS_ACQUIRE = new StaticWeightedSum();
       AS_ACQUIRE.weights[0]  = 0.2;
       AS_ACQUIRE.embedded[0] = MS_NOISE_VECTOR;
       AS_ACQUIRE.weights[1]  = 0.8;
       AS_ACQUIRE.embedded[1] = MS_AVOID_OBSTACLES;
       AS_ACQUIRE.weights[2]  = 0.8;
       AS_ACQUIRE.embedded[2] = MS_AVOID_ROBOTS;
       AS_ACQUIRE.weights[3]  = 1.0;
       AS_ACQUIRE.embedded[3] = MS_MOVE_TO_ATTRACTOR;

//=== Deliver assemblage
       AS_DELIVER = new StaticWeightedSum();
       AS_DELIVER.weights[0]  = 0.2;
       AS_DELIVER.embedded[0] = MS_NOISE_VECTOR;
       AS_DELIVER.weights[1]  = 0.8;
       AS_DELIVER.embedded[1] = MS_AVOID_OBSTACLES;
       AS_DELIVER.weights[2]  = 0.8;
       AS_DELIVER.embedded[2] = MS_AVOID_ROBOTS;
       AS_DELIVER.weights[3]  = 1.0;
       AS_DELIVER.embedded[3] = MS_MOVE_TO_HOMEBASE;
```

**Figure 3.3**: Source code for the declaration of behavioral assemblages used in foraging. The `AS` prefix indicates the declaration of a behavioral assemblage. Motor schemas are cooperatively combined using the `StaticWeightedSum` operator. Note how the syntax supports reuse of motor schemas (e.g. `MS_AVOID_OBSTACLES`) in several assemblages.

```
//=== Perceptual Features
      // true if an attractor is visible
      PF_ATTRACTOR_VISIBLE = new NonZero(PS_CLOSEST_ATTRACTOR);

      // true if an attractor has been grasped
      PF_ATTRACTOR_IN_GRIPPER = new Equal(0, PS_IN_GRIPPER);

      // true if close enough to homebase to deposit the attractor
      PF_CLOSE_TO_HOMEBASE = new Close(PS_HOMEBASE);

//=== Behavioral state machine
      STATE_MACHINE = new FSA(); // declare the state machine
      STATE_MACHINE.state = 0;        // set initial state to 0 (wander)

      // if an attractor is visible, transition to state 1 (acquire)
      STATE_MACHINE.triggers[0][0] = PF_ATTRACTOR_VISIBLE;
      STATE_MACHINE.follow_on[0][0]= 1;

      // if an attractor in gripper, transition to state 2 (deliver)
      STATE_MACHINE.triggers[1][0] = PF_ATTRACTOR_IN_GRIPPER;
      STATE_MACHINE.follow_on[1][0]= 2;

      // if at homebase, go back to state 0 (wander)
      STATE_MACHINE.triggers[2][0] = PF_CLOSE_TO_HOMEBASE;
      STATE_MACHINE.follow_on[2][0]= 2;

//=== Select assemblage based on behavioral state
      FORAGE = new Select(STATE_MACHINE); // declare selector

      // activate appropriate behavior based on state
      FORAGE.embedded[0]    = AS_WANDER;
      FORAGE.embedded[1]    = AS_ACQUIRE;
      FORAGE.embedded[2]    = AS_DELIVER;
```

**Figure 3.4**: A behavioral sequence is configured using the `FSA` operator in Clay. A `PF` prefix indicates the declared object is a perceptual feature. Transitions between behavioral assemblages are triggered by perceptual features.

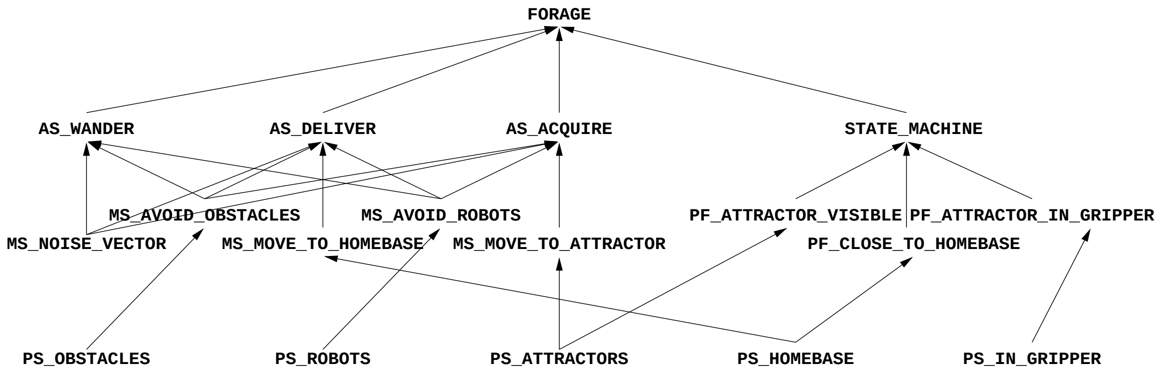



**Figure 3.5**: A graphical representation of the hierarchically configured FORAGE behavior. Prefixes are used to indicate perceptual schemas (PS), motor schemas (MS), behavioral assemblages (AS) and perceptual features (PF).

"plug compatible" with the FSA operator (see the bottom of Figure 3.4 for an example use of FSA). CoordinateLearner is able to learn to select appropriate assemblages rather than following a fixed sequence. At configuration time, an instantiation of CoordinateLearner is provided an embedded reward schema that it uses for learning over time. Any learning module that conforms to Clay's Application Programmer's Interface (API) can be integrated into a behavioral configuration. Q-learning was used in this research.

## 3.4 Reinforcement function specification

The next step in learning multi-robot team development is the specification of appropriate reinforcement (or reward) functions. One focus of this research is to find out if and how the choice of reward function impacts performance and diversity in different tasks. In the experiments described in later chapters, performance is first evaluated in learning teams using a global function. Global functions are those where the entire team of robots is provided the same reward signal at once. Performance with the global function serves as a baseline for comparison with teams using other functions. In addition to the global function robot teams were also trained using a local reward function. Learning agents using a local reinforcement function are rewarded for their individual performance rather than the overall performance of the team. Even though local rewards are targeted to the individual robot, they may, in some cases require global information for implementation. For more information on reward functions see the classification of reinforcement functions in Chapter 4.

The reward functions used experimentally are listed below (they are also described in corresponding chapters later in the dissertation). Each reward function is defined at timestep $t$ based on events occurring at timestep $t-1$ as follows:

- **For multi-robot foraging** three functions were explored; performance-based global, performance-based local and heuristic functions:

$$
\begin{aligned}
R_{\text{global}}(t) &= \begin{cases} 1 & \text{if any agent delivered an attractor at time } t-1 \\ -1 & \text{otherwise} \end{cases} \\
R_{\text{local}}(t) &= \begin{cases} 1 & \text{if the agent delivered an attractor at time } t-1 \\ -1 & \text{otherwise} \end{cases} \\
R_{\text{shaped}}(t) &= R_{\text{event}}(t) + R_{\text{intruder}}(t) + R_{\text{progress}}(t)
\end{aligned}
$$

  The $R_{\text{event}}$, $R_{\text{intruder}}$ and $R_{\text{progress}}$ components of $R_{\text{shaped}}$ encapsulate separate heuristic components of the overall reward. See Chapter 6 for more information on this reward function.

- **For robot soccer** three performance-based functions were examined:

$$
\begin{aligned}
R_{\text{global}}(t) &= \begin{cases} 1 & \text{if the team scored at } t-1, \\ -1 & \text{if the opponent score at } t-1, \\ 0 & \text{otherwise.} \end{cases} \\
R_{\text{local}}(t) &= \begin{cases} 1 & \text{if the agent was closest to the ball} \\ & \text{when its team scores,} \\ -1 & \text{if the agent was closest to the ball} \\ & \text{when the opposing team scores,} \\ 0 & \text{otherwise.} \end{cases} \\
R_{\text{touch}}(t) &= \begin{cases} \gamma^{t_{\text{touch}}} & \text{when the team scores,} \\ -\gamma^{t_{\text{touch}}} & \text{when the opponent scores,} \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

  Even though implementation of $R_{\text{local}}$ requires global information the reward is referred to as local because the reinforcement based on the individual's performance. In $R_{\text{touch}}$ the variable $t_{\text{touch}}$ is time in milliseconds since the agent last touched the ball. The parameter $\gamma$ is set to values between 0 and 1. When $\gamma = 1$ $R_{\text{touch}}$ is equivalent to $R_{\text{global}}$. As $\gamma$ is reduced towards 0, the reward becomes increasingly agent-centered.

- **For cooperative movement** two reward functions were tested:

$$
\begin{aligned}
R_{\text{global}} &= -\text{elapsed time for all robots to cross the field} \\
R_{\text{local}} &= -\text{elapsed time for the individual robot to cross the field}
\end{aligned}
$$

These functions reward agents for moving across the field as quickly as possible. In this task only a single reward is provided at the end of the run.

## 3.5 Simulation

Many autonomous robot designers use simulation as a tool to speed behavioral development. Once behaviors work well in simulation they are moved to mobile robots for further debugging and verification. Simulation plays a valuable role in experimentation. In the time it takes to complete a few runs on mobile robots, simulation systems can complete thousands, perhaps millions of multiagent trials. Simulation is also important for use in learning systems where many thousands of trials may be required for an agent to learn a behavior or strategy. It would be impossible to conduct such learning on a mobile robot in a timely manner. Finally, simulation enables the exploration of a larger experimental space than would otherwise be possible.

The JavaBots system is utilized for simulation and mobile robot experimentation [Bal98, Bal]. Behaviors coded in JavaBots may be run in simulation, and without modification, on Nomadic Technologies' Nomad 150 mobile robots (the ISR Pebbles robot is also supported). The bulk of statistical results in this work were gathered by running robot behaviors in thousands of simulation trials.

In the foraging and cooperative movement investigations, each robot is a kinematically holonomic vehicle (a simulated Nomad 150) controlled by a behavioral system coded in Clay. Simulated motor and sensor capabilities are based on performance of the physical robots. The robots can detect hazards with sonar out to a range of nine meters. Attractors can be detected visually out to three meters across a 90 degree field of view. In soccer experiments, dimensions and dynamics are based on RoboCup F-180 class robots. Speed and turning rates mimic the performance of mobile robots built for RoboCup competition [Sto98].

The external environment perceived by robot control systems in simulation is described in a file read by the simulation application at run time. Multiple robots may be distributed about a "playing field" along with obstacles, opponent robots and attractor objects. Even though some control systems utilize a pseudo-random number generator (e.g. `NOISE`), determinism is supported with a `seed` statement in the description file syntax. Time is measured in simulated seconds. Since reactive control systems are very fast, several thousand control cycles are completed each second. Simulations proceed faster than real time with each control cycle fixed at 200 milliseconds (simulation time).

## 3.6 Implementation on mobile robots

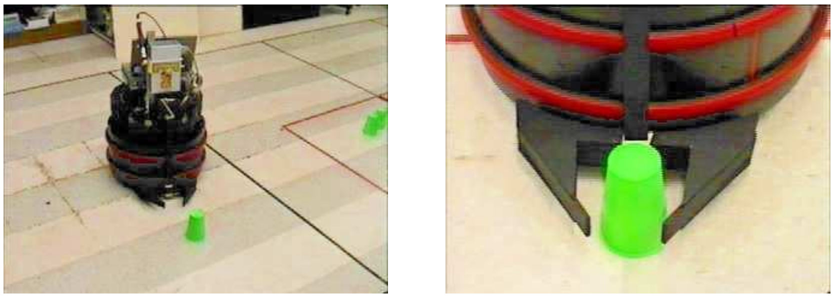


**Figure 3.6**: Nomad 150 robotic platform. Close-up view of the passive gripper (right).

Simulated behavior should be validated with experiments on mobile robots. When discrepancies are found between behavior in simulation and on real robots, the simulator must be revised to reflect real world performance (such discrepancies are most often due to inaccurate sensor or actuator modeling). The multi-robot foraging behaviors developed in this research were prototyped in simulation and verified on Nomadic Technologies' Nomad 150 robots. Since the behaviors are implemented in JavaBots, they can run in simulation and on hardware without revision.

Learning systems are developed and evaluated on mobile robots in the following steps:

1. initialize control system on mobile robot(s) with a random policy;
2. evaluate performance of the initial policy;
3. transfer policy to simulation system;
4. train in simulation;
5. transfer policy back to mobile robot(s);
6. evaluate performance on mobile robot(s) after learning.

These steps are covered in more detail on the chapter concerning multi-robot foraging (Chapter 6).

### 3.6.1 Hardware platform

The Nomad 150 is a three-wheeled commercially available kinematically holonomic vehicle (Figure 3.6, left). Nomad 150s are equipped with a separately steerable turret, 16 ultrasonic range sensors and a ring of rubber bump sensors. The robots were modified at Georgia Tech to add real-time vision and grippers.

Each robot's vision system is able to segment video images into blobs according to color 30 times per second. The location of objects detected as blobs in the image are determined in JavaBots using a lookup table that maps image coordinates to real-world locations. The lookup table is computed before experimental runs by moving a robot to known positions while it tracks a brightly painted object at the origin. The robot's field of view was expanded to approximately 160 degrees by mounting a security door lens (peep-hole) to the robot's video camera. Unfortunately, only the central 90 degrees are usable because objects at the periphery are quite small in the image. The two types of attractor object used in foraging experiments are painted different colors (fluorescent red and green) to enable the robots to tell them apart. Example images generated by the vision system are presented in Figure 3.7.

The robots were equipped with hobby servo-actuated grippers that enable them to grasp and lift attractors. The active grippers work well, but frequently require time-consuming repair. For later experiments the active grippers were replaced with passive devices (Figure 3.6, right). The passive gripper is designed so that once a robot has "captured" an attractor object it will remain under the robot's control as long as the robot does not move backwards. Robots then drop attractors by moving in reverse.

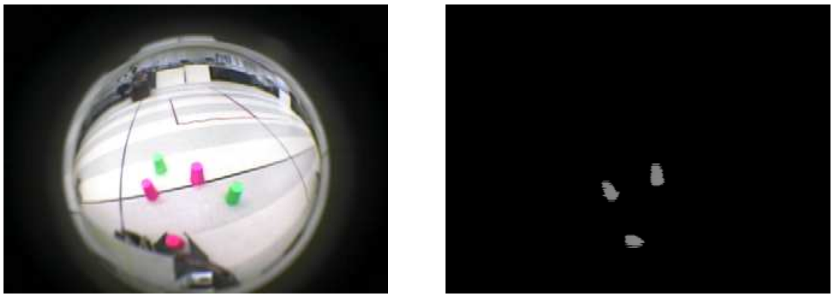


**Figure 3.7**: Mobile robot's eye view of several attractor objects (left). After processing to segment out red objects (right).

## 3.7 Data collection and analysis

Within each task domain an experimental space was explored by varying one or more independent variables and evaluating the resulting systems. Statistical significance was ensured by simulation in multiple "worlds" initialized with distinct random number seeds. Within each task the question of how the choice of reward impacts multi-robot teams was explored by varying the reward function. Additionally, the number of robots was varied from 1 to 8 for simulation experiments in robot foraging, 1 and 2 agents were used mobile robot experiments. The dependent variables in these experiments are

- **performance:** How well does the multiagent system perform in the task?
- **learning rate:** How quickly does the team converge to stable behavior?
- **diversity:** Once the system has converged to stable behavior, what is the degree of diversity in the team?

Performance is defined separately for each task (e.g. score difference in soccer). Since learning agents don't always perform well initially, it is also important to consider their learning rate. Some systems require a long time to converge to optimal or near-optimal solutions while others converge to sufficient, but sub-optimal solutions quickly. So, in addition to measuring against the performance metric, the time required for a team to converge to stable behavior is recorded as well.

The experimental space is summarized below:

- **in multi-robot foraging**
  - **independent variables:**
    - strategy: 3 hand-coded strategies and 3 learning strategies.
    - number of robots: 1 to 8 for each strategy.
  - **worlds:** 5 worlds for each combination of independent variable.
  - **runs in each world:** 100 in hand-coded systems, 300 in learning.
  - **total trials:** 48,000 in simulation, 22 on mobile robots.
- **in robot soccer**
  - **independent variables:**
    - reward function: 3, including $R_{\text{local}}$, $R_{\text{global}}$ and $R_{\text{touch}}$.
    - $\gamma$ for $R_{\text{touch}}$ was varied from 0.1 to 1.0 in steps of 0.1.

- **worlds:** 10 worlds for each combination of independent variable configuration.
- **runs in each world:** 100 for $R_{\text{local}}$ and $R_{\text{global}}$, 200 for each value of $\gamma$ in $R_{\text{touch}}$ runs.
- **simulation trials:** 24,000 in simulation.

- **in cooperative movement**
  - **independent variables:**
    * strategy: 3 hand-coded and 2 learning strategies.
  - **worlds:** 8 worlds for each strategy independent variable configuration.
  - **runs in each world:** 100.
  - **total trials:** 4,000 in simulation.

## 3.8 Discussion and summary

An important contribution of this work is the idea that diversity should be measured as an experimental outcome rather than specified as an initial condition. The view of diversity as a dependent variable enables us to investigate new questions about multi-robot systems, including one focus of this work: the impact of reward structure on diversity in learning multi-robot systems.

The investigation of these questions lead to the development of a new methodology for multi-robot experimentation Steps in the methodology include

1. **task and performance metric specification,**
2. **reinforcement function specification,**
3. **behavioral design,**
4. **simulation,**
5. **implementation on mobile robots,**
6. **data collection and**
7. **analysis.**

Several components of this framework are new. First, a formal view of multi-robot task is adopted; this enables a principled description and exploration of the multi-robot task space. Second, a classification of reward functions is employed. This classification defines an experimental space for investigating the impact of reward on multi-robot systems. Also, to support these experiments, motor schema control and reinforcement learning are integrated using a new object-oriented system for behavioral specification. Finally, new evaluation metrics necessary for the measurement of diversity in multi-robot teams were developed and employed in the analysis of experimental data.

# Chapter 4

# Task and Reward

An objective of this research is the development of a methodology supporting investigation of the impact of reinforcement and task on robot team behavioral diversity and performance. This dissertation examines the relationships experimentally by varying the the independent variables (reinforcement and task) and evaluating their impact on the dependent variables (diversity, performance and learning rate). Characterization of the task and reward structure is crucial for establishing the experimental space of this research.

Task characterization is important because it helps answer questions regarding how the same type of reinforcement can lead to different performance and and diversity levels in different tasks. Without answering "how are soccer and foraging different?" for instance, we can't answer "why is diversity good in soccer but bad in foraging?"

A taxonomy of reward structure is also important. Research presented in later chapters shows that performance and diversity in a learning team depend on the form of reinforcement used to train the robots. The results indicate that there are tradeoffs to consider in the selection of a reward function; for instance some functions provide quicker learning but slightly poorer performance. Without a characterization of the differences between reward functions, it would be impossible for a robot systems designer to consider these tradeoffs intelligently.

At present no taxonomies of task or reinforcement exist. To address this crucial gap a new system for characterizing multi-robot tasks and a taxonomy of reinforcement functions for multi-robot teams are presented. As well as aiding this investigation, these classifications are potentially useful for other researchers investigating multiagent robotics.

## 4.1 Characterization of multi-robot tasks

The reinforcement learning literature considers the task and the performance metric as one and the same. That view is adopted here; a robot team's mission is to maximize performance over time. But there are other components of a task as well. In addition to the performance metric, a task is further defined by the environment and robotic platform. Since the investigation focuses on links between task and diversity, we seek a classification identifying aspects of a task that may be correlated with a requirement for cooperation and/or diversity in a robot team.

The approach is to examine the task performance metric, task environment and robotic platform for constraints and features that distinguish various tasks from one another. The focus is on tasks involving the movement of robots or the movement of objects by robots. In a manner similar to Dudek's taxonomy of multi-robot systems [DJW93], a set of descriptive tags that identify salient features of a task are proposed. Although similar in style to Dudek's formulation, this classification focuses on *task* rather than robot.

The descriptive tags are summarized in Table 4.1 and explained in detail in the following subsections. Several example tasks are examined and classified using the descriptors at the end of this section.

**Table 4.1**: Summary of terms characterizing Multi-robot tasks.

| Descriptor | Meaning |
|---|---|
| **Time** | |
| TIME_LIM | fixed time task |
| TIME_MIN | minimum time task |
| TIME_UNLIM | unlimited time task |
| SYNC | synchronization required |
| **Subject of Action** | |
| OBJECT_BASED | movement/placement of objects is important |
| ROBOT_BASED | movement/placement of robots is important |
| **Limited Resources** | |
| RESOURCE_LIM | limited external resources |
| ENERGY_MIN | minimum energy task |
| COMP_INT | team members compete with each other |
| COMP_EXT | team competes with external agencies |
| **Movement** | |
| CONVERGENCE | multiple robots converge to same position |
| COVERAGE | multiple agents spread apart |
| MOVEMENT_TO | movement to a position |
| MOVEMENT_WHILE | movement while maintaining position |
| **Platform** | |
| SINGLE_AGENT | a single agent can perform task |
| MULTI_AGENT | multiple agents are required |
| DISPERSED | agents must be geographically dispersed |
| SENSOR_COMPLETE | can sense all relevant features |
| SENSOR_LIM | world is partially observable |
| COMM | communication is required |

### 4.1.1 Time

Most task definitions include a time constraint. The AAAI-98 "Find Life on Mars" task for instance, required robots to collect as many objects as possible in 10 minutes. For some tasks a robot's performance is evaluated by how long it takes to complete the task. This section considers various task time constraints.

Evaluations of robot systems are typically conducted in discrete experimental trials. `TIME_LIM` tasks are those where each trial runs over a fixed length of time, while `TIME_MIN` refers to tasks where performance is measured as the time required to complete a task. An example `TIME_LIM` task is "collect as many attractors as possible in 10 minutes." An example `TIME_MIN` task would be "collect five attractors as quickly as possible."

Some tasks do not have a time constraint; they are referred to as `TIME_UNLIM`. `TIME_UNLIM` tasks are either carried out indefinitely, or terminate on non-time-related criteria. An example indefinite time task might be "patrol the building for burglars" or "balance the pole." The soccer experiments explored later in this work (Chapter 7) are `TIME_UNLIM` since they do not end on a time limit, but rather when a total of number of goals are scored.

In addition to elapsed time constraints, some tasks include a synchronization requirement. One example is the task of pushing two buttons simultaneously. These are labeled `SYNC` tasks. Communication between agents is one way to accomplish `SYNC` tasks, but other mechanisms (such as clock synchronization), are also available.

### 4.1.2 The subject of action

Some tasks involve the placement or movement of an external object (e.g. the ball in soccer) while others concern the positioning of the robotic agent itself (formation). Tasks involving external objects are termed `OBJECT_BASED` tasks while those involving the agents themselves are `ROBOT_BASED` tasks.

### 4.1.3 Limited resources and competition

Many tasks, like foraging, involve limited environmental resources; as an agent performs the task it consumes the resource thus making the task more difficult. In multiagent systems, limited resources may force the agents on a team into competition between themselves or with external agencies. In addition to external environmental resources the robot may also have to monitor *internal* resources, such as fuel.

Tasks that involve limited resources are termed `RESOURCE_LIM`. Robotic foraging is typically a `RESOURCE_LIM` task because there are only a fixed number of objects available for the robots to collect. When agents on a team compete with each other for a limited resource, the task is referred to as `COMP_INT` (internally competitive). Foraging is internally competitive since one agent's success in collecting an object, reduces the opportunity for others. When the agents compete with external forces, the task is `COMP_EXT`. Soccer is a `COMP_EXT` task because the agents on one team compete with the (external) agents on the other team for the opportunity to score goals.

`RESOURCE_LIM` refers specifically to resources external to the agent (e.g. attractors) versus internal resources (e.g. fuel). Tasks that call for fuel and/or energy conservation are referred to as `ENERGY_MIN` tasks. "Dock the boat using minimum fuel" is a `TIME_UNLIM` `ENERGY_MIN` task.

### 4.1.4 Movement

The coordinated movement of several robots is important in search, surveillance, grazing and cleaning tasks. Ali has developed a taxonomy of robotic movement tasks in his investigation of human control techniques for multi-robot teams [Ali97]. His terms, `COVERAGE`, `CONVERGENCE`, `MOVEMENT_TO` and `MOVEMENT_WHILE` are adopted here for the description of multiagent movement tasks.

Multi-robot movement tasks are broken into two primary classifications: `COVERAGE` and `CONVERGENCE`. In coverage tasks the robots spread out as much as possible. Examples include search, grazing and cleaning tasks. In convergence tasks robots gather together from dispersed locations. The description of movement tasks is further refined by whether the a robot should move *to* a position or move *while* maintaining a position. In `MOVEMENT_TO` tasks the robots move from their starting locations to a particular configuration, while in `MOVEMENT_WHILE` tasks the robots are to maintain a configuration while moving.

### 4.1.5 Platform dependencies

The capabilities and limitations of the robotic platform can significantly impact the space of potential solutions. A large multi-armed tank robot for example, may be able to individually accomplish many tasks that would otherwise require dozens of smaller robots. Other issues of robot platform, like sensor ranges, can affect the space of possible solutions as well.

Consider the task of transporting a cookie to a homebase. For humans, this is a single-agent problem: pick up the cookie, carry it home. But for ants the task assumes epic multiagent proportions. Tasks in which an individual agent is able to generate positive performance are referred to as `SINGLE_AGENT`. If it is unlikely that a single agent can perform suitably – that several agents are required – the task is `MULTI_AGENT`.

Tasks that require agents in several widely separated places at once are `DISPERSED`. Dispersion might be required because of size, reach, actuator or sensing limitations of individual robots on the team. An example of this type of task is one in which two buttons on opposite sides of a building must be pushed simultaneously (this is also a `SYNC` task). A task requiring continuous surveillance over areas larger than an individual robot's sensor range require dispersion as well. Soccer is a `DISPERSED` task since the ball can move rapidly from one part of the field to another; players must be dispersed to ensure effective play. Many `COVERAGE` tasks, like coordinated search, are `DISPERSED` because the agents must spread apart to ensure more complete sensor coverage. Formation tasks are not `DISPERSED` because, in formation, agents must be relatively close together with respect to the environment.

In some tasks for robots the agents are provided perfect, or nearly perfect sensing. In other words the agent knows, with complete accuracy, every aspect of the environment's state germane to the task. Many tasks examined in the reinforcement learning literature share this attribute (e.g. the acrobot [Boo97]). These tasks are referred to as `SENSOR_COMPLETE`. It is often impossible however for an individual agent to sense all aspects of the environment relevant to the task. Even if the robot has a comprehensive sensor suite, the sensors may be degraded in actual use by noise and occlusion. Such tasks are referred to as sensor limited or `SENSOR_LIM` tasks.

The decision problem for robots in a `SENSOR_LIM` task is *partially observable* [LCK95]. This means that many distinct situations may be perceived as equivalent by the robot. For instance, as a foraging agent delivers an attractor, it might perceive itself as being in the same state whether it is 5 meters or 4 meters from the delivery area.

Finally, tasks that cannot be accomplished without communication are referred to as COMM tasks. The requirement for communication may be a consequence of sensor limitations. It can be argued for example, that with complete sensing, explicit communication is never required. SENSOR_LIM DISPERSED SYNC tasks however are likely to require communication as a means of achieving synchronization.

### 4.1.6 Classification of example tasks

The classification system outlined above is used in later chapters to classify tasks in the experimental investigation. The classification of tasks and the evaluation of diversity and performance of robot systems executing them will help identify which kinds of task are best served by diverse teams. Consider the following two tasks as examples of how classification works.

A foraging task might be expressed as "maximize the number of attractors delivered to homebase in 10 minutes." Or more explicitly as

$$\mathrm{delivered}(A_i, t) = \begin{cases} 1 & \text{if object } A_i \text{ is in the delivery zone at time } t \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

$$P = \sum_{i=1}^{N} \mathrm{delivered}(A_i, t_0 + 10\mathrm{min}) \tag{4.2}$$

where delivered($A_i, t$) indicates whether object $A_i$ is in the delivery area at time $t$, $N$ is the number of objects available for collection. $P$ is performance, the number of objects in the delivery area 10 minutes after the start of the experiment ($t_0$). The performance metric includes constraints on time (a 10 minute limit) and on the spatial arrangement of the collected objects (they must be in the delivery area). For this task there are no constraints on the location or number of robots. This foraging task is

- TIME_LIM because performance is measured over a fixed period,
- OBJECT_BASED since performance is based on the location of objects, not agents.,
- RESOURCE_LIM because as agents collect objects, the the availability of attractors is reduced.,

- `COMP_INT` because robots on a team compete for access to attractors among themselves,
- `SINGLE_AGENT` since an individual agent can perform positively, even though multiple agents may provide improved performance.

Now consider a formation maintenance task: "minimize total position error for four robots in a diamond formation," or mathematically,

$$\text{error}(R_i, t) = |\ \text{position}_i(t) - \text{desired\_position}_i(t)\ | \tag{4.3}$$

$$P = -\sum_{\tau=t_0}^{t_0+10\text{min}} \{\sum_{i=1}^{4} \text{error}(R_i, \tau)\} \tag{4.4}$$

where $\text{error}(R_i, t)$ is the formation position error for robot $i$ at time $t$ and $P$ is the sum of errors of the four robots over 10 minutes. This task is distinguished from the foraging task above in that the positions of the agents themselves are important rather than the positions of objects to be manipulated. This task is

- `TIME_LIM` because the task is carried out over 10 minutes,
- `ROBOT_BASED` because performance depends on the location of agents, not objects,
- `CONVERGENCE` since the agents should maintain specific positions close to one another,
- `MOVEMENT_WHILE` because convergence should be maintained while the robots move,
- `MULTI_AGENT` since it implicitly requires four agents.

## 4.2 A taxonomy of reinforcement functions

The reinforcement function is usually closely coupled to the performance metric for a task. In fact many reinforcement learning investigations consider performance, task and reward as one and the same. Since learning agents strive to maximize the reward signal provided them, performance is maximized when their reward closely parallels performance. It is sometimes the case however, that agents cannot or should not be rewarded strictly according to overall system performance. Some examples include:

- The robot's sensors do not provide enough information for an accurate computation of performance.
- The delay in receiving a reward is too great; learning a sequential task is too difficult and/or takes too long.

- Performance depends on the actions of other agents over which the agent has limited knowledge and/or control.

As a result, the performance metric (task) and reward function are often quite different and must be treated separately. The following sections enumerate the various kinds of reward structure that have been developed (by the author and others) in response to the issues listed above. The taxonomy is summarized in Table 4.2. Each type has advantages and disadvantages; the challenge for the designer is to select a reinforcement function that most effectively balances these trade-offs.

**Table 4.2**: Summary the Multi-robot reward taxonomy.

| Descriptor | Meaning |
|---|---|
| **Source of reward** | |
| INTERNAL_SOURCE | reward is internal based on sensor values |
| EXTERNAL_SOURCE | reward is generated by external agent |
| COMB_SOURCE | combined internal and external reward |
| **Relation to performance** | |
| PERFORMANCE | reward is tied directly to performance |
| HEURISTIC | reward based on intuition of state value |
| **Time** | |
| IMMEDIATE | immediate rewards are provided |
| DELAYED | reward is delayed |
| **Locality** | |
| LOCAL | individual agents receive unique rewards |
| GLOBAL | all agents receive identical reward signal |
| COMB_LOCALITY | combination of local and global |
| **Continuity** | |
| DISCRETE | several discrete reward values |
| CONTINUOUS | reward drawn from continuous interval |

### 4.2.1 Source of reward

A reinforcement function is classified as INTERNAL_SOURCE or EXTERNAL_SOURCE depending on whether the learner computes a reward internally based on its own sensors, or the reward is computed externally by another agent. As an example of an externally generated reward, consider a dog being trained by his master to fetch a ball. Reinforcement in this case is provided through verbal cajoling, e.g. “no!” “good dog,” etc. Notice that the pet’s reward is based on its sensing of the trainer’s mood and *not* the location of the object to be fetched. Similarly we might train a robot by equipping it with “reward” and “punish” buttons to be activated by a trainer according to his evaluation of the desirability of the robot’s behavior. This is the approach Yanco utilized in training a team of communicating robots [YS93].

External rewards might be useful in commercial intelligent systems applications where the manufacturer cannot anticipate how the consumer will use the product. It would be unreasonable for example, to assume that the purchaser of an intelligent water heater would be inclined (or able) to program a reward function for it. A more realistic interface is the provision of a “punish” button to be pressed when the heater fails to turn on and the user receives a a cold shower.

A potential problem with external reward systems however, is the possibility of a mis-match between the agent’s sensors and those of the trainer. Suppose the water heater’s owner takes showers later in the day on weekends than on weekdays. Unless the heater is equipped with a time of day *and* a day of week clock it will be impossible for it to differentiate between weekends and weekdays. The result being that the heater must remain on for the entire morning to avoid negative reinforcement.

In addition to the potential for sensor mis-match, external rewards complicate the learning problem for planning systems. If the reward is not provided to the planner as a function of sensor state and robot action, the robot must additionally learn a model of the external rewarding agent. To avoid this problem, all of the experiments in this research are conducted using internal reward functions.

There are some situations where it might be appropriate for an agent to take part of its reward from an external source and derive part of it internally. As an example, a dog may need to satisfy its hunger by eating (internal), but also wish to please its owner (external). A third descriptor `COMBINED_SOURCE` is used for reward structures that combine internal and external rewards.

### 4.2.2 Relation to performance

A reward function is defined as `PERFORMANCE` if and only if maximum reward implies optimal performance. As an example of a `PERFORMANCE`-based reinforcement function consider the task of docking a boat in minimum time. Performance for this task is $P = -\text{elapsed\_time}$. The function $R(t) = -1$ is an appropriate `PERFORMANCE`-based reward for the task because reward is maximized if and only if performance is maximized. If $T$ is the number of steps required to complete the task, $\sum_{t=1}^{T} -1$ is maximized (least negative) when the task is completed quickly.

In many tasks a `PERFORMANCE`-based reward is *delayed*: it may take many steps for an agent to reach a non-negative reward. In the forage task for instance, measured performance is negative or zero until an attractor is delivered. This delay makes it more difficult for an agent to assign credit or blame to actions taken in the past than if immediate rewards are provided. Heuristic rewards can address this problem by providing more immediate feedback. `HEURISTIC` rewards are based on the programmer's intuition of the value of an agent's actions in particular states.

As an example of a heuristic reward consider reinforcement in a robot soccer task. A `PERFORMANCE`-based function rewards or punishes the agents only when a goal is scored. An alternative heuristic approach might reinforce the agents with a reward from -1 to 1 depending on whether the ball is close to their own team's goal (-1), the center of the field (0), or the opponent's goal (1).

While heuristic reward functions usually provide for quicker learning, problems include the possibility of local maxima and/or global maxima that do not correspond with optimal performance. The learning agents may maximize their reward without necessarily optimizing performance.

### 4.2.3 Time and continuity

As mentioned above, rewards are often `DELAYED`. This means it may require many (perhaps hundreds) of movement steps before an agent receives a positive reward. Reward functions that provide immediate feedback as to the utility of an action in a particular state are called `IMMEDIATE`. `HEURISTIC` rewards often seek to improve performance by by providing more immediate feedback than their `PERFORMANCE`-based counterparts.

There are some tasks that provide `IMMEDIATE PERFORMANCE`-based rewards as well. As an example, consider the task of maintaining a specific water level by opening and closing a valve. The difference between the current water level and the desired level (the error) can be utilized as an `IMMEDIATE PERFORMANCE` reward.

The error signal in water level is a CONTINUOUS reward signal because it is drawn from a continuous interval. Other reward functions provide DISCRETE values. As an example of DISCRETE reinforcement consider the reward for a foraging robot. A discrete function might provide a reward of -1 at every step except +1 when the robot delivers an attractor.

### 4.2.4 Locality

An important consideration for learning multi-robot teams is whether each agent should be rewarded individually or all agents should receive the same reinforcement. GLOBAL reinforcement refers to the case where a single reinforcement signal is simultaneously delivered to all agents, while LOCAL reinforcement rewards each agent individually. In the case of soccer, a global system would reward all team members when any one of them scores a goal. With local reinforcement only the agent that scored the goal would be rewarded. Global rewards correspond more closely with overall system performance, but they may not be appropriate for all tasks.

In a SENSOR_LIM task, it may not be feasible for each agent to monitor activities of all the others in order to compute the global reward. In this case, implementation of global reinforcement may require expensive communication hardware. Also, there may be a weak correlation between an individual agent's actions and the value of the global signal; an agent may receive a positive reward while executing the "wrong" action. This could occur in a globally-rewarded foraging task if two agents simultaneously deposit attractors – one in the correct spot but the other in the wrong spot. Since all agents are simultaneously rewarded for the first agent's successful delivery, the second agent may incorrectly infer it behaved correctly.

Local reinforcement has the advantage of a stronger correlation between reward and the individual's actions, potentially providing for faster learning. But agents using local rewards for training may not be able to optimize performance of the overall system. Some investigators try to balance these goals by combining or blending local and global rewards [Mat94]. These reward systems are referred to as COMBINED_LOCALITY.

The impact of local and global reinforcement on robot team diversity is explored in depth in the foraging, soccer and formation experiments described later in this dissertation (Chapters 6 7 and 8).

## 4.3 Discussion and summary

To facilitate the investigation of how task and reinforcement impact multi-robot system diversity and performance, new classifications of task and reinforcement are introduced.

Task classification is based on features of the performance metric, environment and robotic platform that differentiate one task from another. The characterization focuses on robot and object movement tasks, and on features of these tasks that may correlate with a need for cooperation or diversity. This classification system helps us identify why multi-robot performance and diversity might change for different tasks, even when the same type of reinforcement is utilized to train the agents.

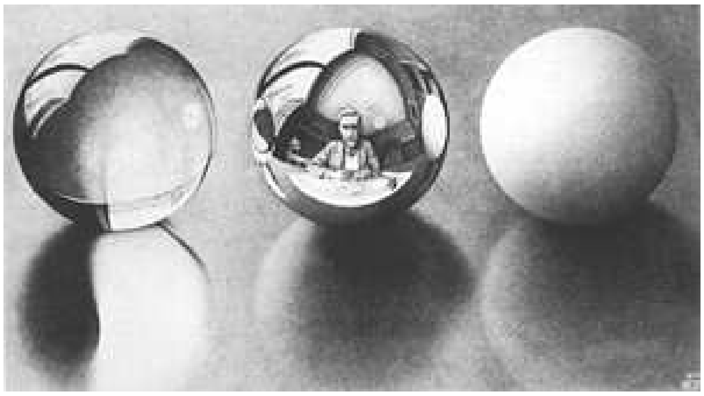
The chapter also presents a new taxonomy of reinforcement function. Ideally, learning robots are trained using a reward signal that parallels the performance metric. It is not always feasible however to utilize performance-based rewards in a multi-robot system. Research presented later in the dissertation shows that the choice of reinforcement can impact learning rate, performance and diversity in different ways. A multi-agent reward structure is classified as

- `INTERNAL` or `EXTERNAL` depending on whether the robot generates its own reward signal or it is provided by an external agent,
- `PERFORMANCE` or `HEURISTIC` depending on whether the reward is closely tied to system performance or based on the programmers intuition,
- `IMMEDIATE` or `DELAYED` depending on whether the rewards are provided at each step, or only at the end of the task,
- `DISCRETE` or `CONTINUOUS` depending on whether the reward is drawn from a continuous or discrete distribution, and
- `LOCAL` or `GLOBAL` according to whether each agent is rewarded individually or all agents are trained with identical rewards.

This classification helps define the experimental space of the research. It will also benefit the field by helping designers more easily identify the tradeoffs they make in the design of reward functions used in their systems.

# Chapter 5

# Behavioral Difference and Diversity



As research expands in multiagent intelligent systems, investigators need new tools for evaluating the artificial societies they study. It is impossible for example, to correlate heterogeneity with performance in multiagent robotics without a quantitative metric of diversity. Currently diversity is evaluated on a bipolar scale with systems classified as either *heterogeneous* or *homogeneous*, depending on whether agents differ [FM97, GM97, Par94]. Unfortunately, this labeling doesn't tell us much about the *extent* of diversity in heterogeneous teams. How can it be determined if one system is more or less diverse than another? Heterogeneity is better viewed on a sliding scale providing for quantitative comparisons. Such a metric would enable the investigation of issues like the impact of diversity on performance, and conversely, the impact of other task factors on diversity.

Diversity may not always be desirable. In fact, experimental results presented later in this dissertation show that for some multi-robot tasks homogeneous robot teams perform better than diverse teams. The aim of this work is to discover when diversity is important and which conditions give rise to it in learning teams. An objective quantitative metric is required for a principled investigation of these issues.

We focus specifically on diversity in teams of mechanically similar agents that use reinforcement learning to develop behavioral policies. Evaluation of diversity in teams of mechanically similar robots is challenging because when agents differ, they differ only in their behavior. Behavior is an especially interesting dimension of diversity in learning systems since as they learn, agents effectively choose between a hetero- or homogeneous society. The metrics introduced here will help researchers investigate the origin and benefits of diversity in these learning systems.

Measurement of diversity is really a three-pronged problem: measurement of individual agent *difference*, agent *classification* (clustering/clumping) based on inter-agent differences, and overall societal *diversity* based on the classification. Each of these topics is covered in a separate section below.

*Social entropy*, inspired by Shannon's information entropy [Sha49], is presented as an appropriate measure of diversity in robot teams. It captures important components of the meaning of diversity, including the number and size of groups in a society. In order to evaluate the diversity of a team, however, a way to categorize or differentiate the behavior of individuals is also required. To address this, a measure of *behavioral difference* that provides for agent categorization is presented in later sections. Difference refers to disparity between two specific agents, while diversity is a measure of the entire society. Finally, the utility of the metrics is demonstrated in several example applications, including a detailed evaluation of a foraging robot team.

## 5.1 Diversity

We begin with an examination of the meaning of diversity and the challenges it presents to measurement.

What does *diverse* mean? Webster [MW89] provides the following definition:

**di.verse** *adj* **1:** differing from one another: unlike **2:** composed of distinct or unlike elements or qualities

Clearly, difference plays a key role in the meaning diversity. In fact, an important challenge in evaluating robot societal diversity is determining whether agents are alike or unlike. Assume for now that any two agents are either alike (in the same behavioral group) or not. It may be true that the degree of difference *is* important but that issue is addressed later.

Consider what *diverse* means for societies composed of distinct groups. To make the discussion more concrete, suppose the "society" under examination is a collection of toy blocks of four different shapes: circles, squares, triangles and stars. Figures 5.1 and 5.2 illustrate several sets of blocks as examples of the different ways groupings can differ. The goal is to develop a quantitative metric that captures the meaning of diversity illustrated in these examples.

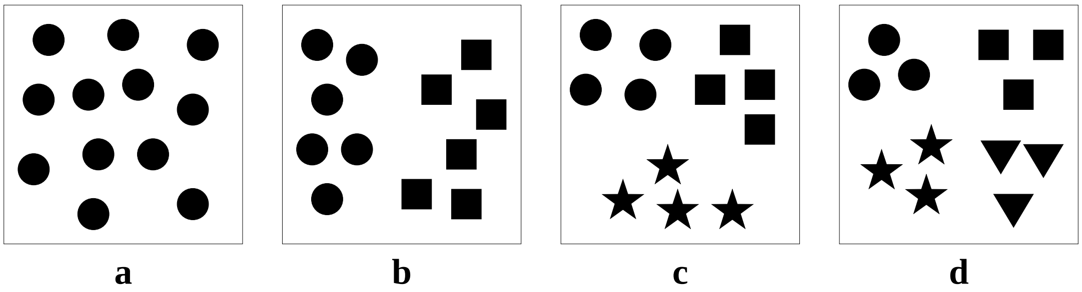



**Figure 5.1**: Four collections of toy blocks. The number of subsets in each group grows from one in **a** to four in **d**. Should measured diversity depend on the number of subsets?

First, how should the number of distinct groups in a society impact measured diversity? In Figure 5.1 we have four sets of 12 blocks. Each set has a different number of subsets of homogeneous shapes; from one subset in Figure 5.1a (all circles) to four subsets in Figure 5.1d. In each case the subsets are uniform: there are the same number of blocks in each subset. This example suggests that the number of subsets in a society is an important component of measured diversity.

Now consider Figure 5.2. Which group of blocks is more diverse? In both cases there are exactly 12 blocks and exactly two different types of block. In Figure 5.2a however, there is a much higher proportion of circles than in 5.2b where there is an equal number of circles and squares. This example suggests that the relative proportion of elements in each subset is important component of diversity.

The examples above highlight the fact that the *distribution* of agents into subsets of homogeneous elements is at the core of the meaning of diversity for multiagent societies. We make the following commitment: **the measured diversity of a**

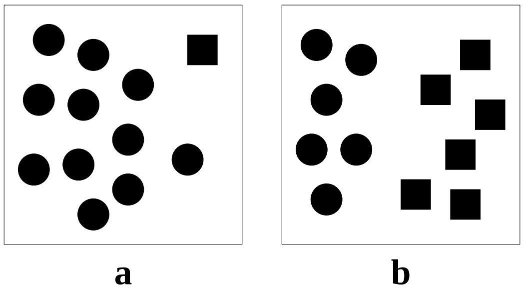


**Figure 5.2**: In both of these groups there are the same number of blocks and the same number of subsets, but the proportion of elements in each subset is different.

**multiagent society depends on the number of subsets it contains and the proportion of agents in each subset.**

### 5.1.1 Introducing social entropy

How should diversity be quantified? Shannon faced a similar problem when he sought to quantify the uncertainty, or randomness, of an information source [Sha49]. The uncertainty of an information source has important implications for communications systems, particularly with regard to the minimum bandwidth required to transmit error-free messages. Interestingly, the properties Shannon sought in a measure of information uncertainty are also important for a metric of societal diversity.

Shannon's solution, *information entropy*, is easily adapted to suit our needs in a diversity metric. The remainder of this section presents the mathematical basis for *social entropy* and explains why it is an appropriate measure of multiagent system diversity.

It is useful at this point to introduce the following notation:

- $\mathcal{R}$ is a society of $N$ agents with $\mathcal{R} = \{r_1, r_2, r_3 ... r_N\}$
- $D(r_i, r_j)$ is the difference between agents $r_i$ and $r_j$. Difference is used to classify agents into homogeneous subsets.
- $\mathcal{C}$ is a grouping of $\mathcal{R}$ into $M$ possibly overlapping subsets.
- $c_i$ is an individual subset of $\mathcal{C}$ with $\mathcal{C} = \{c_1, c_2, c_3 ... c_M\}$
- $p_i = \frac{|c_i|}{|\mathcal{C}|}$ is the proportion of agents in the $i$th subset; $\sum p_i = 1$.

Measured diversity is therefore a function of $M$ and the $p_i$s. Assume that a diversity function exists and call it $H$. The diversity of a society partitioned into $M$ subsets is written $H(p_1, p_2, p_3, ..., p_M)$. So, for instance, the diversity of the group of blocks depicted in Figure 5.2a is $H(\frac{1}{12},\frac{11}{12})$, while the diversity for the group of blocks in Figure 5.2b is $H(\frac{1}{2},\frac{1}{2})$. The diversity of a particular robot society $\mathcal{R}_a$ can also be expressed $H(\mathcal{R}_a)$.

Shannon prescribed three properties for a measure of information uncertainty. With slight changes in notation, they are equally appropriate for a measure of societal diversity:

**Property 1** continuous: $H$ should be continuous in the $p_i$.

**Property 2** monotonic: If all the $p_i$ are equal, $p_i = \frac{1}{M}$, then $H$ should be a monotonic increasing function of $M$. In other words, if there are an equal number of agents in each group, more groups implies greater diversity.

**Property 3** recursive: If a multiagent society is defined as the combination of several disjoint sub-societies, $H$ for the new society should be the weighted sum of the individual values of $H$ for the sub-groups. This property is important for the analysis of recursively composed societies (e.g. [MAC97]).

The meaning of the requirement that $H$ be recursive is illustrated in Figure 5.3. The two groups on the left are combined into a new society on the right. The groups on the left have diversities $H(\frac{5}{6},\frac{1}{6})$ (top) and $H(\frac{1}{2},\frac{1}{2})$ (bottom). The diversity of the new 18 element society is $H(\frac{5}{18},\frac{1}{18},\frac{6}{18},\frac{6}{18})$. Because the sub-groups contribute $\frac{1}{3}$ and $\frac{2}{3}$ of the agents to new society, the recursive criteria requires:

$$H(\frac{5}{18},\frac{1}{18},\frac{6}{18},\frac{6}{18}) \quad = \quad H(\frac{1}{3},\frac{2}{3}) + \frac{1}{3}H(\frac{5}{6},\frac{1}{6}) + \frac{2}{3}H(\frac{1}{2},\frac{1}{2})$$

In general, for a society $\mathcal{R}_c$ composed of two societies, $\mathcal{R}_a$ and $\mathcal{R}_b$, the recursive criteria ensures that:

$$H(\mathcal{R}_c) \quad = \quad H(\alpha,\beta)) + \alpha H(\mathcal{R}_a) + \beta H(\mathcal{R}_b)$$

where $\alpha$ is the proportion of agents in $\mathcal{R}_a$, $\beta$ is the proportion of agents in $\mathcal{R}_b$ and $\alpha + \beta = 1$.

Shannon's *information entropy* meets all these criteria. The information entropy, $H(X)$ of a symbol system $X$ is used in coding theory as a lower-bound on the average number of bits required per symbol to send multi-symbol messages. The random variable $X$ assumes discrete values in the set $\{x_1, x_2, x_3 ... x_M\}$ (the alphabet to be encoded). The information entropy of $X$ is given in bits as:

$$H(X) \quad = \quad -\sum_{i=1}^{M} p_i \, \log_2(p_i) \tag{5.1}$$

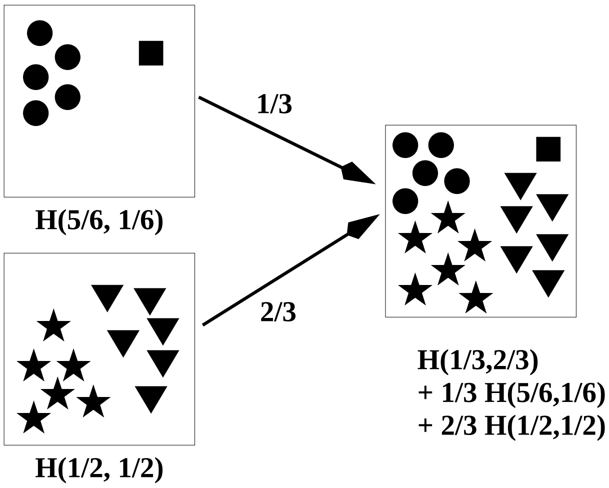



**Figure 5.3**: A new society (right) is generated by combining two others (left). The diversity of the new society is a weighted sum of the individual values of $H$ for the subgroups.

where $p_i$ represents the probability that $\{X = x_i\}$.

Equation 5.1 is adopted for the measurement of multiagent societal diversity. $H(\mathcal{R}_a)$ is the *simple social entropy* of agent society $\mathcal{R}_a$. Several example applications of this metric are given below.

### Example evaluations

Consider the social entropy of a heterogeneous group composed of one square and three star shaped blocks. The society consists of four elements, $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$. One element, $r_4$ (the square) is not equivalent to the others so there are two subsets, $C = \{c_1, c_2\}$, with $c_1 = \{r_1, r_2, r_3\}$ (the star class) and $c_2 = \{r_4\}$ (the square class). Then,

$$
\begin{aligned}
p_1 &= .75 \\
p_2 &= .25 \\
H(\mathcal{R}) &= -\sum_{i=1}^{2} p_i \log_2(p_i) \\
&= -((p_1 \log_2(p_1)) + (p_2 \log_2(p_2))) \\
&= -((.75 \log_2(.75)) + (.25 \log_2(.25))) \\
&= .811
\end{aligned}
$$

The social entropy of this system is .811.

Next the social entropy of a homogeneous group is evaluated. The group consists of elements $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$. Homogeneity implies there is only one class, so $\mathcal{C} = \{c_1\}$, and $c_1 = \{r_1, r_2, r_3, r_4\}$. Then:

$$\begin{aligned} p_1 &= 1 \\ H(\mathcal{R}) &= -\sum_{i=1}^{1} p_i \log_2(p_i) \\ &= -(p_1 \log_2(p_1)) \\ &= -(1 \log_2(1)) \\ &= 0 \end{aligned}$$

The entropy of a number of other example systems is given in Figure 5.4.

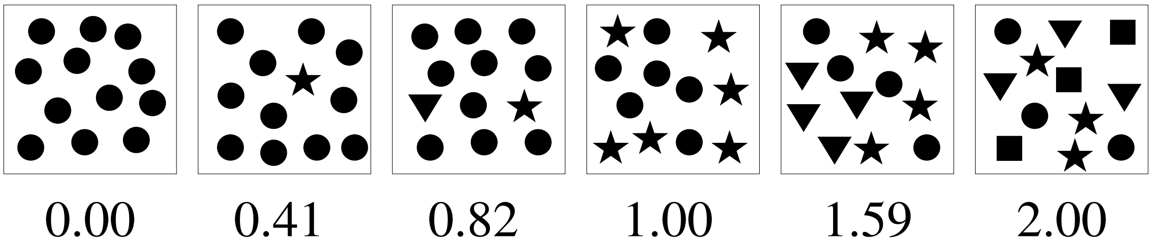



**Figure 5.4**: A spectrum of diversity. In the diagram above, each of the six squares encloses a multiagent system, from least diverse (homogeneous) on the left, to most diverse (maximally heterogeneous) on the right. The *social entropy*, a qualitative measure of diversity, is listed underneath each system.

### 5.1.2 Why entropy is a useful measure of diversity

In addition to Properties 1, 2 and 3, $H$ has a number of additional properties that further substantiate it as an appropriate measure of diversity. First, as we would expect, $H$ is minimized for homogeneous societies; these groups are the least diverse. Also, for heterogeneous groups $H$ is maximized when there are an equal number of agents in each subset. More precisely:

**Property 4:** $H = 0$ if and only if all the $p_i$ but one are zero. In other words $H$ is minimized when the system is homogeneous. Otherwise $H$ is positive.

**Property 5:** For a given $M$ (number of subsets), $H$ is maximized when all the $p_i$ are equal, i.e. $p_i = \frac{1}{M}$. This is the case when there are an equal number of agents in each group.

**Property 6:** Any change toward equalization of the proportions $p_1, p_2, \ldots, p_M$ increases $H$. Thus if $p_1 < p_2$ and we increase $p_1$, decreasing $p_2$ an equal amount so that they are more nearly equal, $H$ increases. An important implication is that there are no locally isolated maxima.

Even if these properties are desirable in a diversity metric, why choose information entropy over any other function possessing the same properties? Because, as it turns out, information entropy (Equation 5.1) is the *only* function satisfying Properties 1, 2 and 3. Shannon proved this result using the mathematically equivalent properties he required of an information uncertainty metric [Sha49].

The utility of these properties for the measurement of diversity has drawn researchers in many other disciplines to adopt similar concepts of diversity. Several examples from the literature are included below.

**Sociobiology:** Wilson, the entomologist and creator of the field of sociobiology, includes this discussion in his book *The Diversity of Life* [Wil92]:

> Suppose that we encounter a fauna of butterflies consisting of 1 million individuals divided into 100 species. Say one of the species is extremely abundant, represented by 990,000 individuals, and each of the other species therefore comprises an average of about 100 individuals. One hundred species are present but, as we walk along the forest paths and across the fields, we encounter the abundant butterfly most of the time and each of the other species only rarely ... In a nearby locality we encounter a second butterfly fauna, comprising the same 100 species, but this time all are equally abundant, represented by 10,000 individuals each. This is a fauna of high equitability, in fact the highest possible. Intuitively we feel that **the high-equitability fauna is the more diverse of the two, since each butterfly encountered in turn is less predictable and therefore gives us more information on average.**

This view embraces the idea that societies with members equally distributed among sub-groups are the most diverse. It also suggests that diversity and information are closely related concepts.

**Ecology:** entropy is used by ecologists as a means of evaluating species diversity and for comparing diversity in differing environments [LVW83, LW80, Mag88]. Consider this passage from Magurran's book *Ecological Diversity and Its Measurement* [Mag88]:

> **The most widely used measures of diversity are the information theory indices.** These indices are based on the rationale that the diversity, or information, in a natural system can be measured in a similar way to the information contained in a code or message. It is calculated from the equation:
>
> $$H = -\sum p_i \log(p_i)$$
>
> where the quantity $p_i$ is the proportion of individuals found in the $i$th species.

**Social Science:** information theoretic models of societal evolution are developed in [Bai90, May98] and others. Of these, Bailey's *Social Entropy Theory* [Bai90] is the most widely known. In his model, individuals are classified according to characters such as race, income and religion and the diversity of the society is calculated using information entropy.

**Biological Classification:** information entropy is an important tool for taxonomists as a mechanism for evaluating classification methodologies [SS73, JS71]. Classification trees are organized to maximize their information content.

**Mathematics and Statistics:** information entropy is suggested as a measure of diversity by a number of mathematicians and statisticians [Bev90, PPM92, MGL92].

**Genetics and Evolution:** Demetrius argues in *The thermodynamics of evolution* that the evolutionary process of genetic mutation leads to greater population diversity [Dem92]. The diversity is modeled as an increase in system entropy.

### 5.1.3 Limitations and alternative measures

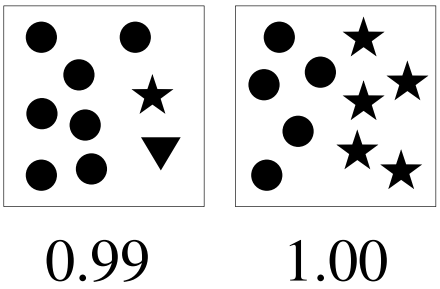
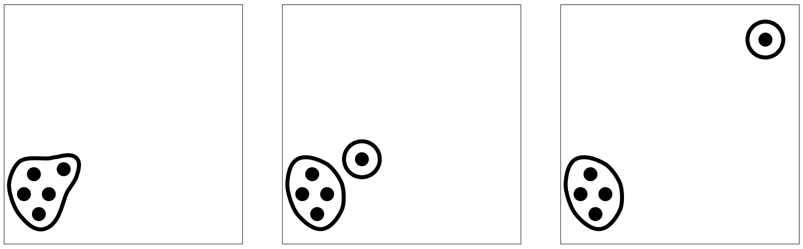



**Figure 5.5**: Two very different systems have similar entropy.

A potential limitation of social entropy as a diversity metric is the loss of information incurred when diversity is summarized in a single number. There are perhaps an infinite number of societies matching any particular value of diversity. Figure 5.5 for example, illustrates two very different societies whose entropy differs by less than 0.01 bits.

A single number does not tell us how many classes of agents there are or how many agents in each class. This loss of information occurs whenever any characteristic of a multi-dimensional system is described in a single value. However such measurements are useful because they enable generalization and comparison. A thermometer, for example, does not reveal the position and velocity of every molecule in the environment but it does enable us to select our wardrobe for the day.

Another limitation of social entropy is that it lacks sensitivity to the degree of difference between agents (later in this chapter an augmented form of social entropy is introduced to address this weakness). Suppose, for example, we are evaluating the diversity of a number of agents distributed in a two-dimensional space (the dimensions may represent aspects of behavior or perhaps morphological axes). Agents that are close to one another are grouped in the same class. Figure 5.6 illustrates.



**a** **b** **c**

**Figure 5.6**: One difficulty in the analysis of diversity. Dots representing agents are plotted in a two-dimensional space. Lines enclose agents grouped in the same class. The entropy metric cannot distinguish between the systems illustrated in **b** and **c**.

The figure shows three systems. In each system, the four elements in the lower left remain unchanged, but from 5.6a to 5.6c a fifth agent appears in several locations progressively more distant from the others. In Figure 5.6a it is just close enough to be grouped with the others, while in 5.6b it is just far enough away to be placed in a separate category. The social entropy metric cannot differentiate between the distribution of agents in 5.6b and 5.6c because there is no difference in the number and size of the subsets. Also, the entropy measure finds a greater difference between the systems in 5.6a and 5.6b than between those in 5.6b and 5.6c.

One solution is to consider the maximum difference between agents as an additional component of diversity; e.g. the distance $d$ in Figure 5.7. In the biological taxonomy literature $d$ is referred to as *maximum taxonomic distance*. Taxonomic distance is useful, but as Figure 5.7 illustrates, it cannot serve as the only measure of diversity. This example shows two societies: one society with most of the agents grouped together, but one "outlier" at distance $d$ (5.7a); and another society with two equally sized groups separated by the same distance (5.7b). Both of these systems have the same maximum difference but quite different distributions of agents into subgroups. Taxonomic difference captures the greatest difference between agents in the society but ignores the distribution of agents in the space. Again, this issue is addressed by an augmented version of social entropy presented later in the chapter.

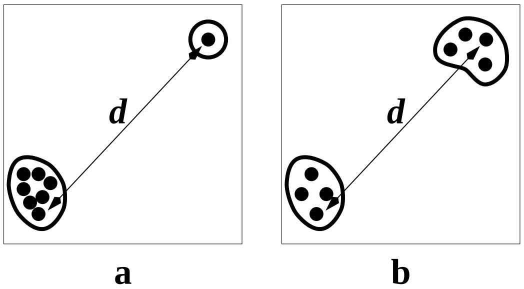



**Figure 5.7**: Maximum taxonomic distance is a useful metric, but it does not account for the distribution of elements in the space.

Meyer and McIntosh have developed an index of ethnic diversity used by *US News* and *USA Today* in stories concerning diversity issues [MM92]. Their index measures the probability that two people chosen at random (with replacement) will differ along at least one ethnic dimension. The index value ranges from 0 to 1, with greater diversity indicated by a larger value. A value of zero applies to a population in which everyone is the same. If every person is different from every other person on at least one dimension, the value is maximized. In practice the index can never reach unity because an infinite number of $p_i$ would be required. The metric has intuitive appeal, and may be of interest as a measure of multiagent social diversity. Using the notation introduced above, Meyer's metric can be written:

$$H_m(X) = 1 - \sum_{i=1}^{M} p_i^2 \tag{5.2}$$

$H_m$ shares mathematical properties 1, 2, 4, 5 and 6 with social diversity, but it does not provide for recursively defined societies (Property 3).

## 5.2 Classification and hierarchic entropy

The discussion of diversity left open the question of how agents are classified into groups. It was assumed that any two agents are either alike (in the same group) or unlike. In actuality, the robotic agents to be classified are distributed in a multi-dimensional space where the dimensions correspond to components of behavior and difference corresponds to the distance between agents in the space. Difference between agents is likely to vary continuously instead of in the binary manner assumed previously.

The limitations of social entropy discussed in Section 5.1.3 suggest that the diversity calculation would be improved if consideration were given to the spatial structure the system. Here "spatial structure" refers to the distribution of elements in the classification space. In other words, the clumpiness of the system and the distribution of the clumps in the space are important.

The challenge of finding and characterizing clumps or clusters of elements distributed in a continuous multi-dimensional space is exactly the problem faced by biologists in building and using taxonomic systems. In the case of biology the dimensions of the space represent aspects of morphology or behavior that distinguish one organism from another. In this research the dimensions are the components of agent behavior that distinguish one robot from another.

The aims of taxonomic classification are distinct from other types of classification in that one goal is to arrange the elements in a hierarchy reflecting their distribution in the classification space. Conversely, many classification tasks only require a simple partitioning of the space (e.g. categorizing e-mail into folders). Taxonomic trees are potentially more useful in the analysis of diversity than simple partitionings because they provide more information about the society's spatial structure.

Biology offers a rich literature addressing this problem. In fact, an entire field — *numerical taxonomy* — is devoted to ordering organisms hierarchically using principled numerical techniques [SS73, JS71]. Many of the approaches in numerical taxonomy are directly applicable to the problem of robot classification. They include mechanisms for building and analyzing classification structures (e.g. taxonomic trees) and for identifying organisms on the basis of these structures.

### 5.2.1 Tools from numerical taxonomy

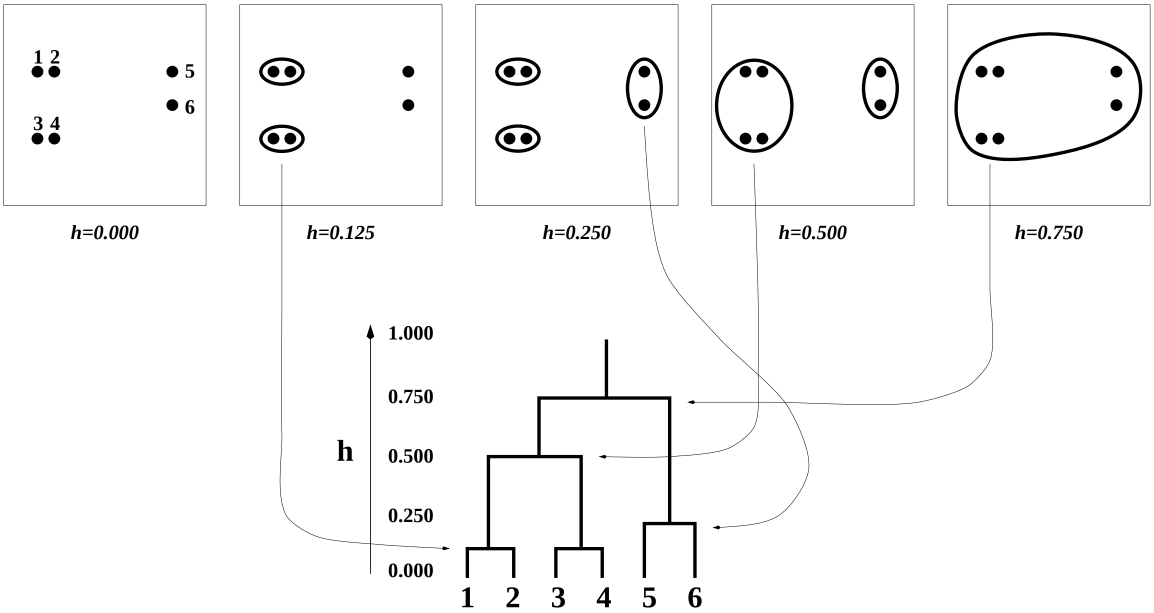



**Figure 5.8**: Example classification using numerical techniques. The top row shows how the system is clustered at several levels, parameterized by taxonomic level $h$ ($h$ is distinct from information entropy $H$). The classification is summarized in a taxonomic tree, or dentogram (bottom). Strong similarities between elements are indicated by grouping near the bottom of the dentogram; weaker similarities between groups are reflected in converging branches at higher levels.

Figure 5.8 provides an example of the numerical taxonomic approach. Six elements (they could be organisms, species or robots) are distributed about a two-dimensional space. The location of each element in the space is determined by the value of each trait (e.g. tail length, weight, etc.) used in the classification. Each trait corresponds to a dimension in the classification space. The goal is to build a taxonomic tree that reflects the spatial distribution of elements in the system: closely related elements should be grouped together at the bottom; similarities between groups are indicated as the branches converge at higher levels. These relations are expressed graphically in a *dentogram* (Figure 5.8, bottom).

Techniques from numerical taxonomy address the problem of how to group organisms, or groups of organisms, at various levels. At the lowest level in biological classification for instance, humans and gorillas are more likely to be grouped together than, say, humans and dogs. But at a higher level, primates are in fact grouped together with canines in the class *mammalia*. Dentograms provide an orderly hierarchical view of the these groupings. While dentograms *per se* are not necessary for the evaluation of diversity, they are useful visualization tools and their construction provides clues for the evaluation of overall societal diversity.

Dentograms are constructed using a clustering algorithm parameterized by $h$, the maximum difference allowed between elements in the same group (more detail on clustering algorithms is presented later). In most applications the difference metric is normalized so that taxonomic distance between any two elements varies between 0 and 1. When $h = 1$ all elements are grouped together in one cluster (see the cluster at the top right in Figure 5.8 for example). As $h$ is reduced from 1 down to 0 cluster boundaries change; the number of groups increases as they split into smaller clusters. The splits are reflected as branches in the dentogram. Finally, when $h = 0$ each element is a separate cluster; a "leaf" at the bottom of the dentogram "tree."

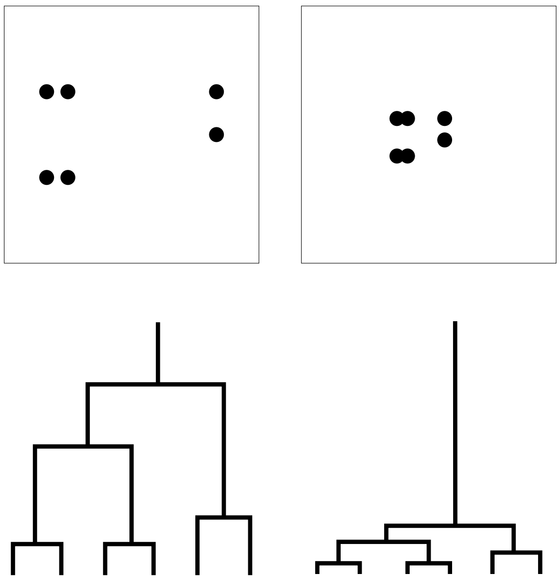


**Figure 5.9**: The branching structure of the dentograms for these two societies is the same. However, the more compact distribution of elements in the system on the upper right is reflected in the branches being compressed towards the bottom of the corresponding dentogram (lower right).

Dentograms can reveal subtle differences in societal structure. Figure 5.9 for example, shows two societies with the same relative arrangement of elements, but one grouping is compact while the other is spread out over a larger area. The difference in scale is reflected in a compressed dentogram for the spatially compact society (Figure 5.9 right). Can these differences be accounted for in the evaluation of diversity?

Before addressing this, it is necessary to examine some of the details of clustering algorithms used to build a taxonomic tree. After that, the discussion returns to how these techniques can be used in the evaluation of diversity.

### 5.2.2 Clustering algorithms

Literally hundreds of clustering algorithms have been developed by researchers in a wide range of fields (Sneath and Sokal present a comprehensive taxonomy of clustering methods in [SS73]). One reason for the proliferation of techniques is the lack of generally agreed upon optimality criteria for evaluating the various methods. Jardine, for instance, suggests information-based metrics for biological clustering applications, but this may not be appropriate in all domains [JS71]. Because we are interested in the advantages of taxonomic representations of societal structure, the field of numerical taxonomy is an appropriate source of techniques for this research.

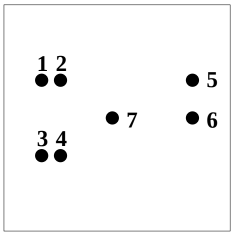
Most clustering methods used in numerical taxonomy are *hierarchic* as defined by Sneath and Sokal [SS73]:

> In *hierarchic* classifications any member of a lower ranking taxon is also a member of a higher ranked taxon, although not all its associates from the lower ranking taxon will necessarily be included in the higher ranking taxon. *Non-hierarchic* classifications do not exhibit ranks in which subsidiary taxa become members of larger more inclusive taxa. For traditional biological taxonomy, hierarchic classifications are required.

Another important distinction between clustering algorithms is whether or not overlap is allowed between clusters. In a *nonoverlapping* method, taxa at any one rank are mutually exclusive; a member of one group cannot also be a member of a second group at the same rank. Nonoverlapping classifications must sometimes arbitrarily assign elements to one or another equally distant group. By relaxing this constraint, *overlapping* methods allow membership in more than one taxa. The two approaches are examined below.

### Nonoverlapping hierarchic clustering

Most clustering algorithms begin with a two-dimensional difference matrix. Each location in the matrix records the difference between two agents in the system to be classified. For instance, cell $ij$ contains the value of $D(r_i, r_j)$, the difference between agents $r_i$ and $r_j$ (note: behavioral difference for robots is defined in the next section). Figure 5.10 illustrates an example system; the associated difference matrix is given in Table 5.2.2.




| element | x | y |
|---|---|---|
| 1 | -0.35 | 0.25 |
| 2 | -0.25 | 0.25 |
| 3 | -0.35 | -0.25 |
| 4 | -0.25 | -0.25 |
| 5 | 0.35 | 0.25 |
| 6 | 0.35 | 0.00 |
| 7 | 0.00 | 0.00 |

**Figure 5.10**: Example society of elements distributed in a two-dimensional space.

**Table 5.1**: Difference matrix for the example society. In this case, differences are calculated as Euclidean distances in the two-dimensional space.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **1** | 0.000 | - | - | - | - | - | - |
| **2** | 0.100 | 0.000 | - | - | - | - | - |
| **3** | 0.500 | 0.510 | 0.000 | - | - | - | - |
| **4** | 0.510 | 0.500 | 0.100 | 0.000 | - | - | - |
| **5** | 0.700 | 0.600 | 0.860 | 0.781 | 0.000 | - | - |
| **6** | 0.743 | 0.650 | 0.743 | 0.650 | 0.250 | 0.000 | - |
| **7** | 0.430 | 0.354 | 0.430 | 0.354 | 0.430 | 0.350 | 0.000 |

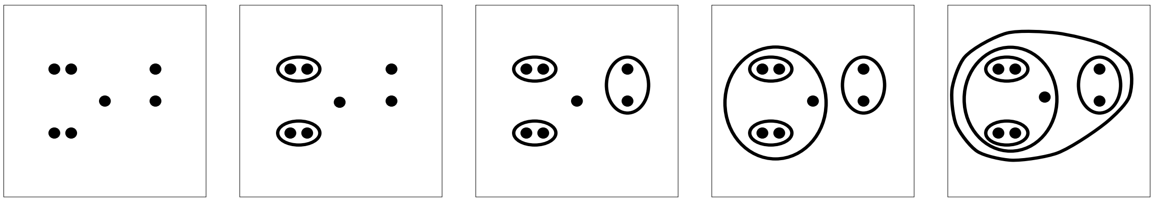
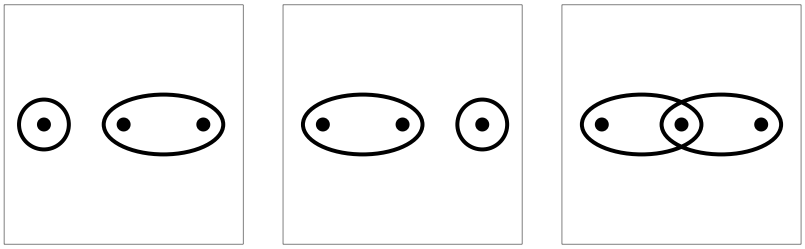


**Figure 5.11**: Example of hierarchic nonoverlapping clustering. Clusters are enclosed in black lines.

To compute the clusters at level $h$, nonoverlapping methods generally proceed in the following steps:

1. While the smallest value in the difference table, $h'$, is less than $h$:
   (a) Group elements with difference $h'$ together in a new cluster(s).
   (b) Remove elements in the new cluster(s) from the difference table.
   (c) Calculate the centroid of the new cluster(s).
   (d) Compute a new difference matrix including the new centroid(s).

The result is a hierarchy of nested clusters (Figure 5.11). Various algorithms differ in specific details, particularly in how they handle the situation where multiple pairs of elements share the same minimum difference value.



**a** **b** **c**

**Figure 5.12**: A dilemma for nonoverlapping algorithms. With which other element should the center element be grouped? Nonoverlapping algorithms must make an arbitrary choice between clustering **a** or **b**. If overlapping is allowed however, a single unambiguous solution emerges (**c**).

A key problem with nonoverlapping methods is that they must sometimes group elements arbitrarily. Figure 5.12 illustrates a case where two equally appropriate clusterings are possible (cases **a** and **b**). The nonoverlapping requirement forces an arbitrary choice of one over the other. When overlapping clusters are allowed, however, a single unambiguous solution is possible (case **c**).

### Overlapping hierarchic clustering

In contrast to the nonoverlapping methods, overlapping clustering techniques allow individual elements to join more than one cluster. Overlapping methods are typically characterized by the degree of overlap allowed. Overlap can be quantified as the diameter of overlap or as the number of elements in the overlapping region.

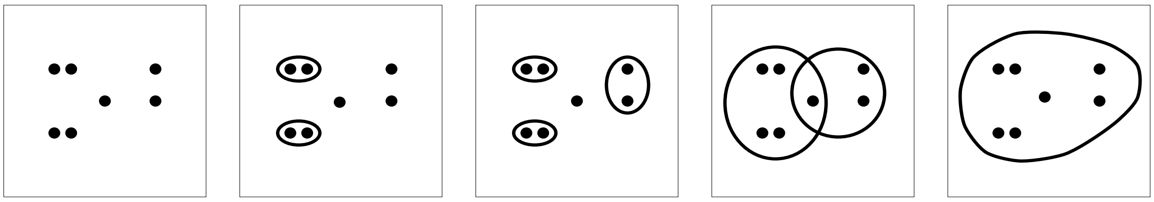


**Figure 5.13**: Example of hierarchic overlapping clustering. Clusters are enclosed in black lines.

The $C_u$ clustering method is used in this research [JS71]. $C_u$ or *u-diametric clustering* methods permit the diameter of overlap between clusters at level $h$ to be at most $uh$ (in this work $u = 1$). A cluster at level $h$ is a maximally linked set such that for all elements $r_i$ and $r_j$ in the cluster $D(r_i, r_j) \leq h$.

Reviewing the notation presented earlier, the society of $N$ elements to be clustered is $\mathcal{R} = \{r_1, r_2, r_3, \ldots r_N\}$. The society will be divided into $M$ possibly overlapping clusters $\mathcal{C} = \{c_1, c_2, c_3, \ldots c_M\}$. The $C_u$ algorithm for clustering at level $h$ proceeds in the following sequence:

1. Initialize $N$ clusters with $c_i = \{r_i\}$.
2. For each cluster $c_i$:
   (a) For each $r_j$ (except $r_i$) in $\mathcal{R}$:
      i. If $(D(r_j, r_k) \leq h)$ for every $r_k$ already in $c_i$) add element $r_j$ to cluster $c_i$.
3. Discard redundant clusters.

An example society grouped using $C_u$ clustering is presented in Figure 5.13. The clusterings for several values of $h$ are illustrated with $h$ increasing from left to right. Notice in the fourth diagram that the element in the middle of the space is claimed by two clusters (contrast this with Figure 5.11). This clustering technique is *hierarchic* because elements grouped at one level, or value of $h$, are also members of higher level taxa. In addition, the taxa (clusters) become larger and more inclusive at higher levels.

### The AutoClass clustering program

In addition to the techniques outlined above, there are a number of other clustering algorithms in current use. One of these, AutoClass, was evaluated for use in this research [CS96]. AutoClass is a Bayesian nonhierarchic nonoverlapping clustering algorithm available in the public domain. According to Cheeseman, one of AutoClass' developers, "the goal of Bayesian unsupervised classification is to find the most probable set of class descriptions given the data and prior expectations."

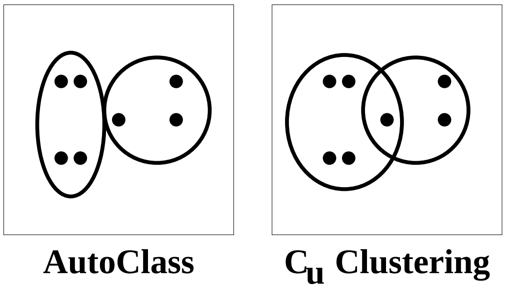



**Figure 5.14**: A comparison of clusterings generated by AutoClass (left) and $C_u$ clustering (right) for the example data. $h = 0.51$ for $C_u$ clustering.

The AutoClass system was used to classify several datasets for comparison with $C_u$ clustering. The AutoClass classification for the system given in Figure 5.10 is provided in Figure 5.14. The result is compared with a $C_u$ clustering of the same system. In these examples $h$ for $C_u$ clustering is set to 0.51; the clusterings indicate element similarities at a particular scale. Note that $h$ can be varied, however, to examine the structure of the society at any scale.

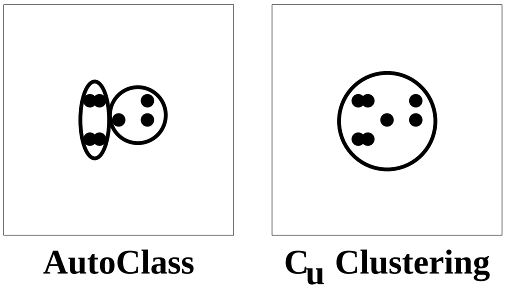



**Figure 5.15**: A set of clusterings for a more compact society. The locations of elements in this example are scaled by $\frac{1}{2}$ in comparison to the previous example. $h = 0.51$ for $C_u$ clustering.

In another evaluation, AutoClass was used to classify a more spatially compact system (for this evaluation, the previous data values are multiplied by $\frac{1}{2}$). The same axes were used in both evaluations, only the locations of the elements in the space were changed. Results of the classification are given in Figure 5.15, and for comparison the same system is classified using the $C_u$ algorithm. The groups found by AutoClass are the same as in the previous example. In contrast, the $C_u$ algorithm groups all of the elements together in one cluster.

Two additional tests were conducted to evaluate AutoClass and $C_u$ clustering at extreme ends of the spectrum. In the first test the data values used in Figure 5.14 were multiplied by 100000 and the resulting system was again classified by AutoClass and $C_u$ clustering. As before, AutoClass groups the elements in the same way. $C_u$ clustering, with $h = 0.51$ places each element in a separate group. Finally, the data values were multiplied by $\frac{1}{100000}$ and grouped by AutoClass and $C_u$ clustering. Again, AutoClass finds the same groups. In contrast $C_u$ clustering places all elements in the same group.

### Discussion on clustering algorithms

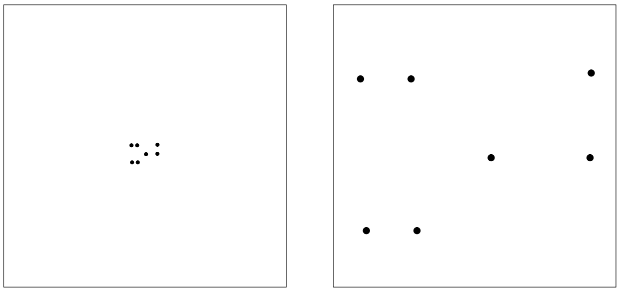


**Figure 5.16**: n example of two societies differing only in the degree of difference between elements. Axes used in both systems are the same. The relative position of each element is the same in each society, but in the system on the right they are much further apart. A useful diversity metric should distinguish between these two societies.

It is impossible to distinguish between the systems illustrated in Figure 5.16 on the basis of clusterings generated by AutoClass because AutoClass is not sensitive to the degree of difference between elements. It would cluster both systems identically. In fairness to AutoClass, it is likely that its designers sought to avoid sensitivity to the spatial scope of the data under evaluation. This feature is desirable in classification applications involving a single dataset. This work, however, is concerned with *comparing* several datasets that may vary in spatial extent (e.g. the systems in Figure 5.16).

The spatial extent of elements in a taxonomic space is a reflection of the degree of difference between agents. It has already been pointed out (in Section 5.1.3) that such differences are important in the evaluation of diversity, especially for distinguishing between societies with similar structure and numbers of elements but with differing spatial size. Overall spatial size for instance, is the only difference between the societies illustrated in Figure 5.16. Because differentiating between societies on the basis of their extent is important in the evaluation of diversity, this work utilizes $C_u$ clustering in diversity calculation.

Note that sensitivity to the degree of difference between elements in hierarchic clustering depends on $h$. Because $h$ is a *parameter* of the clustering algorithm, it can be varied to examine clusterings at any scale. Hierarchic algorithms are, in effect, variable power clustering microscopes. For values of $h$ near zero the tiniest difference between elements will cause them to be grouped separately, while the clusterings at large values of $h$ reveal societal structure at a macroscopic level. This feature is exploited in the development of a diversity measure sensitive to differences in the spatial size of societies.

### 5.2.3 Hierarchic social entropy

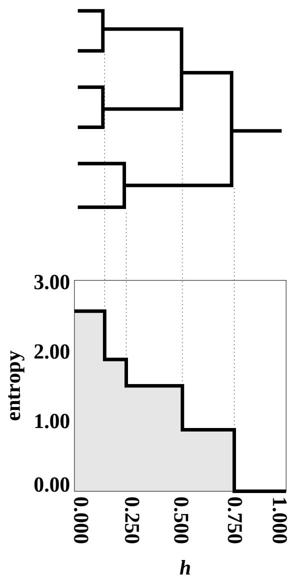



**Figure 5.17**: Entropy depends on $h$. Note that changes in entropy correspond to the branch points in the dentogram. (For easier reference, the dentogram is rotated 90 degrees.)

Now consider how tools from numerical taxonomy can be applied to the measurement of diversity. The discussion of hierarchic clustering algorithms above described how the number and size of clusters depend on $h$. But how is social entropy impacted by changes in $h$? Since the partitioning of a society is based on $h$ the entropy also depends on it. An example of the relationship is illustrated in Figure 5.17. Entropy changes in discrete steps as $h$ increases. Note that points where change occurs correspond to branch points in the dentogram.

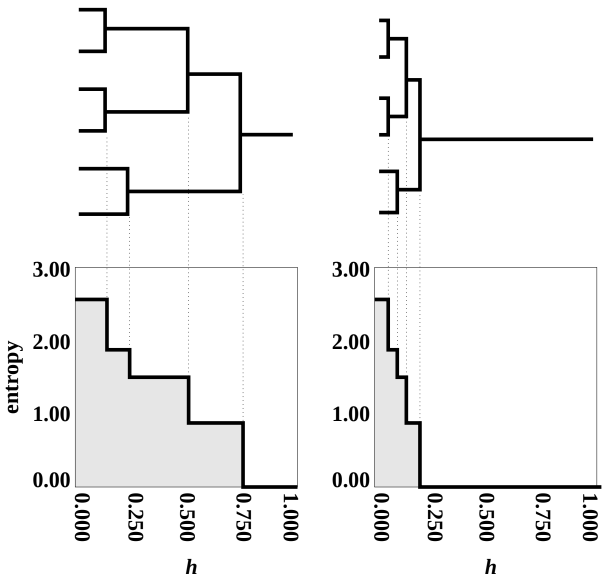



**Figure 5.18**: A comparison of entropy versus $h$ for for two societies.

Compare the dentograms and entropy plots of the two societies in Figure 5.18. As in the earlier example, the two groups have the same relative structure, but the society represented on the right is more compact, resulting in branching compressed towards the bottom of the tree. The difference in scale is also readily apparent in the plots of entropy. Entropy drops to zero much more quickly in the plot corresponding to the compact society. Because the value of simple entropy depends significantly on $h$ when hierarchic clustering is used, we augment the notation to account for this:

$$H(\mathcal{R}, h) \quad = \quad H(\mathcal{R}) \text{ for clustering at taxonomic level } h \tag{5.3}$$

$H$ is a function of $\mathcal{R}$ *and* $h$ because the classification of agents into groups, and therefore the entropy, depends on them both. This highlights the fact that **the entropy of a particular clustering is only a snapshot of the society's diversity**. A comprehensive evaluation of diversity should account for clustering at all taxonomic levels. This is easily accomplished using the area under the entropy plot as a measure of diversity. This augmented metric, called *hierarchic social entropy*, is defined as:

$$S(\mathcal{R}) \quad = \quad \int_0^\infty H(\mathcal{R}, h)dh \tag{5.4}$$

where $\mathcal{R}$ is the robot society under evaluation, $h$ is a parameter of the clustering algorithm indicating the maximum difference between any two agents in the same group and $H(\mathcal{R}, h)$ is the simple entropy of the society for the clustering at level $h$. Note that as $h \to \infty$ a point is reached where all elements are clustered in the same group (the maximum taxonomic distance). $H(\mathcal{R}, h)$ drops to 0 at this point. In the behavioral difference measure used in this work, the maximum possible difference between elements is fixed at 1.0, so the upper limit of the integration is 1 rather than $\infty$ as in the general case.

To clarify how hierarchic entropy is computed, several examples are provided in the next section.

### 5.2.4 Why hierarchic social entropy is a useful metric

Hierarchic social entropy is a continuous ratio measure; it has an absolute zero (when all elements are identical) and equal units (bits). This enables a total ordering of societies on the basis of diversity. It also provides for quantitative results of the form "$\mathcal{R}_b$ is *twice* as diverse as $\mathcal{R}_a$." This is a significant advantage over the categorization of systems as simply "homogeneous" or "heterogeneous." Several other useful properties of hierarchic entropy are examined below.

**Hierarchic social entropy can distinguish differences between societies regardless of scale.** Societies with infinitesimally small differences are compared as easily and precisely as systems spanning millions of units. This property is demonstrated with an example. Figure 5.19 illustrates two societies of three elements arranged in triangles. In both cases the two elements on the left are spaced a distance $x$ apart. A third element is placed either $2x$ or $4x$ from the other elements in societies $\mathcal{R}_{2x}$ and $\mathcal{R}_{4x}$ respectively. Because hierarchic social entropy is scale invariant, it is able to distinguish between these two systems for all values of $x$. This will be demonstrated for $x = \frac{1}{1000000}$, for $x = 1000000$ and proven for all $x$.

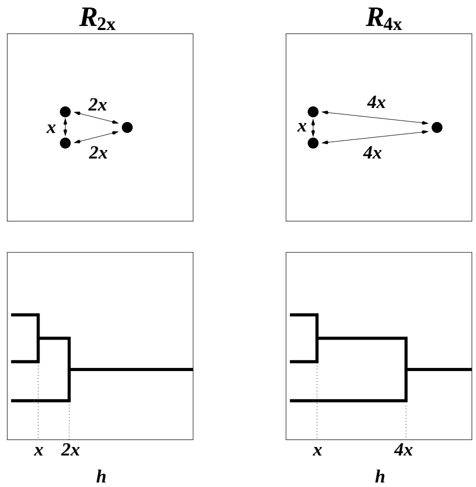



**Figure 5.19**: These two example systems are used to demonstrate how hierarchic social entropy can distinguish differences between societies regardless of scale. Spacing between the elements is parameterized by $x$ (top). Because hierarchic entropy is scale invariant it can distinguish between the two societies regardless of the value of $x$. Dentograms (bottom) illustrate the values of $h$ where clusterings change.

First, observe that due to the spacing of the elements, there are three distinct $C_u$ clusterings for each system (depending on $h$).[1] For society $\mathcal{R}_{2x}$, the three elements are placed in three separate clusters when $0 \leq h < x$. Two clusters are present when $x \leq h < 2x$. Finally, all three elements are grouped together in one cluster when $2x \leq h$. The groupings over all three ranges are illustrated in Figure 5.20 (groupings

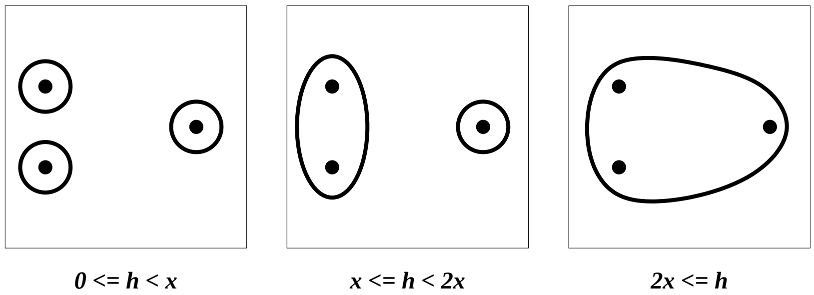



**Figure 5.20**: $C_u$ clusterings of $\mathcal{R}_{2x}$ for different values of $h$.

are similar for $\mathcal{R}_{4x}$ except the final clustering does not occur until $h \geq 4x$). The simple entropy for each clustering of $\mathcal{R}_{2x}$ is

[1]Note: the AutoClass program groups all elements together in the same cluster for both societies when $x = \frac{1}{1000000}$ and when $x = 1000000$. A diversity metric based on AutoClass clusterings would not distinguish between these two societies.

$$
\begin{aligned}
H(\frac{1}{3},\frac{1}{3},\frac{1}{3}) &= 1.585 \text{ for } 0 \leq h < x \\
H(\frac{2}{3},\frac{1}{3}) &= 0.811 \text{ for } x \leq h < 2x \\
H(\frac{3}{3}) &= 0.000 \text{ for } 2x \leq h
\end{aligned}
$$

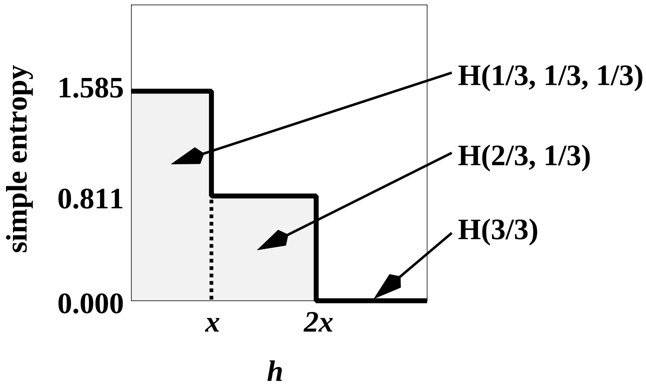



**Figure 5.21**: Simple entropy of $\mathcal{R}_{2x}$ as a function of $h$. There are three distinct regions with different values.

These values and the regions over which they apply are illustrated in Figure 5.21. Similarly, the simple entropy for each clustering of $\mathcal{R}_{4x}$ is

$$
\begin{aligned}
H(\frac{1}{3},\frac{1}{3},\frac{1}{3}) &= 1.585 \text{ for } 0 \leq h < x \\
H(\frac{2}{3},\frac{1}{3}) &= 0.811 \text{ for } x \leq h < 4x \\
H(\frac{3}{3}) &= 0.000 \text{ for } 4x \leq h
\end{aligned}
$$

Now, suppose $x = \frac{1}{1000000}$. Can hierarchic entropy distinguish between these two systems? First we calculate the hierarchic entropy of society $\mathcal{R}_{2x}$. Recall the definition of hierarchic social entropy (Equation 5.4):

$$
S(\mathcal{R}) = \int_0^\infty H(\mathcal{R}, h)dh
$$

As was pointed out above, $H(\mathcal{R}, h)$ takes on distinct values over three regions depending on $h$. Therefore, the integral can be broken into parts corresponding to these regions:

$$\int_0^\infty H(\mathcal{R}_{2x}, h)dh = \int_0^x H(\mathcal{R}_{2x}, h)dh + \int_x^{2x} H(\mathcal{R}_{2x}, h)dh + \int_{2x}^\infty H(\mathcal{R}_{2x}, h)dh$$

Substituting $\frac{1}{1000000}$ for $x$ and the simple entropy values above for $H(\mathcal{R}_{2x}, h)$, we have

$$\begin{aligned}
\int_0^\infty H(\mathcal{R}_{2x}, h)dh &= \int_0^{\frac{1}{1000000}} H(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) + \int_{\frac{1}{1000000}}^{\frac{2}{1000000}} H(\frac{2}{3}, \frac{1}{3}) + \int_{\frac{2}{1000000}}^\infty H(\frac{3}{3}) \\
&= \int_0^{\frac{1}{1000000}} 1.585 + \int_{\frac{1}{1000000}}^{\frac{2}{1000000}} 0.811 + \int_{\frac{2}{1000000}}^\infty 0.000 \\
&= 2.396 \times 10^{-6}
\end{aligned}$$

The hierarchic social entropy of $\mathcal{R}_{2x}$ is $2.396 \times 10^{-6}$. The calculation for $\mathcal{R}_{4x}$ is similar:

$$\begin{aligned}
\int_0^\infty H(\mathcal{R}_{4x}, h)dh &= \int_0^x H(\mathcal{R}_{4x}, h)dh + \int_x^{4x} H(\mathcal{R}_{4x}, h)dh + \int_{4x}^\infty H(\mathcal{R}_{4x}, h)dh \\
&= \int_0^{\frac{1}{1000000}} H(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) + \int_{\frac{1}{1000000}}^{\frac{4}{1000000}} H(\frac{2}{3}, \frac{1}{3}) + \int_{\frac{4}{1000000}}^\infty H(\frac{3}{3}) \\
&= \int_0^{\frac{1}{1000000}} 1.585 + \int_{\frac{1}{1000000}}^{\frac{4}{1000000}} 0.811 + \int_{\frac{4}{1000000}}^\infty 0.000 \\
&= 4.018 \times 10^{-6}
\end{aligned}$$

For system $\mathcal{R}_{4x}$ we have $S(\mathcal{R}_{4x}) = 4.018 \times 10^{-6}$. Therefore when $x = \frac{1}{1000000}$ $S(\mathcal{R}_{2x}) < S(\mathcal{R}_{4x})$ and $\mathcal{R}_{4x}$ is 1.68 times more diverse than $\mathcal{R}_{2x}$.

What if $x = 1000000$? For $\mathcal{R}_{2x}$ the computation proceeds as follows:

$$\begin{aligned}
\int_0^\infty H(\mathcal{R}_{2x}, h)dh &= \int_0^x H(\mathcal{R}_{2x}, h)dh + \int_x^{2x} H(\mathcal{R}_{2x}, h)dh + \int_{2x}^\infty H(\mathcal{R}_{2x}, h)dh \\
&= \int_0^{1000000} H(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) + \int_{1000000}^{2000000} H(\frac{2}{3}, \frac{1}{3}) + \int_{2000000}^\infty H(\frac{3}{3})
\end{aligned}$$

$$
\begin{aligned}
&= \int_0^{1000000} 1.585 + \int_{1000000}^{2000000} 0.811 + \int_{2000000}^{\infty} 0.000 \\
&= 2.396 \times 10^6
\end{aligned}
$$

Similarly, the hierarchic entropy for system $\mathcal{R}_{4x}$ is $4.018 \times 10^6$. So when $x = 1000000$ we again have $S(\mathcal{R}_{2x}) < S(\mathcal{R}_{4x})$; society $\mathcal{R}_{4x}$ is again 1.68 times more diverse than $\mathcal{R}_{2x}$

In fact, $S(\mathcal{R}_{2x}) < S(\mathcal{R}_{4x})$ holds for all values of $x > 0$:

$$
\begin{aligned}
\int_0^{\infty} H(\mathcal{R}_{2x}, h)dh &\stackrel{?}{<} \int_0^{\infty} H(\mathcal{R}_{4x}, h)dh \\
\int_0^{x} H(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) + \int_x^{2x} H(\frac{2}{3}, \frac{1}{3}) + \int_{2x}^{\infty} H(\frac{3}{3}) &\stackrel{?}{<} \int_0^{x} H(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) + \int_x^{4x} H(\frac{2}{3}, \frac{1}{3}) + \int_{4x}^{\infty} H(\frac{3}{3}) \\
\int_0^{x} 1.585 + \int_x^{2x} 0.811 + \int_{2x}^{\infty} 0.000 &\stackrel{?}{<} \int_0^{x} 1.585 + \int_x^{4x} 0.811 + \int_{4x}^{\infty} 0.000 \\
1.585x + 0.811x + 0.000 &\stackrel{?}{<} 1.585x + 0.811 \times 3x + 0.000 \\
2.396x &< 4.018x
\end{aligned}
$$

In addition to scale invariance, hierarchic entropy benefits from several other advantages. **Hierarchic entropy addresses a key weakness of simple social entropy by accounting for continuous differences between elements in the society.** Figure 5.22 illustrates the kind of difference in societal structure hierarchic entropy can distinguish. In an earlier example, simple social entropy could not resolve differences between these systems (Figure 5.22). However, when hierarchic social entropy is employed, the measured diversity of the three systems increases linearly as the one agent is positioned further and further away. As one would expect, the difference in diversity between systems 5.22a and 5.22b is much smaller than that between 5.22b and 5.22c. This is not necessarily the case when simple entropy is used (as the earlier example illustrates)

**Hierarchic entropy preserves the basic properties of simple social entropy.** Hierarchic entropy is a more general metric than simple entropy, subsuming the properties of $H$ at each taxonomic level $h$. In the case where difference between agents is binary, (either alike or unlike), Equation 5.4 degenerates to $H(\mathcal{R})$ (simple entropy) because the clustering does not depend on $h$. However, when continuous differences are important, hierarchic entropy can resolve structural difference in so-

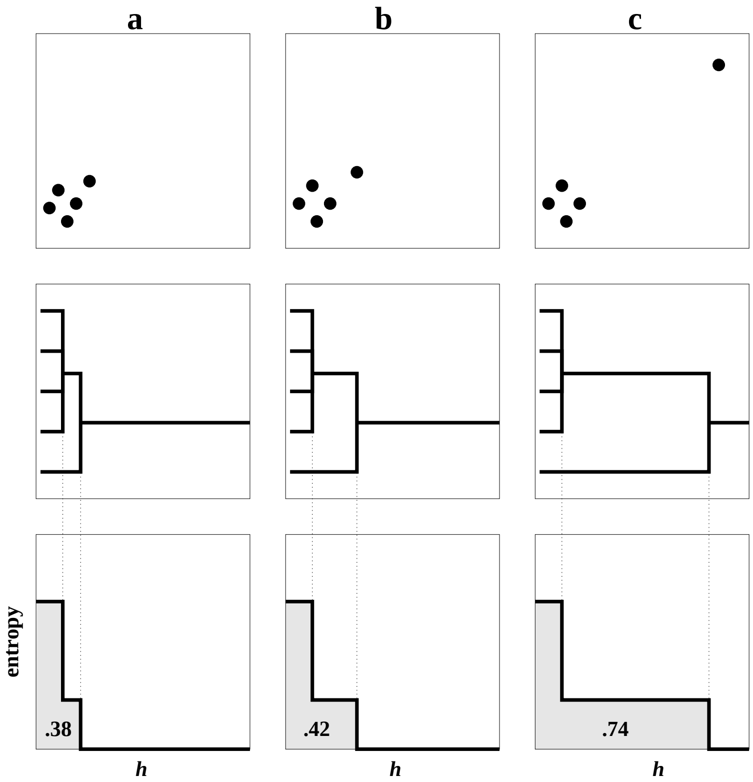



**Figure 5.22**: Hierarchic social entropy (bottom) is computed for three societies (top). The calculated value increases as the element on the upper right is positioned further away from the group. Dentograms for the groups are also displayed (middle row).

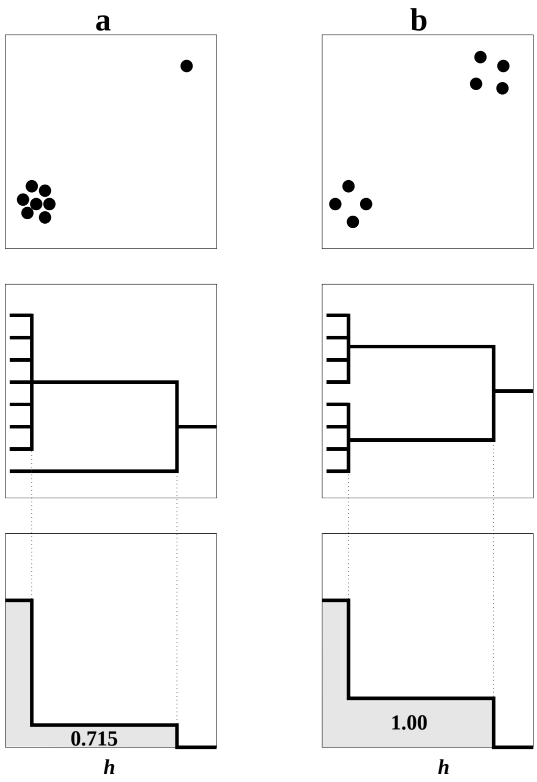



**Figure 5.23**: Hierarchic social entropy retains the basic properties of simple entropy. The computed value (bottom) depends on the distribution of elements in the groups. Dentograms for the two groups are also displayed (middle row).

cieties that simple social entropy cannot.

Figure 5.23 shows how the basic properties of social entropy are preserved with hierarchic entropy. In this example two groups are located a fixed distance apart in the classification space. The two societies pictured differ only in the distribution of elements between the groups. Hierarchic entropy properly captures the increased diversity of the system with agents distributed equally between the groups.

## 5.3 Behavioral difference

To summarize the chapter so far, hierarchic clustering is a means of dividing a society into groups of behaviorally equivalent agents at a particular taxonomic level. Diversity is evaluated at each taxonomic level based on the number of groups and the number of robots in each group at that level. Integrating the diversity across all taxonomic levels produces an overall measure of diversity for the system. Previous sections have described the overall diversity metric and algorithms for clustering the agents into groups. This section focuses on the difference metric used for clustering.

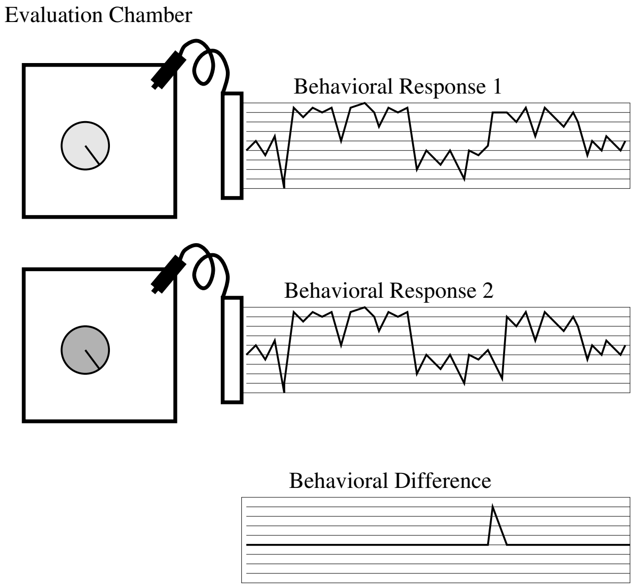



**Figure 5.24**: [Evaluating behavioral difference using an idealized "evaluation chamber." Robots are evaluated in the chamber (left), where their response to every situation is recorded as a trace (readout, right). The behavioral difference between two agents is the difference between their traces (bottom). A single quantitative value is given by integrating the difference.

How should the behavior of two agents be compared? One possibility would be to evaluate their difference in an "evaluation chamber" in which the robots are exposed to all situations and their their responses recorded. Even though it is unlikely such a chamber could be built, the analogy is useful. Figure 5.24 illustrates the procedure. As the agents are exposed to various situations, responses are recorded as a trace. After the experiment is concluded, the traces are compared to evaluate the difference between agents. In the figure, the horizontal axis of the traces represents all distinct perceptual situations a robot might experience, while the vertical component encodes the agent's response.

Since a real evaluation chamber would be practically impossible to build, an alternative method for evaluating behavioral difference is proposed. The technique advocated here is to look for differences in the agents' behavioral coding. In many cases (e.g. [BBC$^+$95, Mat92, GM97]) robot behavior is coded statically ahead of time, thus individuals may be directly compared by evaluating their behavioral configuration. Learning multi-robot systems (e.g. [Bal97c, Mat94]) pose a challenge because their behavior evolves over time. To avoid that problem in this research, the policies of learning agents are evaluated after agents converge to stable behavior.

This approach depends on three key assumptions:

**Assumption 1:** At the time of comparison, the robots' policies are fixed and deterministic.

**Assumption 2:** The robots under evaluation are substantially mechanically similar: differences in overt behavior are influenced more significantly by differences in policy than by differences in hardware.

**Assumption 3:** Differences in policy are correlated with differences in overt behavior.

If these conditions are not met in a particular multi-robot system, the approach may not be appropriate. But the assumptions are reasonable for the conditions of this research, namely: experiments conducted on mechanically similar robots built on the same assembly line. Additionally, the robots all run identical control system software. The control systems running on the robots differ only in the data specifying each agent's policy. The comparison of these policies is the crux of the approach.

### 5.3.1 Example: multi-robot foraging

The objective is to show how behavioral difference can be evaluated by examining differences in robots' behavioral coding. Before proceeding, an example encoding is

**Figure 5.25**: An example task for a multiagent robot team. The robots are to collect red and blue objects and place them into colored bins. The task is similar to the 1997 AAAI Mobile Robot Contest's "Find Life on Mars" event. The object in the foreground is a "black rock" obstacle.

presented. Although this example describes a particular robot architecture, this is only for illustration, the method is applicable to other architectures as well.

Consider how behaviors could be designed for a team of foraging robots (Figure 5.25). The task is to collect colored objects (red) and place them into colored bins (red and blue). For this example, one agent will be programmed to place the objects in the red bin, while the other will deposit them in the blue bin.[2]

In this approach to behavioral configuration, the agent is provided several behavioral assemblages that correspond to steps in achieving the task (e.g. *wander*, *acquire*, *deliver*, and so on). Binary perceptual cues are used to sequence the robot through the steps in achieving the task.

The agents are provided with the perceptual features enumerated in Table 5.2. At the behavior selection level, the robot's perception can be represented by four bits (one bit per perceptual feature). Given the perceptual state, the robot selects from one of the four behaviors listed in Table 5.3. Decomposing the task into a state/action space enables a robot's policy to be enumerated by pairing perceptual states with actions. Some of the 16 states are never actually encountered since it is impossible for an agent to be simultaneously in the red and blue delivery zones.

[2]This task is a simplified version of the task for robots in the AAAI-97 contest. The simplification is necessary in order to allow a complete enumeration of the robots' policies.

**Table 5.2**: Perceptual features available to the foraging robots. Each feature is equivalent to one bit of information; the entire perceptual state is a four-bit value.

| perceptual feature | meaning |
|---|---|
| `red_visible` | a red attractor is visible. |
| `red_in_gripper` | a red attractor is in the gripper. |
| `close_to_red_bin` | close enough to the red delivery area to drop an attractor in it. |
| `close_to_blue_bin` | close enough to the blue delivery area to drop an attractor in it. |

**Table 5.3**: Behaviors the robots select from in accomplishing the foraging task.

| behavior | robot activity |
|---|---|
| *wander* | Search the environment for attractors. |
| *acquire_red* | Proceed to the closest red object and grasp it. |
| *deliver_blue* | Go the the blue delivery area. |
| *deliver_red* | Go to the red delivery area. |

Using this approach, policies for the two robots are now described. One of the robots collects red objects and places then in the red bin, while the other places them in the blue bin. The policy for robot $r_{\text{red}}$ is to search for red attractors using the *wander* behavior. When it sees an attractor, it activates the *acquire_red* behavior. Once it has grasped the object, it uses the *deliver_red* behavior to go to the red bin. Robot $r_{\text{blue}}$ is similar, except it delivers to the blue bin instead. Policies for the two agents are enumerated in Table 7.2. The behaviors in the center of the table are activated when the corresponding perceptual situations on the left are encountered.[3]

The actions selected by the agents described above differ in six of the states. In the case where the robots have a red object in their gripper but aren't close to a bin, they choose different actions (to either go to the red or blue bin). When they are close to the correct bin, they both drop the attractor and resume the *wander* behavior. The next section explains how a numerical value can be assigned this behavioral difference.

[3]Note that in the *wander* behavior, the robot's gripper opens automatically. A transition to the *wander* behavior causes the robot to drop the attractor and begin a new search.

**Table 5.4**: The policies of two foraging robots. Robot $r_{\text{red}}$ collects red objects and places them in the red bin, Robot $r_{\text{blue}}$ collects red objects and places then in the blue bin. Differences between the actions are listed on the right. The state bits represent, from left to right, `red_visible`, `red_in_gripper`, `close_to_red_bin`, and `close_to_blue_bin`. Impossible states are indicated with an asterisk (*).

| state | robot $r_{\text{red}}$ action | robot $r_{\text{blue}}$ action | response difference |
|---|---|---|---|
| 0000 | *wander* | *wander* | 0.0 |
| 0001 | *wander* | *wander* | 0.0 |
| 0010 | *wander* | *wander* | 0.0 |
| 0011* | *wander* | *wander* | 0.0 |
| 0100 | *deliver_red* | *deliver_blue* | 1.0 |
| 0101 | *deliver_red* | *wander* | 1.0 |
| 0110 | *wander* | *deliver_blue* | 1.0 |
| 0111* | *wander* | *wander* | 0.0 |
| 1000 | *acquire_red* | *acquire_red* | 0.0 |
| 1001 | *acquire_red* | *acquire_red* | 0.0 |
| 1010 | *acquire_red* | *acquire_red* | 0.0 |
| 1011* | *acquire_red* | *acquire_red* | 0.0 |
| 1100 | *deliver_red* | *deliver_blue* | 1.0 |
| 1101 | *deliver_red* | *wander* | 1.0 |
| 1110 | *wander* | *deliver_blue* | 1.0 |
| 1111* | *wander* | *wander* | 0.0 |

### 5.3.2 Definition of behavioral difference

To facilitate the discussion, the following additional symbols and terms are defined:

- $i_j$ is $r_j$'s **perceptual state**.
- $a_j$ is the action (behavioral assemblage) selected by $r_j$'s control system based on the input $i_j$.
- $\pi_j$ is $r_j$'s policy; $a_j = \pi(i_j)$.
- $p_j^i$ is the number of times $r_j$ has encountered perceptual state $i$ divided by the total number of times all states have been encountered. Experimentally, $p_j^i$ is computed *post facto*.

The approach is to evaluate behavioral difference by comparing the robots' policies. The two foraging robots introduced earlier, for example, exhibit behavioral differences that are reflected in and caused by their differing policies. In the terminology introduced above, $i$ represents the perceptual features an agent uses to selectively activate behaviors. In the case of the foraging robots, assign a bit to each perceptual feature, so, for example, $i = 0001$ indicates that only the last perception (`close_to_blue_bin`) is activated. For the foraging robots $a$ is the activated behavior (e.g. *wander*, *deliver*).

**Definition 1:** $r_a$ and $r_b$, are **absolutely behaviorally equivalent** iff they select the same behavior in every perceptual state.

In complex systems with perhaps thousands of states and hundreds of actions it may also be useful provide a scale of equivalence. This would allow substantially similar agents to be grouped in the same cluster even though they differ by a small amount. The approach is to compare two robots, $r_a$ and $r_b$, by integrating the differences between their responses, $\mid \pi_a(i) - \pi_b(i) \mid$ over all perceptual states $i$. If the action is a single-dimension scalar, as in a motor current for instance, the difference can be taken directly. However, complex actions like *wander* and *acquire* are treated as nominal values with response difference defined as 0 when $\pi_a(i) = \pi_b(i)$ and 1 otherwise. This approach is often used in classification applications to quantify difference between nominal variables (e.g. eye color, presence or absence of a tail, etc.). Using this notation, a simple behavioral difference metric can be defined as:

$$D'(r_a, r_b) \quad = \quad \frac{1}{n} \int \mid \pi_a(i) - \pi_b(i) \mid di \tag{5.5}$$

or for discrete state/action spaces:

$$D'(r_a, r_b) \quad = \quad \frac{1}{n} \sum_i \mid \pi_a(i) - \pi_b(i) \mid \tag{5.6}$$

where $\frac{1}{n}$ is a normalization factor to ensure the difference ranges from 0 to 1. In the case of the discrete sum, $n$ corresponds to the number of possible states. If $r_a$ and $r_b$ select identical outputs ($a$) in all perceptual states ($i$), then $D'(r_a, r_b) = 0$. When $r_a$ and $r_b$ select different outputs in all cases $D'(r_a, r_b) = 1$. In the numerical taxonomy literature, this difference is called the *mean character difference* [SS73]. The calculation parallels the idealized evaluation chamber procedure introduced earlier (Figure 5.24).

Equations 5.5 and 5.6 weigh differences equally across all perceptual states. This may be problematic for agents that spend large portions of their time in a small portion of the states. Consider two foraging robots that differ only in their reaction to blue attractors. If, in their environment, no blue attractors are present the agents would appear to an observer to have identical policies.

There may be other important reasons certain states are never visited. In learning a policy, for instance, the robots might discover in early trials that certain portions of the state space should be avoided due to large negative rewards. Because these portions of the space are avoided, the agents will not refine their policies there, but avoid them entirely. It is entirely possible for the agents to differ significantly in these portions of the space even though they may appear externally to behave identically.

To address this, the response differences in states most frequently visited should be emphasized while those that are infrequently experienced should be de-emphasized. This is accomplished by multiplying the response difference in each situation by the proportion of times that state was visited by each agent $(p_a^i + p_b^i)$. Formally, **behavioral difference** between two robots $r_a$ and $r_b$ is defined as:

$$D(r_a, r_b) = \int \frac{(p_a^i + p_b^i)}{2} \mid \pi_a(i) - \pi_b(i) \mid di \tag{5.7}$$

or in discrete spaces

$$D(r_a, r_b) = \sum_i \frac{(p_a^i + p_b^i)}{2} \mid \pi_a(i) - \pi_b(i) \mid \tag{5.8}$$

When $r_a$ and $r_b$ select differing outputs in a given situation, the difference is normalized by the joint proportion of times they have experienced that situation.

As an example of how behavioral difference is calculated, suppose the robots introduced earlier are evaluated in an experimental run.[4] During the experiment, the number of times each agent visits each state is recorded. This log, along with the response differences listed in Table 7.2 can be used to compute the behavioral difference between the two agents. The calculation is illustrated in Table 5.5. The number of times each agent visited each state is enumerated, then used to compute $p^i$ for each robot for each state. The normalized behavioral difference at each state is listed in the right column, and summed at the lower right. The value in the lower right-hand corner, 0.55, is the behavioral difference between robots $r_{\text{red}}$ and $r_{\text{blue}}$.

The measure of behavioral difference provides for the following definitions:

[4]This example experiment is for illustrative purposes only.

**Table 5.5**: Sample evaluation of the behavioral difference between the two agents whose policies are listed in Table 4.3. The number of times each state was visited by each agent is listed and used to compute $p^i$ for each state for each robot. In turn, the proportion of visits to each state is used to normalize the response difference between the agents. Note: these values were not gathered from experiment, they are presented for example only.

| state | times $r_{\text{red}}$ visited | $p^i_{\text{red}}$ | times $r_{\text{blue}}$ visited | $p^i_{\text{blue}}$ | difference | normalized difference |
|---|---|---|---|---|---|---|
| 0000 | 100 | 0.1 | 100 | 0.1 | 0.0 | 0.00 |
| 0001 | - | - | 100 | 0.1 | 0.0 | 0.00 |
| 0010 | 100 | 0.1 | - | - | 0.0 | 0.00 |
| 0011 | - | - | - | - | 0.0 | 0.00 |
| 0100 | 200 | 0.2 | 200 | 0.2 | 1.0 | 0.20 |
| 0101 | - | - | 200 | 0.2 | 1.0 | 0.10 |
| 0110 | 200 | 0.2 | - | - | 1.0 | 0.10 |
| 0111 | - | - | - | - | 0.0 | 0.00 |
| 1000 | 100 | 0.1 | 100 | 0.1 | 0.0 | 0.00 |
| 1001 | - | - | 100 | 0.1 | 0.0 | 0.00 |
| 1010 | 100 | 0.1 | - | - | 0.0 | 0.00 |
| 1011 | - | - | - | - | 0.0 | 0.00 |
| 1100 | 100 | 0.1 | 100 | 0.1 | 1.0 | 0.10 |
| 1101 | - | - | 100 | 0.1 | 1.0 | 0.05 |
| 1110 | 100 | 0.1 | - | - | 1.0 | 0.00 |
| 1111 | - | - | - | - | 0.0 | 0.00 |
| totals | 1000 | 1.0 | 1000 | 1.0 | 6.0 | 0.55 |

**Definition 2:** $r_a$ and $r_b$, are $\epsilon$**-equivalent** iff $D(r_a, r_b) < \epsilon$.

**Definition 3:** $\equiv_\epsilon$ indicates $\epsilon$-equivalence, $r_a \equiv_\epsilon r_b$ means $r_a$ and $r_b$ are $\epsilon$-equivalent.

**Definition 4:** A robot society, $\mathcal{R}$, is $\epsilon$**-homogeneous** iff for all $r_a, r_b \in \mathcal{R}$, $r_a \equiv_\epsilon r_b$

$\epsilon$ in these definitions is closely tied to the parameter $h$ used in $C_u$ clustering. A classification at taxonomic level $h$ will consist of $h$-homogeneous clusters.

### 5.3.3 Limitations of the approach

An important limitation of this approach to evaluating behavioral difference is the requirement for an agent's policy to be represented as a deterministic function, e.g. $a = \pi(i)$. This is reasonable for the analysis of policies developed using reinforcement learning techniques since once learning is complete, the policies are fixed. But the approach does not address robots utilizing FSAs for behavioral sequencing. An FSA might generate a different output (action) in the same perceptual state, depending on the sequence of inputs up to that point. To address the problem a quantitative technique for comparing FSAs is required. One avenue of approach would be a comparative analysis of the "languages" (actually sequences of perceptual state) accepted by two agents under evaluation. This is beyond the scope of the present investigation however.

Another potential problem is the implicit assumption that two robots being compared sequence through the state space at the same rate. As an example, two robots might have the same policy but because one of them is equipped with a very slow computer it changes state much less frequently. An analysis of behavioral difference based solely on policy difference could not differentiate between these two agents. The problem could be addressed by augmenting the difference metric with an additional "computational resource" factor. The issue was not a problem in this research because each robot is provided with identical computing resources.

## 5.4 Discussion and summary

Researchers in multiagent systems need new tools for measuring agent difference and diversity in the artificial societies they study. Without such metrics it is impossible to evaluate the quantitative impact of task on diversity or to correlate diversity with performance. To address these issues, this chapter introduces

- a mathematical expression for the **behavioral difference** between two robots,
- rigorous definitions of **behavioral homogeneity** and **heterogeneity** in multi-robot teams, and
- **social entropy**, a new measure of robot team **behavioral diversity**.

Behavioral difference is evaluated by comparing the policies of robots and normalizing the differences with respect to the proportion of times the two agents experience each perceptual state. The behavioral difference measure can be used in turn by clustering algorithms to divide the society into groups of similar agents.

The field of numerical taxonomy provides is a rich source of methods for this work because biologists face many of the same challenges. The hierarchic clustering methods used in taxonomy are advantageous because the generated hierarchies provide more information about the structure of the society than a simple clustering. $C_u$ clustering, a hierarchic overlapping clustering algorithm was adopted for this research.

The diversity of a robot society is computed using the classification of agents into groups. The social entropy (diversity) of a society is computed based on the number of groups and the size of each group. Use of the metric is illustrated in several example evaluations.

The value of social entropy as a diversity metric may be limited when difference between agents is continuous. Simple entropy does not account for the degree of difference between agents, or the structure of the society in the classification space. To address this, a more comprehensive measure, hierarchic social entropy is introduced. In short, hierarchic social entropy is the average entropy of a society across all taxonomic levels ($0 \leq h \leq 1$).

Both simple and hierarchic social entropy are used in the experiments reported in later chapters of this work. Simple entropy is employed when obvious distinctions between agents can be made or when the behavioral difference metric does not apply (as in the case of agents sequenced using FSAs). However, when continuous differences can be computed, the hierarchic social entropy measure is utilized.

# Chapter 6

# Diversity in Robot Foraging

This chapter addresses several important issues concerning performance and diversity in foraging robot teams. In particular, the following questions are investigated:

- Is behavioral diversity correlated with performance in foraging?
- Is it possible for agents using reinforcement learning to master foraging tasks?
- Assuming agents can learn to forage, will they converge to the same or diverse behavioral solutions?
- How does the choice of reinforcement function impact performance and diversity of learning teams?

To address these questions empirically, the methodology introduced in Chapter 3 is followed in the examination of human-designed and learning foraging strategies. Several well-known human-designed foraging strategies are implemented and evaluated in simulation. Learning teams utilizing three types of reinforcement function are

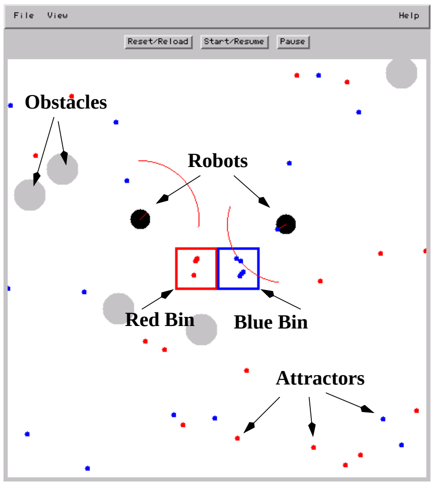



**Figure 6.1**: Simulation of the multi-forage task. Robots are represented as black circles; arcs indicate the visual sensing range. Obstacles are drawn as gray circles. The small discs are attractors. The robots deliver the attractors to the color-coded squares representing delivery areas.

trained and evaluated in simulation and compared with the hand-coded strategies. Finally, the behaviors are validated through implementation on mobile robots.

The remaining sections of this chapter describe the research in steps according to the methodology presented in Chapter 3.

## 6.1 Task and performance metric

The first step in the methodology is task specification. The forage task for a robot is to wander about the environment looking for items of interest (attractors). Upon encountering an attractor, the robot moves towards it and grasps it. After attachment, the robot returns the object to a specified home base. Foraging has a strong biological basis. Many ant species, for instance, perform the forage task as they gather food. Foraging is also an important subject of research in the mobile robotics community; it relates to many real-world problems [Ark92, ABN93, BA95a, GM97, FM97, Bal98a]. Among other things, foraging robots may find potential use in mining operations, explosive ordnance disposal, and waste or specimen collection in hazardous environments (e.g. the Mars Pathfinder rover).

Most robotic foraging tasks investigated to date involve the collection of attractors of a single type and their delivery to a single destination. This basic task is referred to as *simple foraging*. Simple foraging is an important robotic capability, but many practical industrial and military tasks call for more functionality. Consider, for example, a janitorial robot responsible for collecting and sorting recyclable trash objects into glass, aluminum and paper bins. Similarly, many assembly and construction tasks involve collecting parts or materials and placing them in a specific location. These more complex tasks are referred to as *multi-foraging* tasks. In general, **the multi-foraging task calls for several types of objects to be collected and placed in specific locations according to type**. Here *multi* refers to the multiple types of object to deliver, not the number of robots engaged in the task. An example simulation of robots executing a multi-foraging task is presented in Figure 6.1.

### 6.1.1 Example foraging robot systems

Several multi-robot foraging systems have been designed and implemented at Georgia Tech. The systems have been utilized in the study of communication in multiagent systems, cooperation in simple foraging and cooperation in multi-foraging tasks [BBC+95, BA95a]. Two of the resulting systems were entered in recent American Association for Artificial Intelligence (AAAI) Mobile Robot Competitions.

Ganymede, Io and Callisto, Georgia Tech's multi-robot entry in the AAAI-94 competition, were the first multi-agent system to compete and win a AAAI event. The robots are simple foragers; they collect trash objects and deliver them to a wastebasket (Figure 6.2, left). The custom-built robots recognize trash objects, wastebaskets and each other using color vision. Once they find an attractor, they move towards it and grasp it, then proceed to a wastebasket for delivery. Since the grippers cannot be raised, the robots leave the trash objects near the wastebasket. In spite of scoring penalties incurred because the trash was not deposited *in* the wastebaskets, the robots placed first in the contest.

A two robot team, Lewis and Clark, were entered in the "Find Life on Mars," event at AAAI-97. In this multiagent task robots search an obstacle-strewn environment for colored attractors. The robots are to grasp the attractors and deliver them to appropriately colored bins. In the competition, points were awarded when the robots deposited attractors in the correct bin. Figure 6.2 right, illustrates one of the robots competing in the AAAI-97 contest (the second robot is obscured by the

**Figure 6.2**: Two foraging robot teams developed at Georgia Tech participated in recent AAAI Mobile Robot Competitions. Georgia Tech entered and won both competitions with multi-robot teams. Left: in 1994, the "Clean Up the Office" task required robots to collect trash and deliver it to wastebaskets. Right: in 1997 the robots gather differently colored attractors and deposit them in color-coded bins. The author led the design, construction and programming of both multi-robot systems.

bin). Two of the behavior-based strategies evaluated later in this chapter were used in the contest. The robots placed first in three of four events, including both the multiagent challenge and final rounds. The AAAI-97 task, with slight variations, is adopted as the multi-foraging task for this research.

### 6.1.2 Performance metric and task classification

**Performance in the multi-foraging task is measured as the number of attractors collected and properly delivered by the robots in a 10 minute trial.** In terms of the taxonomy introduced in Chapter 5, this task has the following characteristics:

- `TIME_LIM` because performance is measured over a fixed period,
- `RESOURCE_LIM` because as agents collect objects, the the availability of attractors is reduced.,
- `OBJECT_BASED` since performance is based on the location of objects, not agents,
- `COMP_INT` because robots on the team compete for access to attractors among themselves,
- `SINGLE_AGENT` since an individual agent can perform positively, even though multiple agents may provide improved performance, and
- `SENSOR_LIM` since agents only have a limited view of the environment.

Several environmental parameters affect the rate at which the agents collect and deliver the attractors:

- **Number of attractors:** Since performance is measured as the number of attractors collected, more attractors available for collection may tend to provide for increased performance. In simulation runs there are 40 attractors, 20 of each type (red and blue). No team (hand-coded or learning) was able to reach the upper bound of 40 objects collected in 10 minutes.

- **Obstacle coverage:** Higher obstacle density can lead to degraded performance because the robots must slow down and/or take a longer route around hazards to deliver attractors. In simulation runs, each playing field includes five 1 $m^2$ obstacles (5% coverage). The AAAI Competition field included approximately 10 rock piles varying from about 0.5 $m^2$ to 1 $m^2$. In most laboratory runs, no obstacles other than another robot and the arena boundaries were present.

- **Playing field size:** Larger search areas may lead to a decrease in performance. In simulation, the field measures 10 by 10 meters. At the AAAI Competition, the field was a hexagon measuring approximately 8 by 8 meters. Runs in Georgia Tech's Mobile Robot Lab were conducted in a 5 by 10 meter area.

- **Number of robots:** In most cases, increasing the number of robots on a team improves performance. There is some concern however, that as the number of robots increases, interference between the agents will degrade performance [GM97]. In simulation experiments the number of agents is varied from one to eight. In laboratory runs one and two agents were used. At the AAAI Competition two robots were employed.

The next section explains the development of multi-robot behaviors for the multi-foraging task.

## 6.2 Behavioral design of hand-coded strategies

A schema-based reactive control system is used for robot programming. In this approach, the agent is provided several pre-programmed behavioral assemblages that correspond to steps in achieving the task (e.g. *wander*, *acquire*, *deliver*, and so on [AM94]). Binary perceptual features (also referred to as perceptual triggers) are used to sequence the robot through steps in achieving the task. Selection of the appropriate behavior, given the situation, may be hand-coded or discovered by the robot through reinforcement learning.

**Table 6.1**: Behaviors the robots select from in accomplishing the foraging task. Hand-coded agents sequence from one behavior to another according to a fixed strategy. Learning agents must discover a satisfactory strategy autonomously.

| behavior | robot activity |
|---:|---|
| *wander* | Search the environment for attractors. |
| *stay_near_home* | Search the home zone for attractors. |
| *acquire_blue* | Proceed to the closest blue object and grasp it. |
| *acquire_red* | Proceed to the closest red object and grasp it. |
| *deliver_blue* | Go the the blue delivery area. |
| *deliver_red* | Go to the red delivery area. |

While the focus of the dissertation is diversity in *learning* robot systems, hand-coded non-learning teams play an important role as well. The performance of human-designed teams establishes a baseline for comparison with results from learning systems. Also, hand-coded strategies provide additional data points regarding the relationship between diversity and performance in foraging. Three hand-coded strategies for foraging are implemented and evaluated for performance and diversity:

- **Behaviorally homogeneous:** all the robots collect all the different types of attractor and deliver them to corresponding color-coded delivery areas.
- **Territorial:** In this scheme one robot, referred to as the *sorting* agent, is responsible for collecting attractors within a three meter circle centered on the homebase. The other agents search at a distance from the homebase. When these robots find an attractor, they hand it off at the boundary of the "home zone." Final delivery is then completed by the sorting agent. This is a behaviorally heterogeneous approach.
- **Specialization by color:** half the robots specialize in collecting one type of attractor while the rest specialize in collecting the second type. Specialization by color is a heterogeneous strategy as well.

### 6.2.1 Behavioral repertoire

To ensure a fair comparison between the various hand-coded and learning systems, a fixed repertoire of behaviors is utilized across all implementations. A range of behaviors was developed to support several foraging strategies and to avoid bias towards any particular approach. The repertoire is suitable for building behaviorally homogeneous foraging teams as well as territorial and other heterogeneous strategies.

Agents utilizing distinct strategies differ in the order they activate behaviors. Hand-coded agents proceed deliberately from behavior to behavior as they accomplish the task while the learning agents must discover which behavior to activate when. The behaviors developed for foraging teams are summarized in Table 6.1, and described in detail below:

- *wander:* move randomly about the environment in search of attractors. Upon encountering an attractor, hand-coded agents automatically transition to an appropriate *acquire* behavior. Learning systems, in contrast, discover an appropriate follow-on behavior on their own. Motor schemas active in the *wander* assemblage are
  - **noise:** high gain, moderate persistence to cover a wide area of the environment.
  - **avoid_static_obstacles:** gain sufficiently high to avoid collisions.
  - **avoid_robots:** gain sufficiently high to avoid collisions.
  - **avoid_home_zone:** moderate gain to move the agent away from the home zone. This is important for the roaming territorial foragers since it keeps them from interfering with the sorting agent.
- *stay_near_home:* similar to the *wander* assemblage, but with an additional attractive force to keep the agent close to the homebase. This assemblage is utilized in the territorial strategy by a sorting agent. Active schemas include
  - **noise:** high gain, moderate persistence to cover the home zone.
  - **avoid_static_obstacles:** gain sufficiently high to avoid collisions.
  - **avoid_robots:** gain sufficiently high to avoid collisions.
  - **stay_near_homebase:** moderate gain to keep the agent close to homebase.
  - **avoid_delivered_attractors:** repulsion centered on the delivery area with a small radius of influence and moderate gain to keep the agent from pushing over delivered objects.
- *acquire_red:* move towards the closest visible red attractor. When close enough to grasp the attractor, hand-coded agents close their gripper and transition to a *deliver* assemblage. Learning agents must learn which follow-on behavior to activate. Motor schemas activated in this assemblage include
  - **noise:** low gain, to deal with local minima endemic to potential field approaches.
  - **avoid_static_obstacles:** gain sufficiently high to avoid collisions.
  - **avoid_robots:** gain sufficiently high to avoid collisions.
  - **move_to_red_attractor:** high gain to move the agent to the attractor.

- **avoid_delivered_attractors:** repulsion centered on the delivery area with a small radius of influence and moderate gain to keep the agent from pushing over delivered objects.

- *acquire_blue:* move towards the closest visible blue attractor. The same schemas are activated as in the *acquire_red* assemblage, except that **move_to_blue_attractor** replaces **move_to_red_attractor**.

- *deliver_red:* move towards the red delivery area. When close enough to deposit the attractor in the delivery area, hand-coded agents open their gripper and transition to one of the *wander* assemblages. Territorial agents are programmed to drop attractors on the boundary of the home zone. Motor schemas activated in this assemblage include

  - **noise:** low gain, to deal with local minima endemic to potential fields approaches.
  - **avoid_static_obstacles:** gain sufficiently high to avoid collisions.
  - **avoid_robots:** gain sufficiently high to avoid collisions.
  - **move_to_red_bin:** high gain to move the agent to the delivery area.
  - **avoid_delivered_attractors:** repulsion centered on the delivery area with a small radius of influence and moderate gain to keep the agent from pushing over delivered objects.

- *deliver_blue:* move towards the blue delivery area. The same schemas are activated as in the *deliver_red* assemblage, except the **move_to_blue_bin** schema is activated instead of **move_to_red_bin**.

The next section describes perceptual features used to sequence the behaviors for foraging.

### 6.2.2 Perceptual features

A perceptual feature is a single, abstracted bit of environmental or sensor state germane to the robot's task (e.g. whether or not the robot is holding an attractor in its gripper). Agents must decide on the basis of these environmental cues which behavior to activate at each point in time. The hand-coded teams are programmed as Finite State Automatons (FSAs) that sequence from one state to another. Each state of the FSA determines which behavior is to be activated, with transitions between behavioral states triggered when particular perceptual features are activated. This approach is called *perceptual sequencing* [AM94].

**Table 6.2**: Perceptual features available to the foraging robots. Each feature is one bit of environmental state; the entire perceptual state is a nine-bit value.

| perceptual feature | meaning |
|---:|---|
| `red_visible` | a red attractor is visible. |
| `blue_visible` | a blue attractor is visible. |
| `red_visible_outside_homezone` | a red attractor is visible outside the three meter radius home zone. |
| `blue_visible_outside_homezone` | a blue attractor is visible outside the home zone. |
| `red_in_gripper` | a red attractor is in the gripper. |
| `blue_in_gripper` | a blue attractor is in the gripper. |
| `close_to_homezone` | the agent is within 3 meters of the homebase. |
| `close_to_red_bin` | close enough to the red delivery area to drop an attractor in it. |
| `close_to_blue_bin` | close enough to the blue delivery area to drop an attractor in it. |

A fixed set of perceptual features are utilized across all implementations to ensure a fair comparison between the various foraging systems. The perceptual features for foraging are cataloged in Table 6.2. In addition to the features advising the robot whether an attractor is visible, there are also features indicating whether attractors are visible outside the home zone. The visibility cues are used to allow territorial agents to search for attractors inside or outside the zone while ignoring the others. The `close_to_homezone` feature is used by territorial robots as a signal to drop an attractor on the boundary of the zone so that a sorting robot can complete the final delivery.

Learning agents are provided the features as a 9-bit integer. This conglomeration of features is referred to as the agent's *perceptual* state. Altogether there are 512 potential perceptual states. As a practical matter however, some states never occur. It is impossible, for instance, for a robot to be both in the red delivery area and outside the home zone simultaneously.

Note that because the perceptual state is an ambiguous representation of the agent's situation, the decision problem for the robot is *partially observable* [LCK95]. This means that many distinct situations are perceived as equivalent by the robot. For instance, as an agent delivers an attractor, it might perceive itself as being in the same state whether it is 5 meters or 4 meters from the delivery area.

### 6.2.3 Hand-coded sequencing strategies

As mentioned earlier, hand-coded teams are programmed using FSA descriptions for sequenced behavior. The various strategies were all built from the same repertoire of behaviors. The different strategies, homogeneous, specialize by color and territorial, either utilize a different subset of the behaviors or activate the behaviors in a different sequence.

#### Homogeneous Foraging

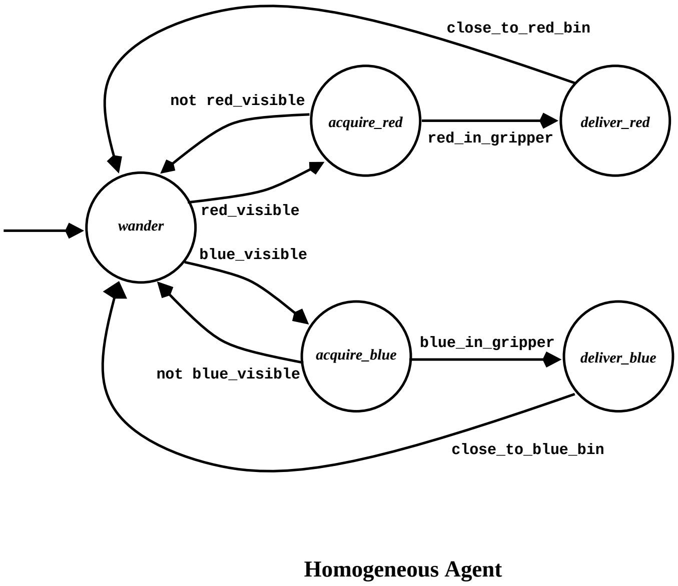



**Figure 6.3**: An FSA representing the homogeneous foraging strategy. This kind of agent can collect all types of attractor.

An obvious approach to the design of a multi-robot multi-foraging team is to program each agent to complete the entire task on its own. This strategy is referred to as *homogeneous* because all the agents are programmed with the same behavior. This approach was used by Georgia Tech's robots in the AAAI-94 competition and in research concerning multiagent communication [BBC$^+$95, BA95a]. The homogeneous approach provides fault-tolerance because when one or more agents fail, the remaining robots can still accomplish the task.[1]

[1]In fact, at the AAAI-94 contest, one of Georgia Tech's three robots failed during the final round. Fortunately, the two remaining robots were able to finish the task.

An FSA representing the homogeneous strategy is shown in Figure 6.3. Each agent begins with the *wander* behavior activated, roaming about the environment in search of attractors. When a robot encounters an attractor, either the `red_visible` or `blue_visible` perceptual feature is triggered, causing the agent to transition to the corresponding *acquire_red* or *acquire_blue* behavior. Upon capturing an attractor, a robot returns back to homebase using one of the *deliver* behaviors. Finally, upon reaching the corresponding delivery area, the agent drops the attractor and transitions back to *wander*.

### Specialization by Color

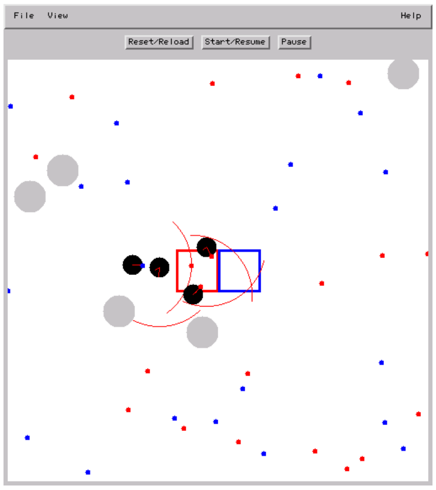



**Figure 6.4:** Inter-robot interference. In this simulation three robots are attempting to deliver attractors to the left (red) delivery area simultaneously. Interference reduces the rate at which agents can deliver attractors.

When several robots simultaneously attempt to deliver attractors to the same delivery area, they may interfere with one another and degrade performance. Figure 6.4 shows an example simulation where robots impede each other's progress as they attempt to deliver attractors to the left (red) delivery area. One way to reduce interference and potentially improve performance is to partition the task so that responsibility for collecting red and blue attractors is divided among the robots. Half of the agents are responsible for collecting the red objects and the other half for blue. This way, the chance of a "traffic jam" at either delivery area is reduced.

FSAs for these specialized agents are illustrated in Figure 6.5. All agents start with the *wander* behavior activated. They begin to search the environment for at-

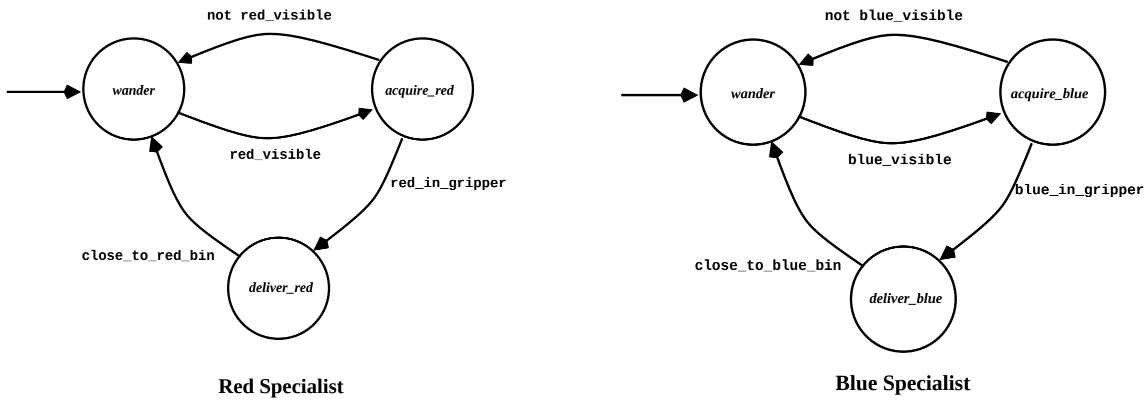



**Figure 6.5**: FSAs representing specialized behaviors for foraging. The FSA on the left shows the sequence behaviors are activated in for an agent specializing in collecting red attractors. The FSA on the right shows the sequence for blue specialists.

tractors. What follows depends on whether the agent is a red specialist or a blue specialist. Red specialists ignore blue attractors, but when they encounter a red attractor while in the *wander* phase, they immediately transition to the *acquire_red* behavior. Similarly, blue specialists ignore red attractors. After acquiring an attractor, the agents deliver it to the appropriate delivery area using one of the *deliver* behaviors, then they switch back to *wander* to search for new items. An additional transition is provided for situations where an agent loses sight of the attractor. In that event it transitions back to *wander*.

### Territorial Specialization

Goldberg, Fontan and Matarić have investigated several heterogeneous foraging strategies as a way to minimize inter-agent interference [GM97, FM97]. In Fontan's approach the search area is divided equally between the agents [FM97]. They hand off collected attractors from area to area, with the last robot completing delivery to the homebase. Alternately, Goldberg proposes caste and pack foraging strategies [GM97]. In the pack scheme, each agent is arbitrarily assigned a place in the "pack hierarchy." Agents higher in the hierarchy are permitted to deliver attractors before the others. In the caste approach, only one agent completes the final delivery; the other robots leave their attractors on the boundary of a designated "home zone." Goldberg's results indicate that interference per unit time is maximized in homogeneous foraging and minimized in pack foraging. In spite of the fact that interference is maximized in the homogeneous system, the researchers report homogeneous foraging

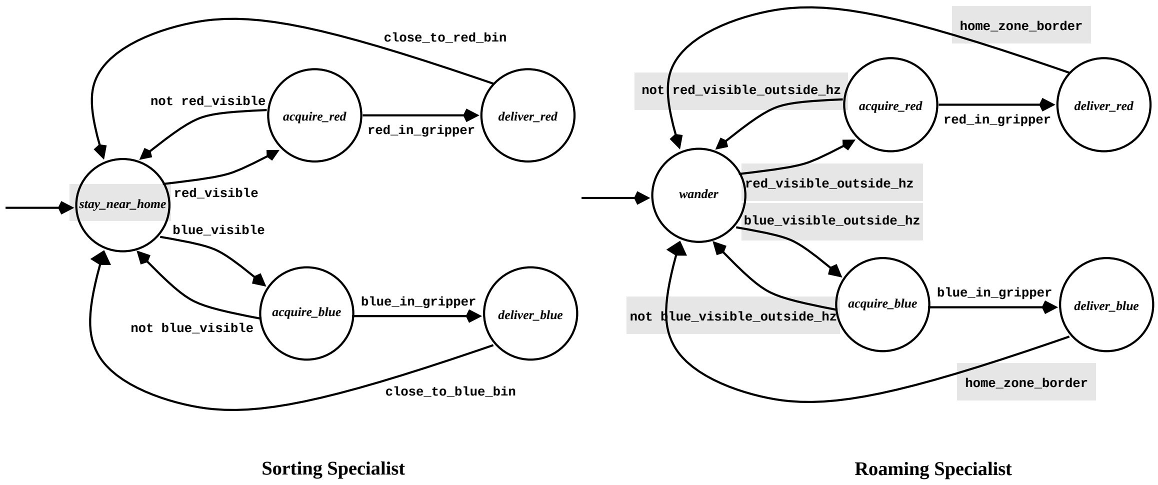



**Figure 6.6**: The Territorial behavioral sequences for foraging. FSA on the left shows the sequence of behaviors for an agent that remains close to the homebase, completing the delivery of the attractors. Agents using the strategy on the right search for attractors away from the home zone and deliver them to the home zone boundary. Differences from the homogeneous strategy are highlighted.

teams perform best.

A territorial/caste strategy is adopted for this investigation. In this scheme, one robot remains close to the homebase in the "home zone," delivering attractors that other agents deposit on the zone's boundary. Figure 6.6 shows the FSAs for robots in this system. The sequencing strategies for the agents are similar to the approach for homogeneous foragers (Figure 6.3). One significant difference is that the sorting agent utilizes a *stay_near_home* behavior rather than *wander* while searching for attractors. This results in the agent staying close to the delivery areas for sorting. The other agents are also similar to the homogeneous strategy, except they are triggered to drop attractors at the boundary of the home zone instead of depositing them in the delivery areas.

## 6.3 Performance of hand-coded strategies in simulation

Now the performance of the hand-coded systems are examined experimentally. The JavaBots system was utilized for simulation and mobile robot experimentation [Bal98, Bal]. Behaviors coded in JavaBots may be run in simulation, and without modification, on Nomadic Technologies' Nomad 150 mobile robots. Statistical results are gathered by running the robot behaviors in thousands of simulation trials.

In simulation, each robot is a kinematically holonomic vehicle (a simulated Nomad 150) which is controlled by one of the behavioral systems described above. Simulated motor and sensor capabilities are based on performance of the physical robots. The robots can detect hazards with sonar out to a range of nine meters. Attractors can be detected visually out to three meters across a 90 degree field of view.

Each type of control system under investigation was evaluated using one to eight simulated robots in five different randomly generated environments. The environments differ in the arrangement of hazards and attractors. The arena is 10 by 10 meters and includes five randomly placed 1 $m^2$ obstacles for 5% obstacle coverage. There are 20 each of red and blue attractors distributed about the environment for collection. For hand-coded systems 100 trials were run in each environment, or 4,000 runs for each control strategy, and 12,000 total.

Time is measured in seconds. Since reactive control systems are very fast, several thousand control cycles are completed each second. The simulation is allowed to proceed faster than real time with each control cycle fixed at 200 milliseconds (simulation time). Each trial runs for 10 simulated minutes or 30,000 control cycles.

### 6.3.1 Task performance

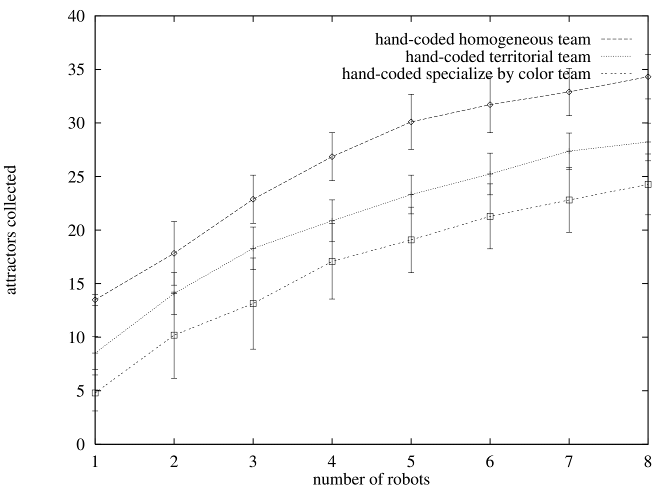



**Figure 6.7**: Performance of the hand-coded teams versus size of the team. Higher numbers are better; error bars indicate 95% confidence intervals. The homogeneous teams perform best in all cases.

Average performance for each of the three systems is plotted versus the number of robots per team in Figure 6.7. Performance is measured as the total number of attractors properly delivered by the team in 10 minutes; larger numbers are better, with 40 being the best possible. The plotted values are determined by computing the average performance of the teams in each of the five randomly generated environments over 100 trials.

In all cases, performance increases monotonically with the number of agents. The data also show that regardless of team size, the homogeneous strategy performs best, followed by the territorial method and finally the specialize-by-color approach. Of the hand-coded systems evaluated **homogeneous systems perform best in this foraging task**. These results confirm those of other researchers in simple foraging [FM97, GM97].

### 6.3.2 Factors impacting performance

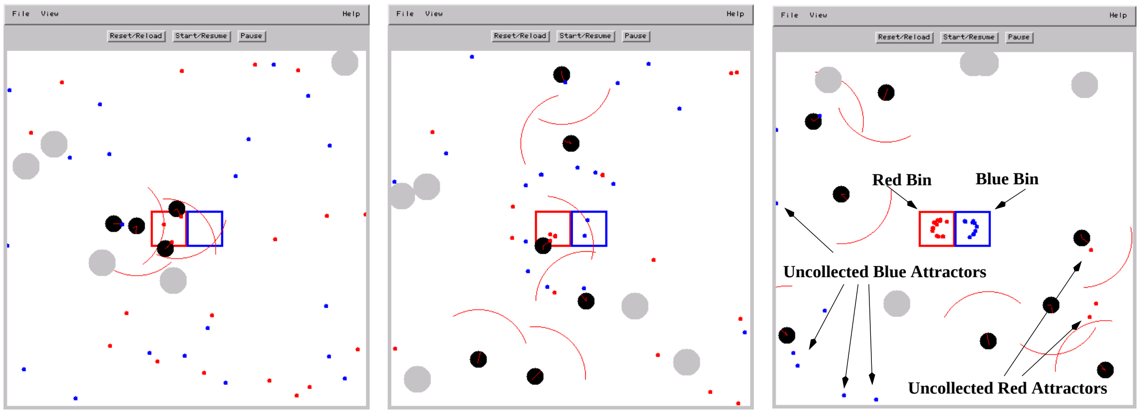



**Figure 6.8**: Simulations highlighting some of the factors that impact performance in hand-coded teams. From left to right: interference in a homogeneous team, an over-worked sorting robot, attractors left uncollected by agents specializing in one color of attractor.

As previously mentioned, inter-robot interference is a concern for homogeneous systems, while the heterogeneous strategies were specifically designed to reduce interference. Interference does occur during the delivery phase in both homogeneous and specialize-by-color strategies (Figure 6.8, left). This study does not include a quantitative measure of interference, but qualitative observations are consistent with results reported in [GM97]; namely that there is more frequent interference between agents in the homogeneous strategy than in other systems. Still, performance is best in homogeneous systems.

In the case of territorial foraging, robot-robot interference occurs much less frequently, but another factor limits performance. In most trials, the roaming agents quickly deliver a large number of the attractors to the boundary of the home zone, but the single sorting robot cannot always keep up. In simulations with seven and eight agents it is not uncommon for undelivered attractors to remain in a ring around the home zone at the end of the trial (Figure 6.8, center). Even though the number of sorters is constant (one), the territorial foraging experiments illustrate how the ratio of sorters to roamers impacts performance. The ratio varies from 1:0 to 1:7 as the number of agents goes from 1 to 8. In the 1:0 case, the sorting agent "starves" because it quickly finds all the nearby attractors. Conversely, in the 1:7 case, the sorter is overworked; there are many more attractors for it to deliver than it is able to.

A different sort of problem crops up for the specialize-by-color teams. Towards the end of trials for these agents there are often uncollected red attractors on the right side of the field and uncollected blue attractors on the left (Figure 6.8, right). This occurs because the agents inadvertently segregate themselves geographically to the left or right depending on whether they collect red or blue attractors. After delivering a blue attractor, for instance, a blue-collecting agent is more likely to remain on the same (right) side of the field as the blue bin. Because of this there are no red-collecting agents on the right side of the field and vice-versa. In large robot teams the robot-robot repulsion employed as part of the *wander* behavior serves as an additional force preventing the agents from diffusing from one side to the other. In addition to segregation, the specialize-by-color agents occasionally interfere with one another when delivering attractors to the same delivery area simultaneously. These factors combine to drive performance down in the specialize-by-color teams.

### 6.3.3 Diversity

The diversity of the three systems is plotted versus the size of the team in Figure 6.9. Diversity is measured using social entropy as described in Chapter 5. Recall that the entropy of a system is determined by grouping the agents according to behavior, then evaluated for diversity based on the number and size of the groups.

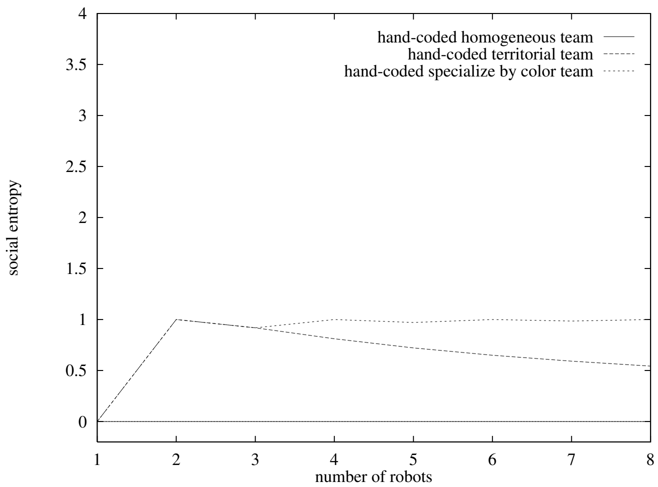



**Figure 6.9**: Diversity, as measured by social entropy, versus size of the team for hand-coded teams. Larger numbers indicate greater diversity.

A higher entropy indicates greater diversity. The homogeneous teams always exhibit zero diversity, while the heterogeneous teams vary in entropy from 0.54 to 1.0.

The territorial system always contains one unique robot (the sorting agent that stays near homebase), while the rest are identical. In this system, entropy is 1.0 for two agents, then gradually declines as the number of agents increases. The value drops to 0.54 for eight robots.

For even numbers of robots the specialize-by-color team consists of half red-collecting robots and half blue-collecting robots; this equates to an entropy of 1.0. For odd numbers of robots the entropy is slightly less than 1.0. The value oscillates about and gradually converges to 1.0. This is borne out in the graph (Figure 6.9).

One goal of this research is to determine the relationship between diversity and performance in multiagent tasks; is the relative diversity of two multiagent teams a predictor of their relative performance? This question is addressed using *Spearman's Rank-Order Correlation Test* [PTVF88]. Spearman's test measures correlation between pairs of data points, where each pair reflects the ranking of each item according to separate metrics. In this case, we compare ranking in performance with ranking in diversity. The correlation value, $r$ ranges from $-1$ (negatively correlated), to 0 (uncorrelated) to 1 (positively correlated). Statistical significance of the correlation is indicated by the probability that the same correlation could have occured by chance. The equations used in Spearman's test are described in Appendix B.

Consider the plots of robot system performance and diversity in Figures 6.7 and

6.9. For each robot team size ($> 1$) the systems can be ranked by diversity and performance.[2] Spearman's test indicates that **diversity and performance are strongly negatively correlated in this foraging task**, with $r = -0.92$. Greater diversity is associated with lower performance. The probability of the null hypothesis, that the rankings occur by chance, is 0.000043.

## 6.4 Design of learning strategies

This section describes the implementation of learning foraging strategies and investigates how the choice of reinforcement function impacts performance and diversity in a learning team. The general approach for training a team is to provide each agent a reward function that generates feedback at each movement step regarding the agent's progress. Q-learning is utilized as the means of associating actions with state [WD92]. The learning agents are initialized with random Q-tables, thus random, poorly performing policies. Since each agent begins with a different policy the teams are initially maximally diverse. They improve their policies using the reinforcement functions described in the following subsection.

The learner seeks to maximize its reward, discounted over time, by selecting appropriate actions according to its situation. At each step the learner activates one of its six behaviors according to a 9-bit perceptual state summarizing its situation. By utilizing this approach, agents can learn the task automatically. They can also adapt to changes in the environment and failures of the other agents. More details on Q-learning are provided in Chapter 2.

Learning teams are given the same suite of behaviors and perceptual features as the hand-coded teams. In this way differences in performance can be attributed to the strategy or policy in use rather than the behaviors or state information available to the agent.

### 6.4.1 Reinforcement functions for foraging

To evaluate the impact of reinforcement on diversity and performance in learning teams, three reward functions are evaluated:

[2]Since diversity has no meaning for a single agent system, only teams of two or more agents are considered. Ties are declared in cases where values are exactly the same or when confidence intervals overlap.

- **Local performance-based reinforcement:** each agent is rewarded individually when it delivers an attractor.
- **Global performance-based reinforcement:** all agents are rewarded when any agent delivers an attractor.
- **Local shaped reinforcement:** each agent is rewarded progressively as it accomplishes portions of the task [Mat97].

Robot teams using these reward systems are evaluated in terms of performance, learning rate and diversity. Diversity and performance of the learning teams are also compared against the corresponding values for the hand-coded systems.

In both types of performance-based reinforcement the reward is tied directly to the performance metric; in this case, attractor delivery. A performance-based reward structure is advantageous for the designer because it allows her to succinctly express the task for an agent. There is no need to enumerate how the task should be carried out (as was necessary in the hand-coded teams). Instead, the agents learn behavior sequences autonomously. In contrast, heuristic or shaped reinforcement functions provide rewards to the agent as it achieves parts of the task; for instance, when grasping an attractor, when heading for the delivery area, and when depositing it in the delivery area.

Assuming the task proceeds in discrete steps, the local performance-based reinforcement function for foraging at timestep $t$ is:

$$R_{\text{local}}(t) = \begin{cases} 1 & \text{if the agent delivered an attractor at time } t-1 \\ -1 & \text{otherwise} \end{cases} \tag{6.1}$$

The global performance-based function is defined as:

$$R_{\text{global}}(t) = \begin{cases} 1 & \text{if any agent delivered an attractor at time } t-1 \\ -1 & \text{otherwise} \end{cases} \tag{6.2}$$

The global function will reward all team members when an attractor is delivered. The global function is implemented using an inter-robot communication scheme that allows the agents to communicate their individual rewards. In terms of the reinforcement function taxonomy developed in Chapter 4, $R_{\text{global}}$ and $R_{\text{local}}$ are similar in that they are both `INTERNAL_SOURCE`, `PERFORMANCE`, `DELAYED` and `DISCRETE` reward functions. Of course they differ in locality; one is `LOCAL` while the other is `GLOBAL`

A potential problem with these reward functions is that the reinforcement is *delayed*. The agent must successfully complete a sequence of steps before receiving a reward. This makes credit assignment in the intervening steps more difficult. To address this issue, Matarić proposes an alternate reward scheme where the agent is provided intermediate rewards as it carries out the task [Mat97]. The agent is not only rewarded for delivering an attractor, but also for picking one up, for moving towards a delivery area when it is holding an attractor, and so on. This heuristic strategy is referred to as *shaped reinforcement*. The reward function is defined as a sum of three sub-functions, as follows:

$$R_{\text{shaped}}(t) = R_{\text{event}}(t) + R_{\text{intruder}}(t) + R_{\text{progress}}(t) \tag{6.3}$$

$R_{\text{event}}(t)$ encapsulates the reward for events like delivering an attractor or dropping it in the wrong place. $R_{\text{intruder}}(t)$ is used to punish the agent for prolonged interference with other agents. Finally, $R_{\text{progress}}(t)$ is activated when the agent is holding an attractor, and rewards the agent for moving towards the delivery point. $R_{\text{event}}(t)$ is defined more formally as:

$$R_{\text{event}}(t) = \begin{cases} 1 & \text{if delivered attractor at time } t-1 \\ 1 & \text{if picked up attractor at time } t-1 \\ -3 & \text{if dropped attractor outside bin at time } t-1 \\ -1 & \text{otherwise} \end{cases} \tag{6.4}$$

Matarić sets $R_{\text{event}}$ to 0 in the default case, instead of -1 as above. The choice was made to use -1 here since Q-learning converges more quickly with negative rewards before task completion. $R_{\text{progress}}(t)$ is defined as:

$$R_{\text{progress}}(t) = \begin{cases} 0.5 & \text{if holding attractor and moving towards bin at time } t-1 \\ -0.5 & \text{if holding attractor and moving away from bin at time } t-1 \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

Since the individual behaviors used in this work already include a provision for agent avoidance, $R_{\text{intruder}}(t)$ is not used. $R_{\text{shaped}}$ is an `INTERNAL_SOURCE`, `HEURISTIC`, `IMMEDIATE`, `DISCRETE` and `LOCAL` reward function.

For performance-based rewards, Q-learning is not sensitive to the specific value of reinforcement. For instance, (+1, -2) will work as well as (+20, -0.5). An important constraint, however, is that performance is maximized if and only if reward is maximized (this is the definition of performance-based reward). Heuristic rewards (like shaped reinforcement) do not usually meet this criteria.

## 6.5 Performance of learning strategies in simulation

Statistical results were gathered by running the robot behaviors in thousands of simulation trials. The same simulation system, parameters and other experimental conditions used for evaluation of the hand-coded systems were also employed in the evaluation of the learning systems. Each type of learning system under investigation was evaluated using one to eight simulated robots in five randomly generated environments. As in the evaluation of hand-coded systems, performance is the number of attractors collected in 10 minutes. 300 trials were run in each environment, 12,000 runs for each control strategy and 36,000 overall.

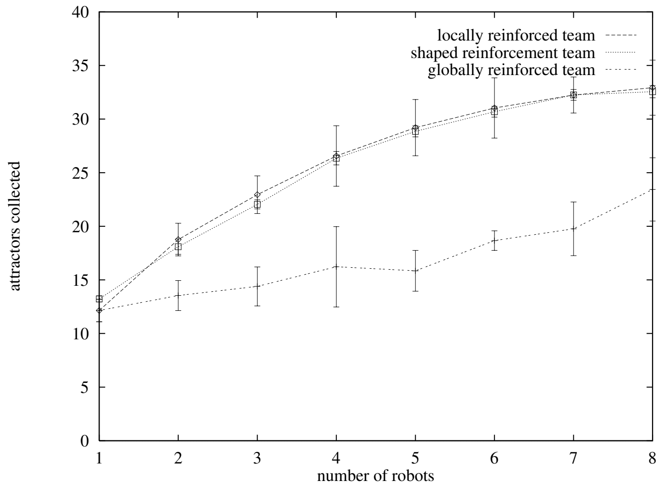



**Figure 6.10**: Performance for learning systems versus the number of robots on a team.

### 6.5.1 Task performance

Agents are able to learn the task using all three types of reinforcement. In fact for local and shaped reinforcement, the teams perform as well as hand-coded teams. In most cases the agents converge to stable performance after about 150 trials.

A plot of the average performance for each learning system versus the number of agents on the team is presented in Figure 6.10. The local performance-based and heuristic reinforcement systems perform best. Performance in the globally reinforced system is worse than the other learning teams. Note that the performance plots for teams using local and shaped rewards are nearly identical and that one's confidence interval overlaps the other's mean value. For this task there is no statistically significant difference in performance between the two approaches.

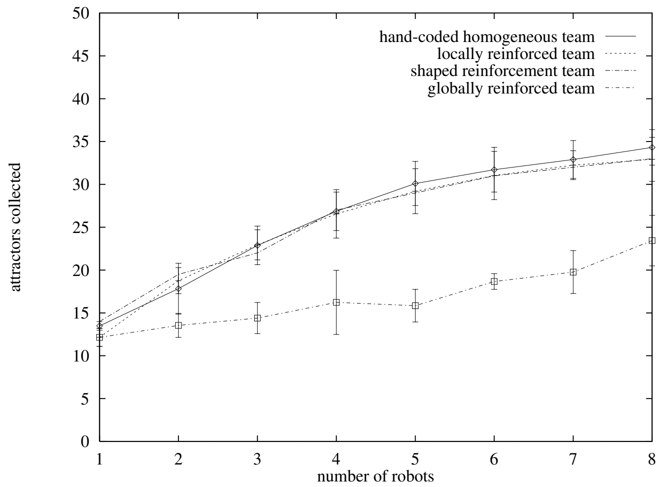



**Figure 6.11**: Performance of the best learning and non-learning systems combined in one graph.

When the performance of learning and non-learning systems are viewed together (Figure 6.11), one can see that there is no statistically significant difference between the homogeneous hand-coded systems and the best learning systems. **Local and shaped reinforcement systems perform as well as the best hand-coded systems.**

### 6.5.2 Learning rate

The rate at which agents converge to stable policies is evaluated by tracking the number of times an agent's policy changes during each trial. A policy change is a revision of the agent's Q-table such that it will select a different action in some perceptual state. The average number of policy changes per trial is graphed for each system in Figure 6.12. The figure shows plots for systems with eight agents. **All three reinforcement strategies show good convergence properties.**

Agents using shaped reinforcement converge at a rate about equal to the globally reinforced agents. With shaped reinforcement, however, the average number of changes converges to and remains at zero after 220 trials. The globally reinforced systems converge to zero after 290 trials. The locally reinforced teams do not fully converge, but settle to around five changes/trial. Overall, shaped reinforcement

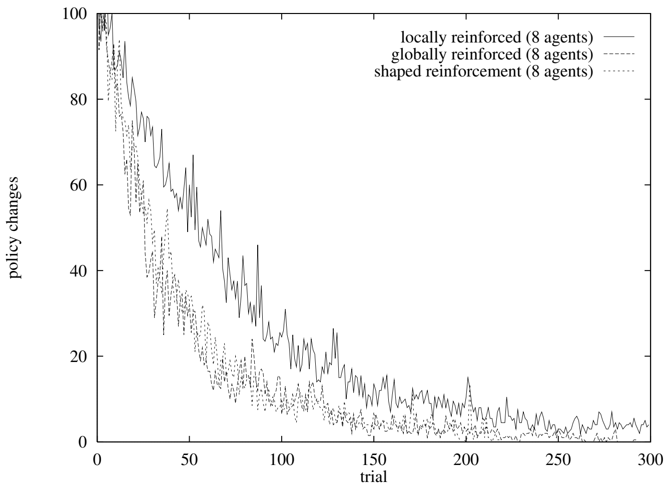



**Figure 6.12**: Convergence for learning systems, measured as policy changes per trial; lower numbers indicate convergence to a stable policy. Shaped reinforcement shows the best convergence properties (reaching zero after 220 trials).

converges most quickly.

Learning rate can also be evaluated by observing how the performance of a system changes over time. Plots of performance versus trial number are shown in Figure 6.13. All three systems increase their performance at a rapid rate over the first 50 trials. Performance with global reinforcement levels off at around 20 after 50 trials. The locally rewarded teams level off after approximately 150 trials at between 30 and 35 attractors. The shaped reinforcement systems reach the same level, but they stabilize more quickly. These findings roughly correspond with the convergence data.

### 6.5.3 Diversity

As in the hand-coded teams, diversity is measured using social entropy. Unlike the hand-coded teams however, agents are not as easily categorized into different behavioral groups. To facilitate categorization, the behavioral difference metric introduced in Chapter 5 is used, with similar agents being grouped together as a caste.

An example of how the robots are grouped according to their behavioral difference is provided in Figure 6.15 and Figure 6.16. In the example agent similarity matrix (Figure 6.15) each entry represents the behavioral difference between two robots. Difference can vary from 0.0 (no difference) to 1.0 (maximum difference). For instance, in the second column and first row, one can see that agent 0 and agent 1 differ by 0.196. Agents are considered $\epsilon$-equivalent if their behavioral difference is

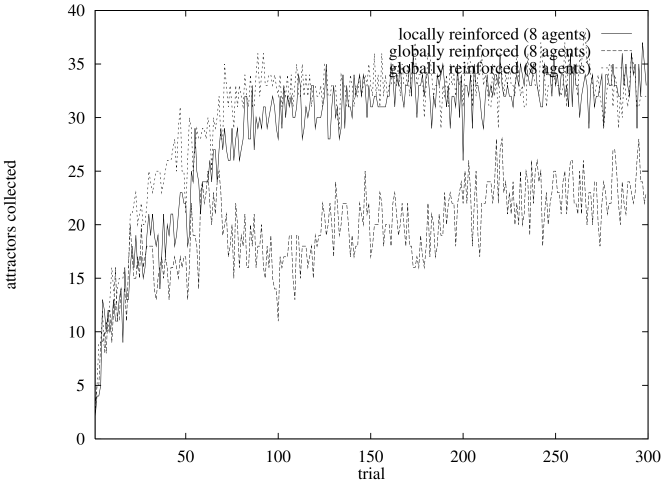



**Figure 6.13**: Performance versus trial number for learning systems with eight robots. These plots indicate how performance improves as the agents learn over time. Local and shaped-reinforcement teams stabilize at about the same level, with globally-reinforced teams performing worst.

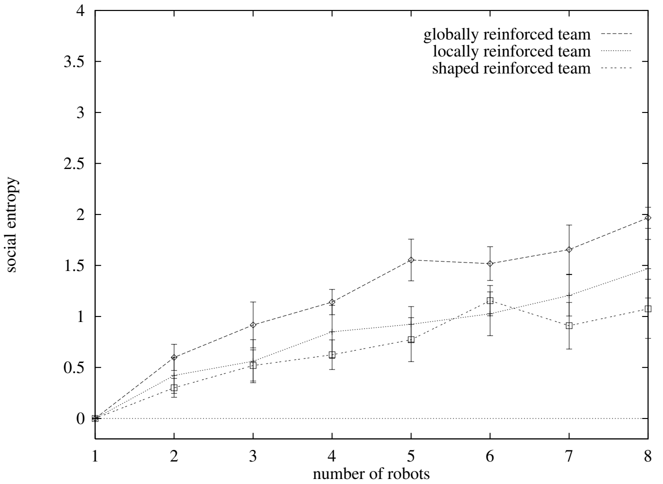



**Figure 6.14**: Hierarchic social entropy versus size of the team for learning teams; larger numbers indicate greater diversity, error bars indicate 95% confidence intervals. Shaped-reinforcement generates the least diverse teams, while globally-reinforced teams are the most diverse. For some robot team sizes differences between the globally and locally reinforced teams is not significant.

less than or equal to $\epsilon$. If we let $\epsilon = 0.2$, agents 0 and 1 are grouped in the same caste. Continuing with the comparisons, two castes emerge. The first caste contains agents 0, 1 and 2, while the other contains agents 2 and 3. These relationships are illustrated in Figure 6.16. Social entropy, a measure of the randomness in the system, is computed using the size and number of castes. All other things being equal, a system with more castes will have a greater entropy. For this example the entropy is 0.97.

To avoid a bias in measurement that might be caused by selecting a particular $\epsilon$, the entropy of a robot system is computed using hierarchic social entropy (covered in detail in Chapter 5). Essentially, entropy is computed for the system at each taxonomic level (value of $\epsilon$) then averaged across all of them. The result is the *hierarchic social entropy* of the system.

| | **agent 0** | **agent 1** | **agent 2** | **agent 3** |
|---|---|---|---|---|
| **agent 0** | 0.000 | 0.196 | 0.203 | 0.571 |
| **agent 1** | 0.196 | 0.000 | 0.378 | 0.586 |
| **agent 2** | 0.203 | 0.378 | 0.000 | 0.199 |
| **agent 3** | 0.571 | 0.586 | 0.199 | 0.000 |

**Figure 6.15**: Agent similarity matrix. Each entry in the table indicates the behavioral difference between two corresponding agents. This four robot team was trained using shaped reinforcement. With $\epsilon = 0.2$ the entropy of this system is 0.97.

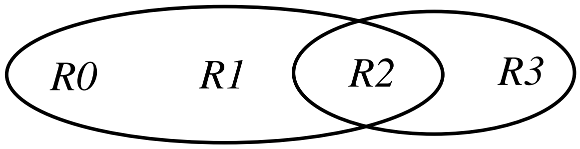



**Figure 6.16**: Division of the example team into castes based on behavioral difference. In this case $\epsilon = 0.2$.

Using the approach outlined above, diversity is determined for robot groups using each type of reinforcement, in each of the five random environments for 2 to 8 robots. The hierarchic entropy of the three types of learning systems is plotted versus the size of robot teams in Figure 6.14. In all cases with two or more agents, the globally reinforced teams are most diverse. In all but one case the teams using shaped reinforcement are the least diverse and locally reinforced teams lie between the two extremes.

As in the hand-coded teams, **diversity and performance are negatively correlated in learning teams**. For each robot team size ($> 1$) the systems are ranked by diversity and performance.[3] The rankings are evaluated using Spearman's Rank-Order Correlation Test (see Appendix B for details). Spearman's test indicates the rankings are strongly negatively correlated, with $r = -0.96$. The probability of the null hypothesis being true (that the rankings occur by chance) is 0.000028.

## 6.6 Implementation on mobile robots

To verify the simulation results, the hand-coded and learning systems were ported to Nomad 150 mobile robots. Because the control systems are implemented in JavaBots, they can run in simulation and on hardware; the same behaviors and features can be utilized on mobile robots as in simulation (the hardware platform is covered in more detail in Chapter 3). Four of the configurations evaluated in simulation were run on mobile robots:

1. One robot using the homogeneous foraging strategy.
2. Two robots using the homogeneous foraging strategy.
3. One robot using a policy learned with local reinforcement.
4. Two robots using a policy learned with local reinforcement.

Additionally, homogeneous and heterogeneous systems were evaluated in the AAAI-97 Mobile Robot Competition task. The AAAI-97 experiments are covered in Section 6.6.1.

The same behaviors prototyped and evaluated in simulation were utilized in these trials on mobile robots. Snapshots of one of the robots executing *wander*, *acquire* and *deliver* behaviors in the laboratory are presented in Figure 6.17. The robots utilize a passive gripper to collect attractors. The gripper is designed so that a captured object remains under the robot's control until the robot drops it by backing up.

The Mobile Robot Laboratory provides an arena measuring approximately 5 meters by 10 meters for the robot experiments. A total of 20 attractor objects, 10 of

[3] Differences in performance between teams using local and shaped rewards are not statistically significant because one's confidence interval overlaps the other's average value (Figure 6.10). Likewise, in several cases the confidence intervals for diversity overlap. For ranking purposes in cases where overlap occurs, the overlapping strategies are ranked as ties.

each type (red and green), were distributed randomly about the lab for each trial. Both the size of the arena and the number of attractors available for collection are halved in comparison with the environment used in simulation experiments.

**Figure 6.17**: Nomad 150 robot equipped with passive gripper demonstrates three foraging behaviors. From left to right: *wander*, *acquire_blue* and *deliver_blue*.

In the first set of experiments, robots were programmed to execute the homogeneous strategy introduced earlier in the chapter (configurations 1 and 2). Five trials of 10 minutes were run for each number of robots. At the end of each trial, performance was evaluated as the total number of attractors properly delivered. Performance is summarized in Table 6.3.

In laboratory evaluations qualitative behavior was essentially identical to that of homogeneous teams in simulation. As in simulation, the agents occasionally interfered with one another when they deliver attractors to the same bin. In these experiments with 20 attractors, each robot routinely collected and delivered 8 objects. As expected, two robots perform better than a single robot. Performance is slightly worse than the same strategy in simulation experiments. The decrease is most likely due to the reduced number of attractors available for collection (20 versus 40).

Learning systems were evaluated in a second set of experiments (configurations 3 and 4). Performance was evaluated before and after learning using local performance-based rewards on one and two robots. In each case, the robots were initialized with a random policy (the behavior for each situation is set randomly), then evaluated in a 10 minute trial. The policies were transferred to the simulation system and trained over 300 trials. After training, the policies were transferred back to the robots for another evaluation. The process was repeated five times for each number of robots. Performance of the robots running learned policies is summarized in Table 6.4. Snapshots of one of the learned policy trials are presented in Figure 6.18.

**Table 6.3**: Summary of performance in hand-coded foraging robot trials.

| configuration/trial | | attractors collected |
|---|---|---|
| **1 robot, homogeneous strategy** | trial 1 | 10.0 |
| | trial 2 | 8.0 |
| | trial 3 | 11.0 |
| | trial 4 | 12.0 |
| | trial 5 | 8.0 |
| | **average** | **9.8** |
| **2 robots, homogeneous strategy** | trial 1 | 15.0 |
| | trial 2 | 16.0 |
| | trial 3 | 17.0 |
| | trial 4 | 12.0 |
| | trial 5 | 15.0 |
| | **average** | **15.0** |

As in simulation the robots perform much better after the learning phase. However, they do not collect as many attractors as comparable simulated systems. Again, this is likely due to the reduced number of attractors available for collection. **As in simulated systems, learning systems perform nearly as well as hand-coded robots.**

### 6.6.1 Experiments in the AAAI "Find Life on Mars" task

The AAAI competition task is slightly different from the task evaluated in the previous section. Results in the AAAI task are valuable because they serve to further establish the usefulness of behaviors prototyped in simulation and they highlight a situation where heterogeneous strategies are important. The results also illustrate how JavaBots facilitates reconfiguration of robot behavior.

#### Task differences

Rather than a delivery area for each type of attractor as in the task described earlier, in the AAAI task robots must deposit attractors in bins with doors. The doors of the bins are painted an identifying color to help the agents find them. The robots had to be equipped with active grippers to enable them to lift the objects and drop them in the bins. Also, some of the attractors *move*. The battery-powered squiggle balls roll around at about twice the maximum speed of the robot platforms. The robots were to sort the attractors according to whether they were "alive" (rolling)

**Table 6.4**: Summary of performance in learning foraging robot trials.

| configuration/trial | | performance before training | after |
|---|---|---|---|
| **1 robot, Q-learning, local reward** | trial 1 | 1.0 | 9.0 |
| | trial 2 | 0.0 | 10.0 |
| | trial 3 | 0.0 | 8.0 |
| | trial 4 | 0.0 | 7.0 |
| | trial 5 | 0.0 | 8.0 |
| | **average** | **0.2** | **8.4** |
| **2 robots, Q-learning, local reward** | trial 1 | 0.0 | 15.0 |
| | trial 2 | 1.0 | 15.0 |
| | trial 3 | 0.0 | 16.0 |
| | trial 4 | 1.0 | 14.0 |
| | trial 5 | 0.0 | 13.0 |
| | **average** | **0.4** | **14.6** |

or "dead."

In the first phase of the competition, the Challenge Round, the attractors and doors were matched in color, but in the Final Round there were six different colors for the attractors (three colors per bin). This presented a problem since the robots can only track three colors at a time. The solution, detailed below, was to utilize a heterogeneous strategy where one robot collects three types of attractor and the other collects the others.

### Behavioral design for the AAAI task

We now proceed with a description of the behavioral strategies employed on the robots; details of performance are covered in the next section. For the Challenge Round robots were programmed using the homogeneous multi-foraging strategy outlined in Section 6.2.3. Several problems were discovered when the strategy was first tested on the robots. First, because an agent's turret heading was coupled to the heading of its wheels, the gripper would not always be properly aligned with the bin door at the time of final delivery. Second, the robots sometimes approached the delivery bins from an oblique angle. This is a problem because docking is controlled by visual tracking and the doors could not always be detected from an oblique angle. Third, the black rock hazards were low enough to the ground that the robots could see over them. This led to situations where a robot could be drawn to an attractor

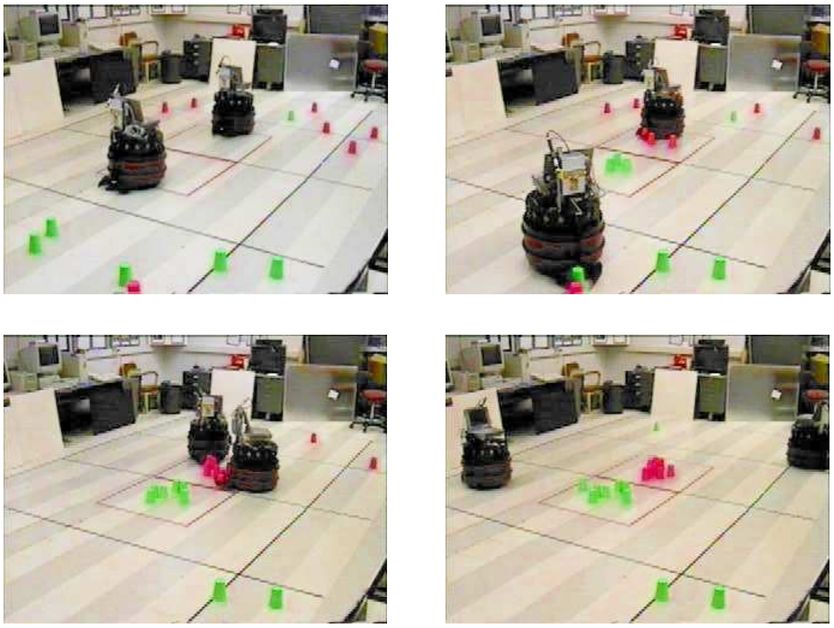


**Figure 6.18**: Two robots demonstrate learned foraging polices in the Mobile Robot Lab. A total of 16 attractors were collected in this 10 minute trial. Sequence is from top left to bottom right.

beyond a hazard and become stuck.

The problem with gripper alignment was solved by de-coupling the heading of the turret from the heading of the robot's wheels. The *delivery* behaviors were revised to have the turret always pointing towards the delivery door. It was also found that de-coupling is advantageous for *acquire* behaviors as well, especially when the attractor is near a hazard. In this case, the robot approaches the attractor slowly with occasional side to side motions. Unless the turret and base are de-coupled the gripper swings back and forth with the changes in heading. Also, in order for a robot to intercept a rolling attractor, it would have to be facing the attractor at all times. **These revisions were folded back into the behaviors for the simulation results reported earlier.**

The last two problems were addressed by adding additional behaviors and perceptual features to the behavioral repertoire. To ensure the robots approach bin doors head-on, two additional behaviors, *predock_red* and *predock_blue* were developed. Each of these behaviors draw the robot to a position one meter in front of the corresponding door. From the pre-dock position to final docking, the door is easily tracked visually. The schemas active in these behaviors are identical to those used in the *deliver* behaviors, except the destination is different. Upon reaching the pre-dock location, the agents are allowed to transition to the delivery behavior.

To address the problem of robots getting stuck while attempting to acquire an attractor on the other side of a hazard, a progress timer was added to the list of perceptual features. The idea is that if a robot does not acquire an attractor within a reasonable time (60 seconds was used), it must be stuck. When the timer, `no_progress`, is activated the agent transitions to a *reset_to_home* behavior. This draws it back towards the homebase, where it resets for another foraging run. The schemas active in the *reset_to_home* behavior are identical to those used in *deliver*, except the destination is different. The agent is held in the *reset_to_home* behavior for 20 seconds, at which point it transitions back to the *wander* behavior.

A complete FSA describing the homogeneous foraging configuration used at the competition is shown in Figure 6.19. As in the hand-coded homogeneous strategy introduced earlier, the agents begin using the *wander* behavior then transition between the behaviors as described above. Photographs of one of the robots sequencing through a delivery cycle are shown in Figure 6.20.

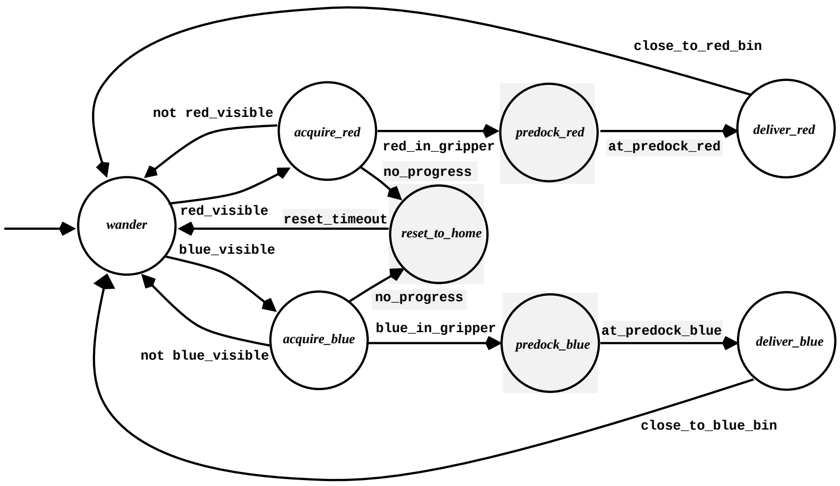



**Figure 6.19**: FSA representing the behavioral configuration used by Lewis and Clark at the AAAI-97 Mobile Robot Competition. Some additional behaviors and perceptual features (highlighted in gray shading) were added to address specific differences between the contest task and the general multi-foraging task.

Changes in the task for the Final Round presented additional challenges. The robots had to collect and deliver objects of six different colors instead of two as in the Challenge Round. This was a problem because the vision systems can only

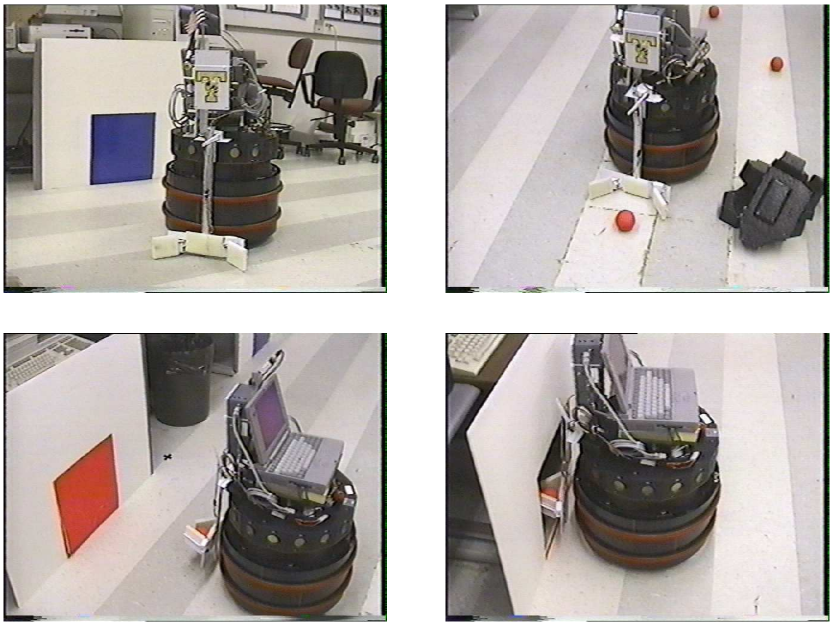


**Figure 6.20**: Robot with an active gripper demonstrates a delivery sequence in the Mobile Robot Laboratory. From top left to bottom right: *wander*, *acquire_red*, *predock_red* behavior, *deliver_red*.

track *three* colors, and at least one of those tracking channels has to be dedicated to detection of the delivery door. The issue was addressed by exploiting a heterogeneous foraging strategy. Each robot was programmed to specialize in the collection of three of the six types of attractor.

### Performance in the AAAI task

Lewis and Clark were evaluated in the Mobile Robot Laboratory and in the competition at AAAI-97. In laboratory evaluations (Figures 6.20 and 6.21) subjective behavior is essentially identical to that of homogeneous teams in simulation. As in simulation, the agents occasionally interfere with one another when they deliver attractors to the same bin. In experiments with 10 attractors distributed about the laboratory, the robots routinely collected eight of them in a 10 minute trial.

The robots participated in the first round of the contest, referred to as the Challenge round, using the homogeneous foraging strategy. In the first round the robots had difficulty detecting the rock hazards visually. The sonar sensors were not effective at detecting the hazards because they are mounted too high on the robot to detect the shorter rocks. The robots encountered the hazards on several occasions.

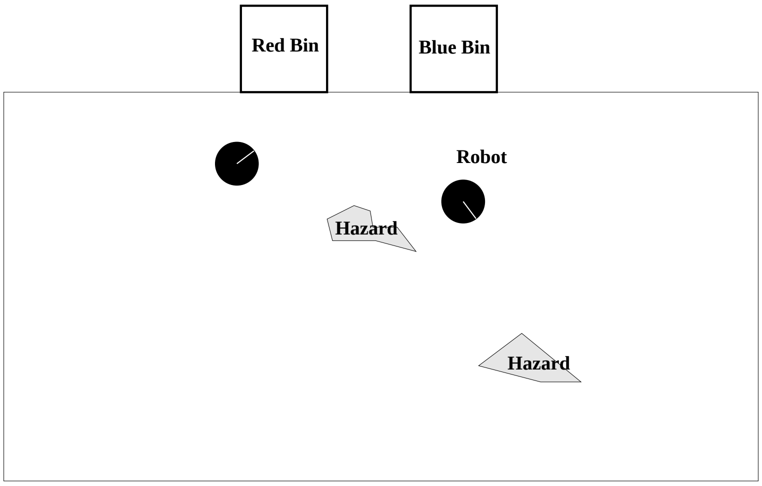



**Figure 6.21**: Diagram of laboratory arrangement for mobile robot experiments in the "Find Life on Mars" task.

In one case, one of the robot's grippers was ripped off the vehicle (fortunately this occurred towards the end of the trial). Despite this setback, the agents were able to deliver a significant number of attractors and win the Challenge Round. One of the robots even captured a squiggle ball – this was a rare event at the competition.

Between the Challenge and Final rounds the Georgia Tech team developed a solution to the difficulty of hazard detection: the ultrasonic sensors were repositioned to aim downward at a 45 degree angle. The hazards could now be reliably detected.

The robots were re-programmed for heterogeneous foraging in the Final Round to enable collection of all six types of attractor. Performance in the Final round was much improved over earlier trials. The robots picked up 10 attractors and placed 9 of them in the correct delivery bin. The success of a behaviorally heterogeneous team in this situation illustrates how computational limits of individual agents can necessitate diversity in a multi-robot solution. Each robot is potentially capable of detecting all six types of attractor, but computational limits of the embedded vision computers allow only three at one time – one robot cannot complete the entire task alone. In terms of the taxonomy presented in Chapter 4 this instance of the multi-foraging task is `MULTI_AGENT` instead of `SINGLE_AGENT`. Perhaps `MULTI_AGENT` tasks are more likely to require heterogeneous solutions.

## 6.7 Discussion and summary

This chapter reports several important new findings in multi-robot foraging research:

- A link between diversity and performance in foraging teams is reported for the first time: in both learning and hand-coded systems, diversity is negatively correlated with performance; homogeneous teams perform best.
- Teams trained using Q-learning perform as well as the best human-designed systems. This is the first direct comparison between learning and hand-coded foraging systems.
- This work is the first to uncover a relationship between choice of reinforcement function used to train robots and diversity in the resulting team. In these experiments:
  - Locally reinforced teams tend to converge to homogeneity.
  - Globally reinforced teams tend to converge to heterogeneity.
  - Heuristic or shaped reinforcement leads to homogeneity.
- This work is the first to investigate how the choice of reinforcement function impacts performance in foraging teams. In these experiments:
  - Teams trained with local heuristic (shaped) and local performance-based reinforcement perform best, on par with hand-coded systems.
  - Globally reinforced teams perform worst.
- Experiments show that in this task, heuristic or shaped reinforcement does not provide an advantage over more simply expressed performance-based reinforcement.

The conclusions outlined above are based on statistical analysis of thousands of simulation trials. The behaviors, perceptual features and behavioral sequences used in simulation were also verified on mobile robots. Hand-coded and learning systems were evaluated on one and two robots. Qualitatively, mobile robot behavior matches that predicted in simulation, including inter-agent interference and overall performance.

Among the hand-coded foraging systems examined, homogeneous teams perform best. It was noted that homogeneous foragers tend to interfere with one another as they deliver attractors to the delivery areas (interference in homogeneous foraging was also noted by Goldberg in [GM97]). To address this potential problem, territorial and specialize-by-color strategies were designed with the goal of reducing inter-agent interference. Even though interference is reduced in these heterogeneous teams, performance is *worse*. In fact, diversity is negatively correlated with performance in the hand-coded teams. (Spearman's $r = -0.92$ and prob $= 0.000043$).

In experiments with three separate reward strategies, reinforcement learning was demonstrated as a capable tool for training multi-robot foraging teams. Experimental results show that the choice of reinforcement function significantly impacts diversity and performance in learning robot foragers. Agents using local reinforcement strategies converge to more homogeneous societies and perform better than robots using a global reward structure. This is probably because local reinforcement rewards individuals for their actions, thus making reinforcement of the same state/action pair more likely in different agents. A link between local reinforcement and homogeneity is also present in learning soccer agents (Chapter 7). Agents using global reinforcement converge to more diverse and poorly performing societies in the foraging task. The advantages of homogeneous behavior in hand-coded systems are echoed in results with learning systems. In learning systems, diversity and performance are negatively correlated with $r = -0.96$ and prob $= 0.000028$.

In addition to the local and global **performance**-based reward structures, a local **heuristic**, or shaped reinforcement method was also evaluated. Agents trained using shaped reinforcement perform as well as the best human-designed team. In terms of performance and learning rate however, there is no advantage to shaped reinforcement over standard performance-based rewards.

# Chapter 7

# Diversity in Robot Soccer


This chapter describes research investigating specialization in learning robot soccer teams. Each agent is provided a common set of skills (motor schema-based behavioral assemblages) from which it builds a task-achieving strategy using reinforcement learning. The agents learn individually to activate particular behavioral assemblages given their current situation and a reward signal. This work was conducted following the methodology introduced in Chapter 3.

The experiments in JavaBots robot soccer simulations evaluate the agents in terms of *performance*, *policy convergence*, and *behavioral diversity*. As in foraging experiments (Chapter 6) the results show that in some cases robots will diversify by choosing heterogeneous behaviors. An interesting contrast with foraging results however is that diverse soccer teams perform better than homogeneous teams. The degree of diversification and the performance of the team depend on the reward structure. When the entire team is jointly rewarded or penalized (global reinforcement), teams tend towards heterogeneous behavior. When agents are provided feedback individually (local reinforcement), they converge to identical policies.

## 7.1 Task and performance metric

Robot soccer is an increasingly popular focus of robotics research [KAK+97]. It it is an attractive domain for multiagent investigations because a robot team's success against a strong opponent often requires some form of cooperation. Also, it is familiar to many audiences and it provides opportunities for diversity among the team members.


**Figure 7.1**: Simulated and real robot soccer: a match at the 1998 Robot World Cup in Paris (left) and a JavaBots simulation (right). Photograph courtesy Hiroaki Kitano.

The task is patterned after the official RoboCup rules for small-size league play [Com97]. Each team is composed of five robot players. Once play begins the teams attempt to push and/or kick the ball (an orange golf ball) into the opponent's goal. The game is played on a green field the size of a table tennis table. Boundaries are 10cm tall walls – the golf ball bounces back instead of going out-of-bounds. Goals are 50cm wide. When a goal is scored the ball is reset to the middle of the field and the players are re-positioned. Official RoboCup matches include two 10 minute halves. A photograph of a RoboCup soccer game is presented in Figure 7.1 (left).

In the Java-based soccer simulation used in this research (Figure 7.1, right) a robot's control system interacts with a well-defined sensor-actuator interface. The simulation proceeds in discrete steps. In each step the robots process their sensor data, then issue appropriate actuator commands. The simulation models physical interactions (robot, ball and wall collisions), sensors and motor-driven actuators.

When the ball is bumped by a robot it immediately accelerates and rolls away. The direction the ball rolls after being bumped varies randomly from -10 to +10 degrees off center. Rolling friction is modeled with constant deceleration after the bump. Dynamics are based on actual RoboCup robot performance [Sto98]. Each agent is provided the following sensors:

- **velocity sensor:** provides present heading and speed of the robot,
- **bump sensor:** returns a force vector in the direction of any bump,
- **ball position sensor:** provides an egocentric vector to the soccer ball,
- **defended goal sensor:** provides an egocentric vector back to the robot's own goal,
- **opponent goal sensor:** provides an egocentric vector the opponent's goal,
- **team sensor:** returns an array of egocentric vectors pointing to the robot's team members,
- **opponent sensor:** an array of egocentric vectors pointing to the robot's opponents,
- **score sensor:** indicates whether the team has just scored or was scored against,
- **robot ID:** a unique integer from 1 to the size of the team.

Robots are able to sense all information germane to the task. This approximates the sensor system available to many of the real robot teams competing at RoboCup; information is gathered by a video camera mounted above the playing field. Future revisions of the simulator may address challenges faced by autonomous robots without accurate global sensors, e.g. sensor noise, occlusion and field-of-view constraints.

The following actuator interface is provided to the control system:

- **set drive speed:** a real value from -1 to 1 is sent to the robot's drive motor, indicating how fast the robot should go.
- **set heading:** a real value from 0 to $2\pi$ is sent to the robot's steering actuator indicating the desired heading for the robot.
- **kick:** if the ball is near the robot's kick actuator it is immediately accelerated in the direction of the robot's heading.

Now consider the performance metric for soccer. How can we objectively evaluate a robot soccer team? In a human game the object is to have scored the most points when time runs out. It is only necessary to score one more point than the other team. Here, we take the stance that greater score differentials indicate better performance. Hence, the performance metric for robot teams is

$$P = S_{\text{us}} - S_{\text{them}} \tag{7.1}$$

where $S_{\text{us}}$ and $S_{\text{them}}$ are the scores of each team at the end of the game.

In terms of the taxonomy introduced in Chapter 4 this task and performance metric have the following characteristics:

- `TIME_LIM` because performance is measured over a fixed period (except for simplified soccer),
- `OBJECT_BASED` since performance is based on the location of an object, the ball,
- `COMP_EXT` because robots on the team compete for positive performance (goals) against an (external) opposing team,
- `COMP_INT` because robots on the team compete for goals amongst themselves,
- `MULTI_AGENT` since a single agent is unlikely to net a positive score differential against a multiagent opponent,
- `SENSOR_COMPLETE` since agents can sense all aspects of the environment germane to the task perfectly.

The first set of experiments in the investigation were conducted in slightly simplified soccer domain. The domain is simplified as follows: Teams are composed of four players instead of five. The goal spans the width of the field's boundary instead of a 50cm wide zone. Play is continuous; after a scoring event, the ball is immediately replaced to the center of the field without repositioning the agents. Another important difference in the simplified task is that there is no time limit imposed; play continues until a total of 10 points are scored (the simplified is not `TIME_LIM`). To distinguish between the two tasks the simplified version is referred to as *simplified soccer*, while the more complex task is *RoboCup soccer*.

## 7.2 Behavioral design

Behavior-based approaches are well suited for robot soccer since they excel in dynamic and uncertain environments. The robot behaviors described here are implemented in Clay (Chapter 3), an object-oriented recursive system for configuring robot behavior. Clay integrates primitive behaviors (motor schemas) using cooperative and competitive coordination operators. Both static and learning operators are available.

Experiments in soccer are conducted by engaging an *experimental* team against a fixed opponent *control* team in soccer contests. We begin by describing the control team's behavioral configuration.

Since the experimental team's performance is significantly impacted by the skill of its opponent, it is important to avoid variability in the control team's strategy to ensure consistent results. The control team will always follow a fixed policy against the teams under evaluation. The control team's design is based on the following observations. First, points are scored by bumping the ball across the opponent's goal. Second, robots must avoid bumping the ball in the wrong direction, lest they score against their own team. A reasonable approach then, is for the robot to first ensure it is behind the ball, then move towards it to bump it towards the opponent's goal. Alternately, a defensive robot may opt to remain in the backfield to block an opponent's scoring attempt.

**Table 7.1**: The control team's policy summarized as look-up tables. The 1 in each row indicates the behavioral assemblage selected by the robot for the perceived situation indicated on the left. The abbreviations for the assemblages are introduced in the text.

**Control Team Forward**

| **perceptual** | **assemblage** | | |
|---|---|---|---|
| **feature** | *mtb* | *gbb* | *mtb f* |
| *not behind_ball* | 0 | 1 | 0 |
| *behind_ball* | 1 | 0 | 0 |

**Control Team Goalie**

| **perceptual** | **assemblage** | | |
|---|---|---|---|
| **feature** | *mtb* | *gbb* | *mtb f* |
| *not behind_ball* | 0 | 1 | 0 |
| *behind_ball* | 0 | 0 | 1 |

Each robot selects from a set of behavioral assemblages to complete the task. The behaviors are sequenced to form a complete strategy. This style of behavior-based robot design, referred to as *temporal sequencing*, views an agent's strategy as a Finite State Automaton. Temporal sequencing is discussed in Chapter 3. The strategies may be equivalently viewed as lookup tables (Table 7.1). Here we focus on the lookup table view since it is also useful for discussing learned policies. The behavioral assemblages developed for these experiments, and the motor schemas activated are:

- *move_to_ball (mtb)*: The robot moves directly to the ball. A collision with the ball will propel it away from the robot. Individual motor schemas active in this assemblage include:

- **move_to_kickspot**: high gain to draw the robot to a point one-half of a robot radius behind the ball. If the robot bumps the ball from that location, the ball is propelled in the direction of the opponent's goal.
- **avoid_teammates**: gain sufficiently high to keep the robots on the team spread apart This schema was not activated in the simplified soccer experiments, but was found to be useful in later work.

- *get_behind_ball (gbb)*: The robot moves to a position between the ball and the defended goal while dodging the ball to avoid bouncing it in the wrong direction. Activated motor schemas are
  - **move_to_halfway_point**: high gain to draw the robot to a point halfway between the ball and the defended goal.
  - **swirl_ball**: a ball dodging vector with gain sufficiently high to keep the robots from colliding with the ball.
  - **avoid_teammates**: gain sufficiently high to keep the robots from colliding.

- *move_to_back_field (mtbf)*: The robot moves to the back third of the field while being simultaneously attracted to the ball. The robot will kick/bump the ball if it is comes within range. Active schemas include
  - **move_to_defended_goal**: high gain to draw the robot to the defended goal. A "dead zone" centered on the goal area permits the robot to roam freely if it is near the goal.
  - **move_to_kickspot**: gain sufficiently high to draw the robot to the ball if it is near the goal, but not high enough to pull the robot away from the goal.

The overall system is completed by sequencing the assemblages with a selector that activates an appropriate skill depending on the robot's situation. This is accomplished by combining a boolean perceptual feature, *behind_ball* with a selection operator. The selector picks one of the three assemblages for activation, depending on the current value of *behind_ball*.

The control team includes three "forwards" and one "goalie." The forwards and goalie are distinguished by the assemblage they activate when they find themselves behind the ball: the forwards move to the ball while the goalie remains in the backfield. Both types of player will try to get behind the ball when they find themselves in front of it.

## 7.3 Design of learning strategies

To isolate the impact of learning on performance, the learning teams were developed using the same behavioral assemblages and perceptual features as the control team. This approach ensures that **the performance of a learning team versus the control team is due only to differences in policy.**

Clay includes both fixed (non-learning) and learning coordination operators. The control team's configuration uses a fixed selector for coordination. Learning is introduced by replacing the fixed mechanism with a learning selector. A Q-learning module is embedded in the learning selector [WD92].

The Q-learner automatically tracks previous perceptions and rewards to refine its policy. At each step, the learning module is provided the current reward and perceptual state. It learns over time to select the best assemblage given the situation.

### 7.3.1 Reinforcement functions for soccer

The policy an agent learns is likely to depend on the reward function used to train it. One objective of this research is to discover how *local* versus *global* reinforcement impacts the diversity and performance of learning teams. Global reinforcement refers to the case where a single reinforcement signal is simultaneously delivered to all agents, while with local reinforcement each agent is rewarded individually. To that end, we consider two reinforcement functions for learning soccer robots. Assuming the game proceeds in discrete steps, the global reinforcement function at timestep $t$ is:

$$R_{\text{global}}(t) = \begin{cases} 1 & \text{if the team scored at } t-1, \\ -1 & \text{if the opponent scored at } t-1, \\ 0 & \text{otherwise.} \end{cases}$$

This function will reward all team members when any one of them scores. Thus a goalie will be rewarded when a forward scores, and the forward will be punished when the goalie misses a block. Observe that the global reinforcement function and the performance metric (Equation 8.1) are related:

$$P = \sum_{t=0}^{t=N} R_{\text{global}}(t)$$

where $N$ is the number of steps in the game. $R_{\text{global}}$ is a performance-based reward. A close correlation between reward function and performance metric is helpful, since reinforcement learning mechanisms seek to maximize their reward. In terms of the taxonomy presented in Chapter 4 $R_{\text{global}}$ is an `INTERNAL_SOURCE`, `PERFORMANCE`, `DELAYED`, `DISCRETE` and `GLOBAL` reward function. Now, consider a local function where each agent is rewarded individually:

$$R_{\text{local}}(t) = \begin{cases} 1 & \text{if the agent was closest to the ball} \\ & \text{when its team scored at } t-1, \\ -1 & \text{if the agent was closest to the ball} \\ & \text{when the opposing team scored at } t-1, \\ 0 & \text{otherwise.} \end{cases}$$

Even though global information is required to implement this reward function, in this context `LOCAL` refers to the fact that the reward is based on the individual's performance, not the entire teams'. This function will reward the agent that scores and punish an agent that allows an opponent to score. There may not be much benefit, in terms of reward, for a robot to serve a defensive role in this model since it would receive frequent negative but no positive rewards. In terms of the reward taxonomy, $R_{\text{local}}$ is classified the same as $R_{\text{global}}$, except its locality is `LOCAL` rather than `GLOBAL`. The $R_{\text{local}}$ reward is `INTERNAL_SOURCE`, `PERFORMANCE`, `DELAYED`, `DISCRETE` and `LOCAL`.

A potential problem with the $R_{\text{local}}$ function is the implicit assumption that the agent closest to the ball is the one responsible for a scoring event. It may be that the closest robot just happened to be near the goal while another agent kicked the ball for a score from a distance. To address this, a separate reward function, based on time since the ball was touched was investigated:

$$R_{\text{touch}}(t) = \begin{cases} \gamma^{t_{\text{touch}}} & \text{if the team score at } t-1, \\ -\gamma^{t_{\text{touch}}} & \text{if the opponent scores at } t-1, \\ 0 & \text{otherwise.} \end{cases}$$

$t_{\text{touch}}$ is time in milliseconds since the agent last touched the ball. $\gamma$ is a parameter set to values between 0 and 1 that indicates how quickly a potential reward should decay after the ball is touched. Note that $R_{\text{touch}}$ can be written in terms of $R_{\text{global}}$

$$R_{\text{touch}}(t) = R_{\text{global}}(t)\gamma^{t_{\text{touch}}}$$

If $\gamma = 1$, $R_{\text{touch}} = R_{\text{global}}$. As $\gamma$ is reduced towards 0 the reward becomes progressively more agent-centered or local. The $R_{\text{touch}}$ reward is `INTERNAL_SOURCE`, `PERFORMANCE`, `DELAYED`, `CONTINUOUS` and `COMB_LOCALITY`.

## 7.4 Performance with local and global rewards

The first set of experiments were conducted in the simplified soccer task using the $R_{\text{local}}$ and $R_{\text{global}}$ reward functions. Experimental data were gathered by simulating thousands of soccer games and monitoring robot performance. The learning robots are evaluated on three criteria: task performance (score), policy convergence, and diversity of behavior.

For each trial, the learning robots were initialized with all Q-values set to zero. A series of 100 10-point games were played. Information on policy convergence and score was recorded after each game. The robots retain their learning set between games. An experiment is composed of 10 runs, or a total of 1000 10-point games. Each run uses the same initial parameters but different pseudo-random number seeds.

### 7.4.1 Task performance

Performance is measured as the difference between the learning team's score and the opponent's score (Equation 8.1). A negative value indicates the team lost the game, while positive values indicate the team won the game. When rewarded using the global reinforcement signal $R_{\text{global}}$, the learning teams out-score the control team by an average of six points to four, yielding a performance of 2.0. The average includes the initial phase of training. When trained using the local reward $R_{\text{local}}$, the learning teams lose by an average of four points to six, or a performance of -2.0. In these soccer experiments, **teams trained using global reinforcement perform best.**

### 7.4.2 Learning rate

Learning rate is evaluated by checking for policy convergence. Convergence is tracked by monitoring how frequently an agent's policy changes. Consider a robot that may have been following a policy of moving to the ball when behind it, but due to a recent reinforcement it switches to the *get_behind_ball* assemblage instead. Switches like this are tracked as policy changes.

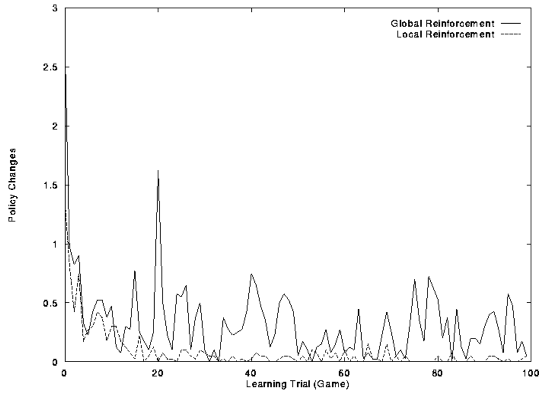



**Figure 7.2**: Policy convergence measured as average number of policy changes per trial for teams using local and global rewards.

The data, plotted in Figure 7.2, shows good convergence for robots using local rewards. The average number of changes per game drops to 0.05 after 100 games. An individual simulation to 1000 games using the local reward resulted in convergence to zero. The number of policy changes for robots using $R_{\text{global}}$ initially decreases, but does not converge in the first 100 games. The average number of policy changes is 0.25 per game after 100 games. In these experiments **teams using local rewards show better policy convergence properties than teams using global rewards.**

**Table 7.2**: The nine soccer robot policies possible for the learning agents discussed in the text. Each policy is composed of one row for each of the two possible perceptual states (not behind ball or behind ball). The position of the 1 in a row indicates which assemblage is activated for that policy in that situation. The policies of the goalie and forward robots introduced earlier (Figure 2) are in bold.

| | *mtb* | *gbb* | *mtbf* | *mtb* | *gbb* | *mtbf* | *mtb* | *gbb* | *mtbf* |
|---|---|---|---|---|---|---|---|---|---|
| *not bb* | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| *bb* | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| *not bb* | **0** | **1** | **0** | 0 | 1 | 0 | **0** | **1** | **0** |
| *bb* | **0** | **0** | **1** | 0 | 1 | 0 | **1** | **0** | **0** |
| *not bb* | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| *bb* | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

### 7.4.3 Diversity

After the training phase, robots are evaluated for behavioral diversity by examining their policies. The teams are classified as hetero- or homogeneous depending on whether the robot's policies are the same. Altogether there are 9 possible policies

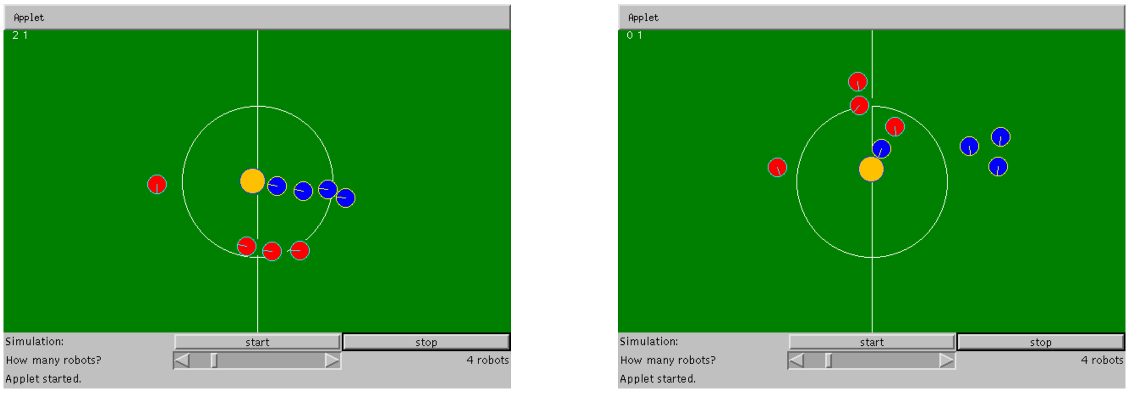



**Figure 7.3**: Examples of homo- and heterogeneous learning soccer teams. In both cases the learning team (dark) defends the goal on the right. The agents try to propel the ball across the opponent's goal by bumping it. A homogeneous team (left image) has converged to four identical behaviors which in this case causes them to group together as they move towards the ball. A heterogeneous team (right) has settled on diverse policies which spread them apart into the forward and middle of the field.

for the learning agents since for each of the two perceptual states, they may select one of three assemblages. Table 7.2 summarizes the possible policies. Based on these nine policies, there are a total of 6561 possible 4 robot teams.

Two example teams, one homogeneous, the other heterogeneous, are illustrated in Figure 7.3. All members of the team on the left have converged to identical policies. In fact, *all* robots in the 10 locally-reinforced teams converged to the same "forward" policy used by the control team (Table 7.1). All 10 teams converged to fully homogeneous behavior.

In contrast, all of the 10 globally-reinforced teams diversify to heterogeneous behavior. In all cases, the agents settle on one of three particular policies. All the teams include one robot that converges to the same "forward" policy used by the control team; they also include at least one agent that follows the same policy as the control team's "goalie." The other robots learn a policy of always selecting the *get_behind_ball* assemblage, no matter the situation (for convenience this policy is referred to as a "mid-back"). In cases where the team had not fully converged, investigation reveals that the changes are due to one agent alternating between the "goalie" and "mid-back" policies. In summary, the globally-reinforced teams always converged to one "forward," one or two "mid-backs" and one or two "goalies."

To quantify the varying degree of diversity in these teams, social entropy (presented in Chapter 5) is used as a measure of behavioral heterogeneity. Social entropy, inspired by Shannon's Information Entropy [Sha49], evaluates the diversity of a robot society based on the number of behavioral castes it includes and the relative size of each. $H(\mathcal{R})$, the social entropy of the robot society $\mathcal{R}$, ranges from a minimum of zero, when all agents are identical, to a maximum when each robot forms a different caste. The maximum entropy for a team of four soccer robots is 2.0. $H(\mathcal{R}) = 0$ for the homogeneous teams trained using local reinforcement and $H(\mathcal{R}) = 1.5$ for the heterogeneous teams.

## 7.5 Performance using $R_{\text{touch}}$

Another set of experiments were conducted in the RoboCup soccer task using the $R_{\text{touch}}$ reward function for learning. As in the previous experiments, data were gathered by simulating thousands of soccer games and monitoring task performance (score difference), policy convergence, and diversity of behavior. To investigate how the value of $\gamma$ in $R_{\text{touch}}$ impacts performance and diversity, simulations were run as $\gamma$ was swept from 0.1 and 1.0 in steps of 0.1. For each value of $\gamma$, 10 runs of 100 trials were conducted. For each run the simulator was initialized with a different random number seed. Each trial is two simulated minute game. The shorter time was used so that policy changes and performace could be evaluated with a finer resolution than a ten minute trial would permit.

At the beginning of each run the learning robots were initialized with a random policy; Q-values were set to random values between -1 and 1. Next a series of 100 two minute trials are conducted with information on policy convergence and score recorded after each trial. The robots retain their learning set between trials. Each run uses the same initial parameters but different random number seeds.

### 7.5.1 Task performance

Performance is the difference between the learning team's score and the fixed opponent's score at the end of each trial (Equation 8.1). For each value of $\gamma$ between 0.1 and 1.0, average performance is computed from the results of 10 experimental runs. These results are plotted in Figure 7.4. In all cases, **the teams trained using $R_{\text{touch}}$ out perform the pre-programmed control team**. From the graph it is apparent that **performance improves as $\gamma$ increases, to a maximum when** $\gamma = 1.0$. When $\gamma = 1.0$ the learning teams out-score the control team by an average

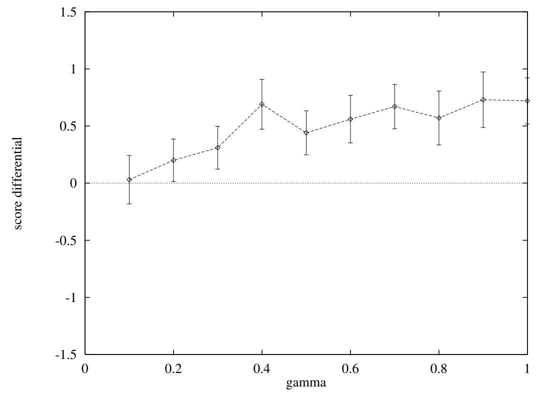



**Figure 7.4**: Score differentials for teams using the $R_{\text{touch}}$ reward function as $\gamma$ is swept from 0.1 to 1.0. Error bars show 95% confidence intervals. Positive numbers indicate the experimental team is winning on average.

of 0.72 points per two minute trial. This result supports the earlier experiments involving $R_{\text{local}}$ and $R_{\text{global}}$ that indicated performance was best with global rewards (recall that when $\gamma = 1.0$, $R_{\text{touch}} = R_{\text{global}}$).

### 7.5.2 Learning rate

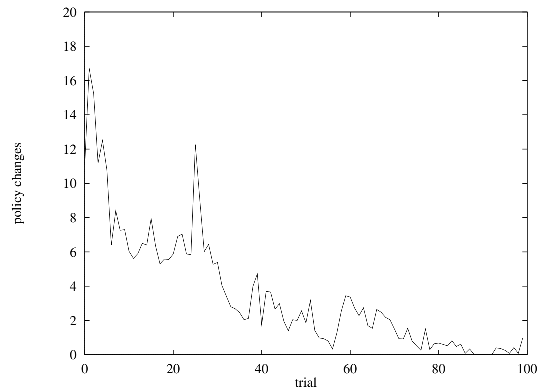



**Figure 7.5**: Policy changes versus trial number for teams using the $R_{\text{touch}}$ reward function ($\gamma = 1.0$).

As in the first set of experiments, learning rate is measured as policy convergence and is tracked by monitoring how frequently an agent's policy changes. The average number of policy changes per trial for 10 runs is plotted in Figure 7.5. This graph is for teams trained using $R_{\text{touch}}$ with $\gamma = 1$. Convergence rates are similar for other values of $\gamma$ as well.

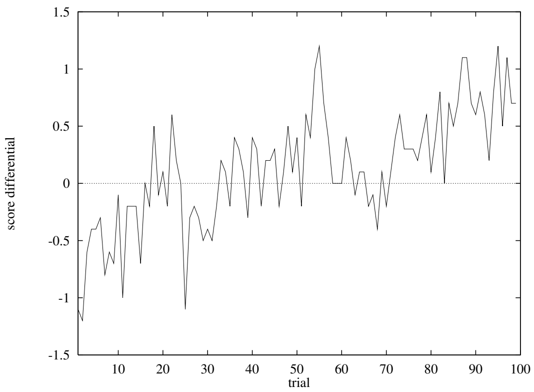



**Figure 7.6**: Score versus trial number for teams trained using $R_{\text{touch}}$ ($\gamma = 1.0$).

Learning rate can also be evaluated by monitoring performance over time. Average performance for teams trained using $R_{\text{touch}}$ with $\gamma = 1$ is plotted versus trial number in Figure 7.6. In early trials, performance is negative, but it improves throughout the run, leveling off near 0.7.

**Agents trained using $R_{\text{touch}}$ show good convergence properties.**

### 7.5.3 Diversity

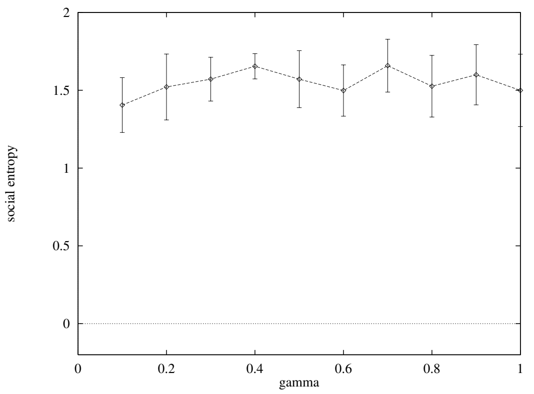



**Figure 7.7**: Hierarchic social entropy for soccer teams trained using $R_{\text{touch}}$ as $\gamma$ varies from 0.1 to 1.0. Error bars show 95% confidence intervals.

Diversity in is measured after the learning phase is complete using hierarchic social entropy (Chapter 5). For teams of five robots, entropy can range from a minimum of 0.0 (all agents are identical) to 2.32 (all agents are different). The graph in Figure 7.7 plots diversity for learning soccer teams as $\gamma$ is swept from 0.1 to 1.0. Measured diversity is approximately 1.5 for all values of $\gamma$. The data indicate that

**diversity is not impacted by $\gamma$ in robot teams trained using $R_{\text{touch}}$.**

Recall that when $\gamma = 1.0$, $R_{\text{touch}}$ returns a reward equivalent to $R_{\text{global}}$. Note that even when $\gamma$ is set to a small value $R_{\text{touch}}$ will always return some positive or negative reward to all agents whenever a scoring event occurs. In this regard, $R_{\text{touch}}$ is a global reward function, regardless of $\gamma$. This may be why diversity is apparent in the soccer teams across all values of $\gamma$. It is also interesting to note that the level of diversity in these learning teams (1.5) is the same as that found in the experiments in the simplified soccer domain with fewer robots.

## 7.6 Performance versus human-designed teams

The results reported up to this point show that a simulated robot team can learn a winning soccer strategy against a fixed control team. The learning teams are provided the same behavioral assemblages as the fixed opponent so that any difference in performance is due to the sequencing strategies the agents learn, not the behavioral assemblages themselves. This experimental approach leaves open the possibility that

1. the strategy utilized by the fixed opponent team may be poor, and finding a way to beat it is easy, or

2. the behavioral assemblages may be too simple and could never be utilized in a really successful robot soccer strategy.

Either of these possibilities would reduce the significance of the results.

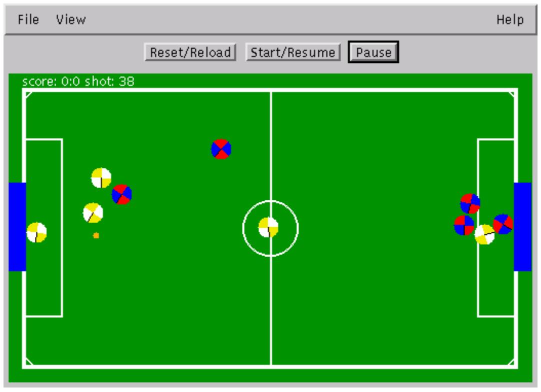



**Figure 7.8**: Example team trained using $R_{\text{touch}}$ in trials against **DTeam**. The dark colored learning agents defend the goal on the right. Three agents have converged to a defensive role while two play offensive positions.

To address this, learning soccer teams were tested in experiments against teams developed by others. Students in classes[1] taught in the College of Computing at Georgia Tech were assigned the task of developing a multiagent robot soccer team using the JavaBots system. The students were provided the same fixed opponent team used in the earlier experiments as a "straw-man" for testing their own teams. Since students' grades were linked to how well their teams performed, it can be assumed they did their best to develop effective strategies. Approximately 16 teams were developed by students in these classes. The best three were chosen for evaluation here. All three of these teams used fixed strategies, they do not learn:

- **BriSpec** designed by Brian McNamara. The members of this team play three different positions. One player always remains at the back of the field; it aligns itself between the ball and the goal. Three players play mid-field positions; they stay behind the ball and attempt to spread out from one another. The remaining agent stays in front of the ball, in expectation of a pass.
- **DTeam** by David H. Johnson. The players on this team diversify to fill four specialized roles. Two of the players exploit weaknesses in the simulator dyanmics and soccer rules as follows: One player always moves to block the opponent's goalie. The simulated dynamics are such that one player cannot push another, so the blocking is usually effective. Another player always waits near the center of the field for the ball to appear. This behavior exploits the simulation's deadlock prevention scheme; the ball is repositioned to the center of the field when 60 simulated seconds elapse with no score. This player often is the first to get the ball when it is repositioned. The remaining three players serve as a goalie and two forwards. DTeam was programmed using Clay.
- **Kechze** by Kent Lyons, Christopher Jurney and Zellyn Hunter. Kechze is similar to DTeam in that it exploits the ball re-positioning rule of the simulation. This team is rule-based however (instead of using motor schemas as DTeam does). Like the other teams, Kechze players fill specialized roles. But Kechze has an important refinement that enables it to improve its performance: the agent assigned to wait in the middle of the field for the ball does not do so until a certain time elapses. This delay is in recognition of the simulator's timed ball relocation scheme. The Kechze team gains full use of that player for more time. In contrast, the player that fills this role on DTeam always moves directly to the center position and does not contribute to play until the ball is re-located.

The $R_{\text{touch}}$ reward function with $\gamma = 1.0$ was used to train agents in learning trials against these teams. For each of the three experiments, the learning agents were initialized with a random policy (Q-values were set to random values between

[1] CS 7100 taught in Fall 1997 by Irfan Essa and CS 4324 taught in Spring 1998 by Chris Atkeson.

-1 and 1). Next, a series of 200 two minute trials were played between the learning teams and each opponent team.

Overall performance is evaluated as the average score difference in the last 10 trials of each experimental run. Plots of performance versus learning trial for each of the three opponent teams are provided in Figures 7.9 and 7.10. In each case, the learning teams converge to a winning strategy with a positive (winning) score differential. An example team trained versus **DTeam** is illustrated in Figure 7.8. In these experiments **learning soccer teams using the behaviors developed in this work out perform the best human-designed strategies**.

Table 7.3 summarizes these results, as well as the other experiments examined in this chapter. The only losing learning teams were those programmed to use the $R$_local reward. All other teams converged to winning and relatively diverse strategies.

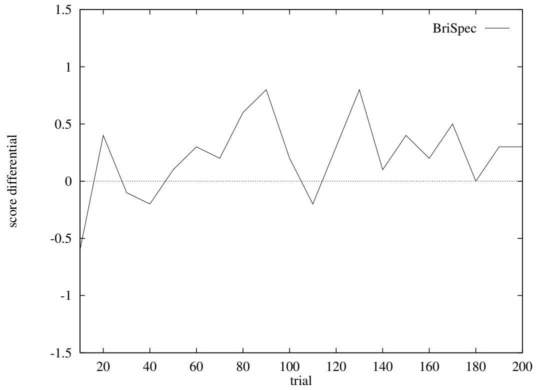



**Figure 7.9**: Performance versus trial number for games against the **BriSpec** team. Each point is the average performance of the learning team over 10 trials.

## 7.7 Discussion and summary

The relative benefits of three different reinforcement functions for robot soccer teams have been evaluated in terms of team performance, learning rate, and social entropy in the resulting team. The three reward functions, $R_{\text{local}}$, $R_{\text{global}}$ and $R_{\text{touch}}$ were evaluated on learning teams as they engaged a fixed opponent team and three other human-designed teams in thousands of trials. The primary results are

- individual learning robots will, in many cases, automatically diversify to fill different roles on a team;

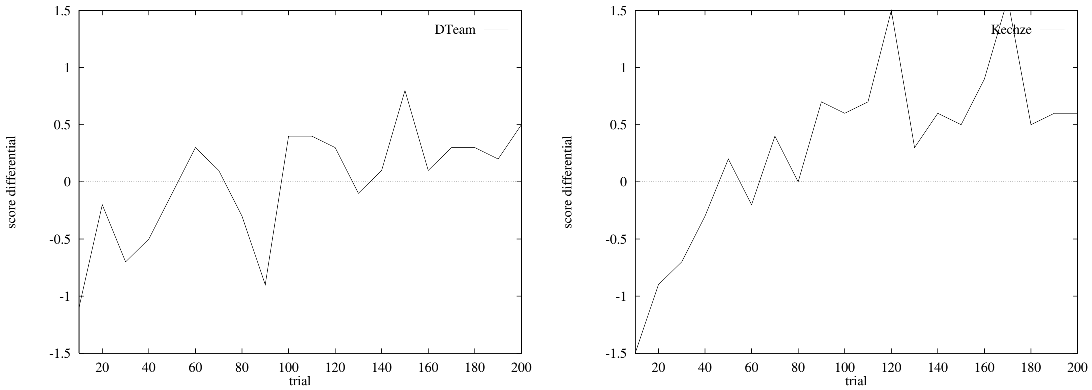



**Figure 7.10**: Performance versus trial number for games against two human-designed teams: **DTeam** (left) and **Kechze** (right). Each point on each curve represents the average performance of a learning team over 10 trials.

**Table 7.3**: Performance and diversity results from robot soccer experiments. Except in the case of agents trained using $R_{\text{local}}$ all teams converge to winning strategies.

| **reward function** | **opponent** | **domain** | **performance** | **social entropy** |
|---|---|---|---|---|
| $R_{\text{local}}$ | fixed | simplified | -2.0 | 0.0 |
| $R_{\text{global}}$ | fixed | simplified | 2.0 | 1.5 |
| $R_{\text{touch}}$, $\gamma = 0.1$ | fixed | RoboCup | 0.0 | 1.4 |
| $R_{\text{touch}}$, $\gamma = 1.0$ | fixed | RoboCup | 0.7 | 1.5 |
| $R_{\text{touch}}$, $\gamma = 1.0$ | BriSpec | RoboCup | 0.3 | 1.7 |
| $R_{\text{touch}}$, $\gamma = 1.0$ | DTeam | RoboCup | 0.5 | 1.9 |
| $R_{\text{touch}}$, $\gamma = 1.0$ | Kechze | RoboCup | 0.6 | 1.2 |

- after a training period, teams of learning robots out perform the best human-designed teams;
- global reinforcement leads to better performance and greater diversity, but slow policy convergence for robot teams;
- local reinforcement leads to poorer performance and more homogeneous behavior, but faster policy convergence.

The performance of teams using $R_{\text{local}}$ and $R_{\text{global}}$ for learning in a simplified soccer domain show that local rewards provide quicker learning, while global reinforcement leads to better performance and greater diversity. Also, the globally-reinforced teams perform significantly better than the pre-programmed control team. The locally-reinforced teams converge to "greedy" behaviors that maximize their individual reward, but lead to poor team performance. This may suggest that defensive play is important in soccer but there is no incentive for a robot to fill a defensive role. With the local reward strategy a goalie would be "punished" every time the opponent scores and never receive a positive reinforcement. Quick convergence in the locally-reinforced teams is due to the close relationship between an individual agent's actions and the rewards it receives with local reinforcement strategies.

Additional experiments were conducted in the RoboCup task using the $R_{\text{touch}}$ reward function. This function provides a reward based on time since the robot last touched the ball. If a goal is scored and the agent touched the ball recently, its reward is greater than than if it touched the ball further in the past. A parameter of the reward function, $\gamma$ sets the rate at which the reward decays. Rewards using $R_{\text{touch}}$ reward function with $\gamma = 1.0$ are identical to those generated by the $R_{\text{global}}$ function. Experiments conducted by sweeping $\gamma$ from 0.1 to 1.0, show that performance is best with $\gamma = 1.0$. Diversity is not impacted by the value of $\gamma$; all teams using $R_{\text{touch}}$ converged behavioral diversities of approximately 1.5 (the same as teams using global reinforcement). This result is probably due to the fact that, no matter what value $\gamma$ is set to, all robots receive some non-zero reward at every scoring event – hence the reward always has a global nature.

In all of these experiments a fixed opponent team was configured from the same behaviors available to the learning teams. This approach was utilized to ensure that differences in performance were due to a team's policy or learning strategy and not the behaviors from which it selects. This leaves open the possibility however, that the fixed opponent is easy to beat, thus the learning systems are not adequately challenged. Experiments against three human-developed soccer teams were conducted to address this. In all three cases **learning teams out-performed the human-developed soccer teams**.

# Chapter 8

# Diversity in Cooperative Movement

This chapter describes experiments involving teams of simulated robots learning a cooperative navigation task. The agents select from one of several formation strategies (including no formation at all) as they move across obstacle-strewn terrain. At issue is whether the agents benefit from formation behavior, and if so, whether teams perform best when all agents choose the same behavior. Teams using fixed homogeneous strategies are also evaluated for comparison.

Each robot is provided a common set of cooperative movement skills (motor schema-based behavioral assemblages) from which it learns to select using reinforcement learning. The agents learn individually to activate a particular behavioral assemblage given a reward signal. In contrast to the domains examined in earlier chapters, it is not necessary for the agents to learn a sequence of behaviors to succeed in this task. The agents learn which one of four cooperative movement behaviors to

**Figure 8.1**: A team of four robotic scout vehicles manufactured for DARPA's Demo II project. These robots were the target platform for earlier research in robot formations. Photograph courtesy of Lockheed-Martin.

activate; the same behavior is active for the entire trial.

The experiments in navigation simulations evaluate the agents in terms of *performance*, *policy convergence*, and *behavioral diversity*. As in foraging and soccer experiments (Chapters 6 and 7) the results show that robots will diversify by choosing heterogeneous behaviors. An interesting result is that teams using diverse movement behaviors perform better than homogeneous teams. In contrast to the results in other tasks, however, the degree of diversification does not depend on the reward structure. Navigating teams learn to perform equally well using local or global rewards.

The chapter proceeds with a discussion of the task, behaviors for accomplishing it and a description of the experimental results. Experiments follow the methodology introduced in Chapter 3.

## 8.1 Background and related work

The development of this task domain and the behaviors designed for it are extensions of previous research conducted in the Mobile Robot Laboratory at Georgia Tech [BA95b, BA99]. The earlier work was focused on developing behaviors for a team of robotic vehicles to be fielded as a scout unit by the U.S. Army (Figure 8.1). Formation is important in this and other military applications where sensor assets are limited. Formations allow individual team members to concentrate their sensors across a portion of the environment, while their partners cover the rest. Air Force fighter pilots for instance, direct their visual and radar search responsibilities depending on their position in a formation [For92]. Robotic scouts also benefit by directing their sensors in different areas to ensure full coverage [CGH96]. The approach is potentially applicable in many other domains such as search and rescue, agricultural coverage tasks and security patrols.

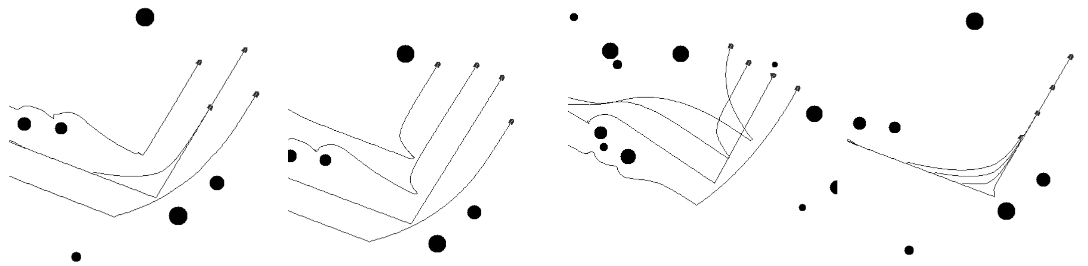


**Figure 8.2**: Four robots in leader-referenced *diamond, wedge, line* and *column* formations. These formation behaviors, developed in earlier research, are targeted for heterogeneous teams where each robot is assigned a specific position in formation.

Several formation strategies for scout robots were developed to enable the team to move cooperatively in military scenarios. In the scout domain the multi-robot team is heterogeneous because each agent is assigned a position in the formation according to an identification number. This is important in applications where one or more of the agents are dissimilar. In Army scout platoons for instance, the leader is not usually at the front of the formation, but in the middle, or to one side.

Important contributions of this earlier work include behaviors for four-robot *diamond*, *line*, *column*, and *wedge* formation types and a performance analysis of each formation type in turns and across obstacle-strewn terrain. Results from the earlier work are compared with the performance of the new behaviors presented in this chapter. The four formation types developed previously are illustrated in Figure 8.2.

The earlier strategy works well, but it is limited to formations with a specific number of robots. The location of each robot in each formation is predefined and formations are not easily scalable to larger numbers of agents. The expectation is that in large-scale homogeneous teams agents should automatically move to the closest appropriate location. To provide this capability, a new, scalable formation technique is introduced here. An example large-scale robot formation using the new technique is given in Figure 8.3.

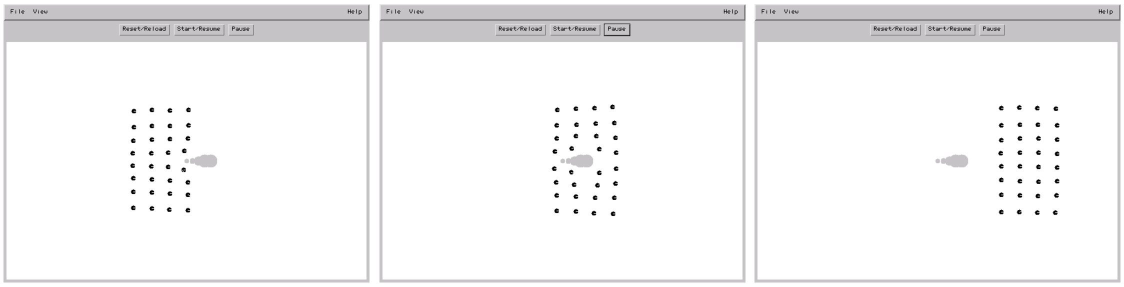


**Figure 8.3**: Large scale formation: 32 robots (black circles) moving from left to right in formation encounter an obstacle (grey object). These robots utilize the new scalable strategies introduced here. Sequence is from left to right.

The next section describes the task and experimental environment in detail. Following that, the new formation behaviors are introduced.

## 8.2 Task and performance metric

The task examined in these experiments is for a team of robots to move across a field as quickly as possible while avoiding collisions with obstacles and other robots. Performance is defined as

$$P \quad = \quad -t \tag{8.1}$$

where $t$ is the time in milliseconds for the *entire* team of robots to move across the field. This is equivalent to the performance of the last agent to cross the field. Several other performance measures were considered, including the average time for all agents to complete the task and the time of the first robot to move across. The time for the last agent to complete the task was chosen because it indicates, to some degree, the extent of cooperation between the robots. Other measures might show improved performance when individual agents "abandon" their partners in an effort to cross the finish line more rapidly. Note however, that even though it may be advantageous for the robots to move in a group, this is not explicitly part of the performance measure.

In terms of the taxonomy introduced in Chapter 4 this task and performance metric have the following characteristics:

- `TIME_MIN` because the task must be completed in minimum time,

- ROBOT_BASED since performance is based on the location of the robots,
- MOVEMENT_TO because the robots must move to a location,
- MULTI_AGENT because the task implicitly requires all robots to complete the task,
- SENSOR_LIM since agents only have a short-range view of the environment (e.g. obstacles).

Figure 8.4 illustrates the JavaBots simulation environment used in the experiments. The field measures 20m by 60m. 30 obstacles, each 1m$^2$ in area, are distributed randomly about a 20 by 30 meter zone in the middle of the field (5% obstacle coverage). The robots are initialized on the left side of the field. They then navigate to the right side, through the obstacles to the finish line on the right. Timing stops when the last robot crosses the line.

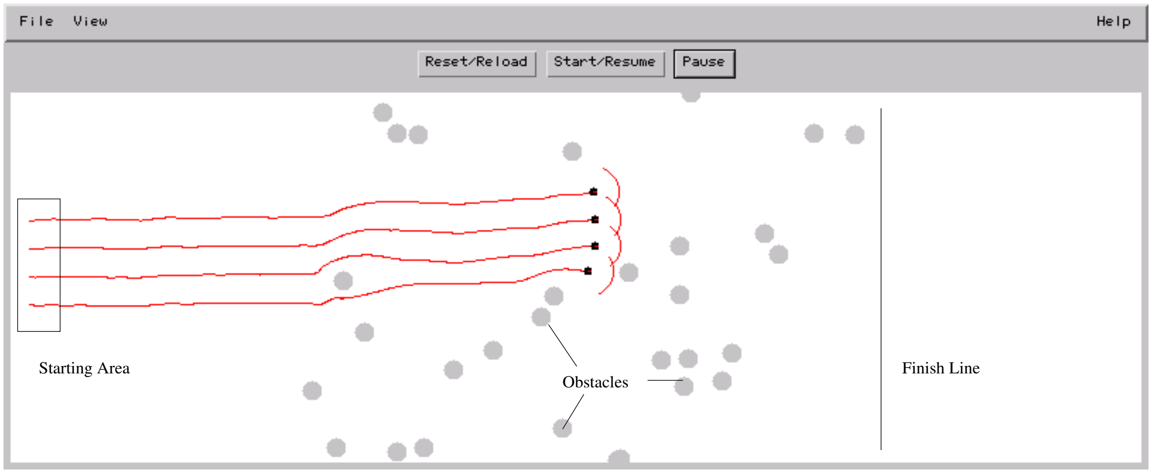



**Figure 8.4**: The simulation environment used in the experiments. Robots are initialized on the left. They navigate from left to right through the obstacles.

An example experimental run is illustrated in Figure 8.5. The agents are initialized line abreast on the left side of the field. This initial configuration was chosen because it ensures all robots are equidistant from the finish line. The first 20m of the field are clear of obstacles to enable the robots to settle into formation positions before encountering the obstacle field. After crossing the obstacle-free section the robots encounter a 30m long zone cluttered with hazards. As the figure shows, interaction with obstacles sometimes results in a rearrangement of the formation (the reason for this will become apparent as the behaviors are described in the next section).


**Figure 8.5**: Sequence of images from an experimental run with four robots programmed to use the *diamond* behavior. The top image illustrates how agents are initialized line abreast on the left side of the field. The agents settle into formation as they cross the obstacle-free area (second image). The robots regroup in a different arrangement after encountering an obstacle (third image). Finally (bottom) the team crosses the finish line.

Two aspects of the experimental setup should be considered when reviewing performance results. First, the arrangement of agents at the beginning of each run may bias the shape of the formation towards line abreast. Second, the measured time to complete the task includes the time taken for the agents to cross the initial, obstacle-free area. Thus overall performance is a combination of performance in obstacle-free and cluttered terrain.

Now consider how behaviors can be designed for this task.

## 8.3 Behavioral design

Formation maintenance is accomplished in two steps: first, a perceptual process, **detect_formation-position**, determines the robot's proper position in formation based on current sensor data; second, the motor process **maintain_formation**, generates motor commands to direct the robot toward the correct location. The motor schema paradigm enables the formation behavior to be simultaneously active in combination with other navigation behaviors.

The overall navigational strategy is similar to the approach developed in earlier research [BA95b]: several motor schemas, **move_to_goal**, **avoid_static_obstacles**, **avoid_robots** and **maintain_formation** implement the overall behavior for a robot to move to a goal location while avoiding obstacles, collisions with other robots and remaining in formation (a mathematical description of these motor schemas and the gains used in these experiments are provided in Appendix A). An additional background schema, **noise**, serves as a form of reactive "grease", dealing with some of the problems endemic to purely reactive navigational methods such as local maxima, minima and cyclic behavior [Ark89]. The key extension that distinguishes the new formation behaviors from previous work is the perceptual technique used to determine the proper formation position for each robot.

Instead of having each agent assigned to a particular position as in the previous approach, it may be advantageous to develop a more general technique. Design goals for the new formation strategy include

- **scalability:** the approach should easily scale to any number of agents,
- **locality:** the behaviors should depend only on the local sensors of each agent,
- **flexibility:** the behaviors should be flexible so as to support many formation shapes.

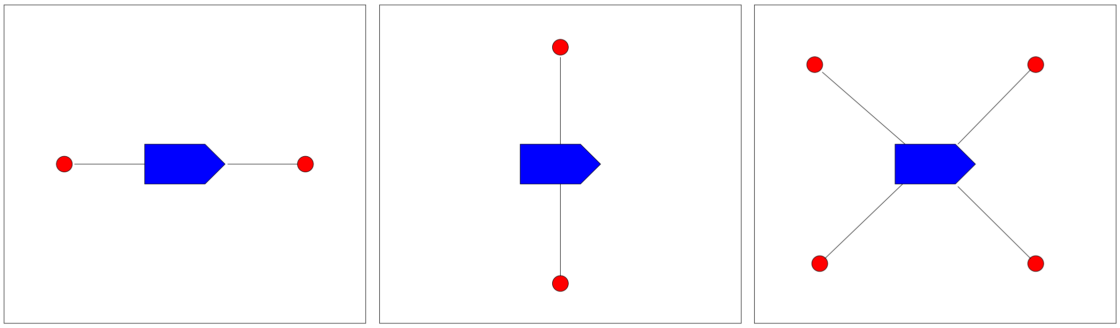


**Figure 8.6**: Attachment site geometries for different formations. From left to right: *column*, *line* and *diamond*. Robots are represented as five-sided polygons while attachment sites are shown with small circles.

This new strategy is based loosely on the way molecules form crystals. From the point of view of each robot in the group, every other robot has several local "attachment sites" other robots may be attracted to. Different formation shapes are created when different attachment site geometries are employed. Figure 8.6 illustrates the three attachment site geometries examined in this work. To determine a formation position each robot builds a list of potential attachment sites for all of the robots within sensor range based on the formation type it is using. An attractive vector is generated towards the closest site.

In addition to the motor schemas mentioned earlier, a low-gain attractive force, **move_to_unit_center**, is added to draw all of the robots together. As the team converges, the robots "snap" into position, and a regular geometric structure emerges. Example formations resulting from the integration of these behaviors are illustrated in Figure 8.7.

Note that for each attachment site geometry there are many potential robot team arrangements. It is also possible for interaction with obstacles to "unsnap" the formation into smaller sub-formations. In many cases however, the formations re-group after splitting around obstacles.

Performance of these behaviors are now examined in homogeneous teams of navigating robots.

## 8.4 Fixed formation strategies

As a baseline for comparison with learning teams, four fixed strategies were developed and evaluated. In each case, all the robots utilize the same attachment geom-

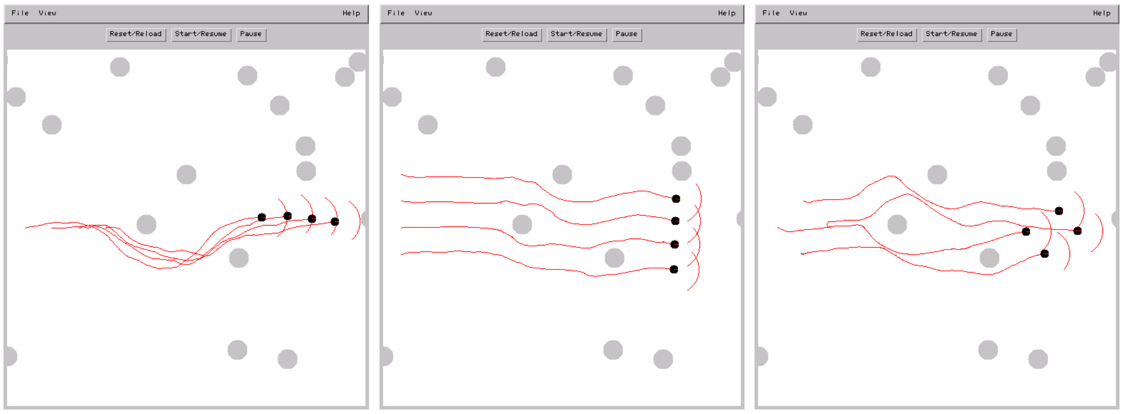


**Figure 8.7**: Example four-robot formations resulting from the use of different attachment site geometries. From left to right: *column*, *line* and *diamond*. In each of these short demonstration runs the robots were initialized in proper formation positions, experimental runs are over a longer course.

etry. Experiments were run with one to eight robots using *diamond*, *line*, *column* and *no_formation* geometries. The *no_formation* assemblage utilizes the same navigational behaviors and gains as in the other assemblages, except **maintain_formation** is not activated. The group of robots are still attracted to one another because the **move_to_unit_center** motor schema is activated.

Performance was evaluated by running each simulated robot team through five different randomly generated worlds 50 times. A total of 250 simulations were run for each number of robots for each formation geometry, or a total of 8000 trials overall. The average time for robots to complete the traverse is plotted for each strategy in Figure 8.8.

The relative performance of teams using *diamond*, *line* and *column* geometries mirrors similar results reported earlier [BA95b]. The earlier experiments in navigation across an obstacle field showed that *column* formation provides the best performance (in terms of path length). Even though performance is measured in these experiments using time instead of path length, the results are similar. The *column* geometry provides the best performance for navigation across the obstacle field. This is because the formation as a whole presents a smaller cross section to the obstacles as it moves across the field. For similar reasons, the *line* formation performs worst; it presents the broadest cross section.

In contrast to the performance of other strategies, the performance of teams using *no_formation* improves consistently as the number of robots increases. This

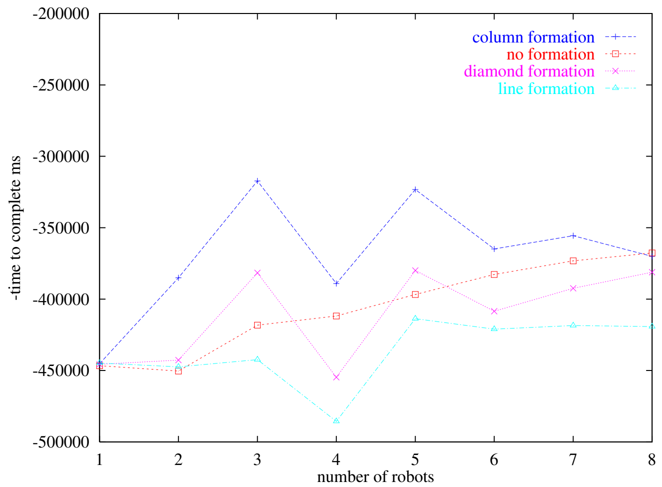



**Figure 8.8**: Average performance for fixed homogeneous teams.

is probably because in all strategies except *no_formation*, when a robot gets stuck, other robots are likely to remain near it and get stuck also. In the *no_formation* strategy, the low-gain **move_to_unit_center** behavior slows progress of the other agents, but it will not stop them. In addition the **move_to_unit_center** behavior provides the side-effect of pulling "stragglers" out of the areas they may be stuck in.

## 8.5 Learning cooperative movement strategies

Learning teams were developed using the same behavioral assemblages used in the fixed systems. This ensures that the performance of learning teams in comparison to the fixed teams is due only to differences in policy.

In contrast to the experiments in soccer and foraging, this task does not require a sequence of behaviors. Each agent selects a single behavior to follow for an entire trial, at which point it receives a reward. For the purposes of incorporating Q-learning, however, the problem can be viewed as a sequential task with one step. The Q-learner automatically tracks previous rewards to refine its choice of action for each trial.

### 8.5.1 Reinforcement functions for cooperative movement

The policy an agent learns will depend on the reward function used to train it. One objective of this research is to discover how *local* versus *global* reinforcement impacts the diversity and performance of learning teams. Global reinforcement refers to the case where the reinforcement signal is simultaneously delivered to all agents, while with local reinforcement each agent is rewarded individually. To that end, we consider two reinforcement functions for the learning robots. The local reinforcement function is:

$$R_{\text{local}} = -t \tag{8.2}$$

where $t$ is the elapsed time in milliseconds from the start of the trial until the robot crosses the finish line. This effectively rewards minimum-time completion because shorter times result in a less negative reward. In terms of the reward taxonomy $R_{\text{local}}$ is classified as an `INTERNAL_SOURCE`, `PERFORMANCE`, `IMMEDIATE`, `CONTINUOUS` and `LOCAL` reward.

The global reinforcement function is:

$$R_{\text{global}} = -t_{\text{team}} \tag{8.3}$$

where $t_{\text{team}}$ is the time when the last agent on the team crosses the line. In terms of the taxonomy presented in Chapter 4 $R_{\text{global}}$ is an `INTERNAL_SOURCE`, `PERFORMANCE`, `IMMEDIATE`, `CONTINUOUS` and `GLOBAL` reward function.

Experimental data were gathered by running thousands of trials and monitoring robot performance. The learning robots are evaluated on three criteria: task performance ($-t$), policy convergence, and diversity of behavior. For each type of reward and each number of robots (1 to 8), experiments were conducted in 5 randomly generated environments. In each environment the learning robots were initialized with Q-values set to random values between $-1$ and 1. The agents were then trained over 300 trials. Information on policy convergence and performance was recorded after each trial. The robots retain their learning set between trials. Overall, a total of 24,000 trials were run for the learning systems.

### 8.5.2 Task performance

Performance was evaluated for each number of robots for each reward type by averaging the results of the final 50 trials in each of the five experimental environments. Each data point therefore represents average performance in 250 trials. Performance for locally and globally rewarded teams is plotted in Figure 8.9.

The difference in performance between teams trained with local versus global rewards is not statistically significant. But both types of team out-perform the best homogeneous fixed strategy. This is interesting because it means the agents have discovered a better strategy than the homogeneous *column* formation for navigation across cluttered terrain. The *column* formation was shown in previous work ([BA95b]) and again in this research (Figure 8.8) to provide the most efficient (homogeneous) team navigation across cluttered terrain. Reasons for this result are examined in Section 8.6.

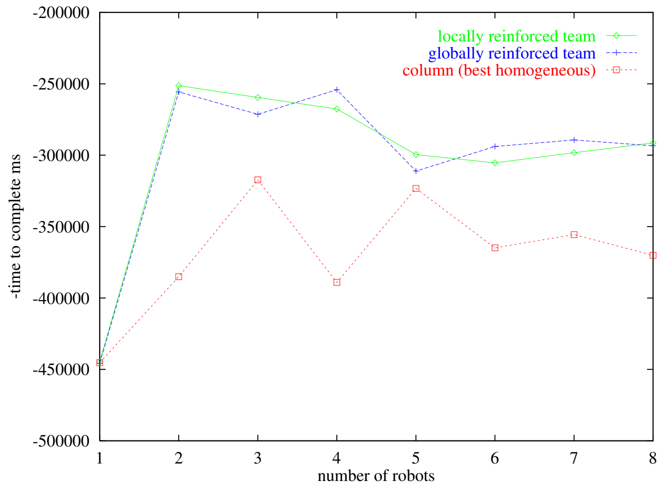



**Figure 8.9**: Performance for learning navigating teams.

### 8.5.3 Learning rate

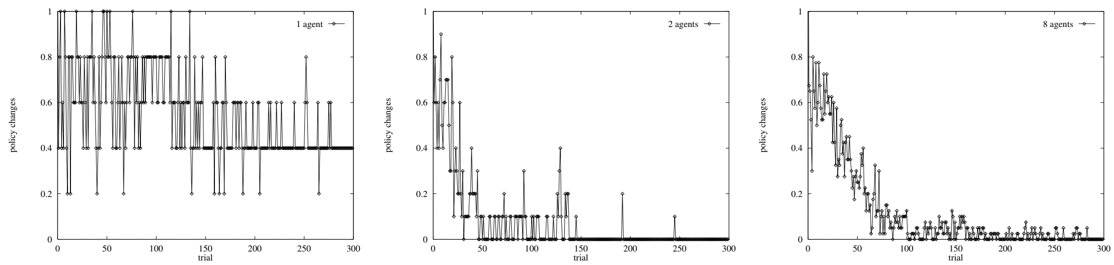



**Figure 8.10**: Policy convergence measured as average number of policy changes per trial for teams using local and rewards. Left to right: one agent, two agents, eight agents. Plots for teams using global rewards are similar.

Learning rate is evaluated by checking for policy convergence. Convergence is tracked by monitoring how frequently an agent's policy changes. At the end of a trial, after receiving its reward, an agent may switch from one behavior, say *diamond*, to another, perhaps *column*. Switches like this are tracked as policy changes. Because each trial is only a single step, a robot can only switch policies zero or one times per trial.

The data, depicted in Figure 8.10 shows convergence properties for one, two and eight robots using local rewards (plots for global rewards are similar). For two and eight robots, convergence is good, with policy changes dropping off to zero in both cases. In the case of one agent, however, convergence is poor. This is because, in the absence of any other robots to move in formation with, all navigational strategies are equally beneficial. None of the four strategies provides any advantage over the others, so the agents oscillate from one strategy to the other.

### 8.5.4 Diversity

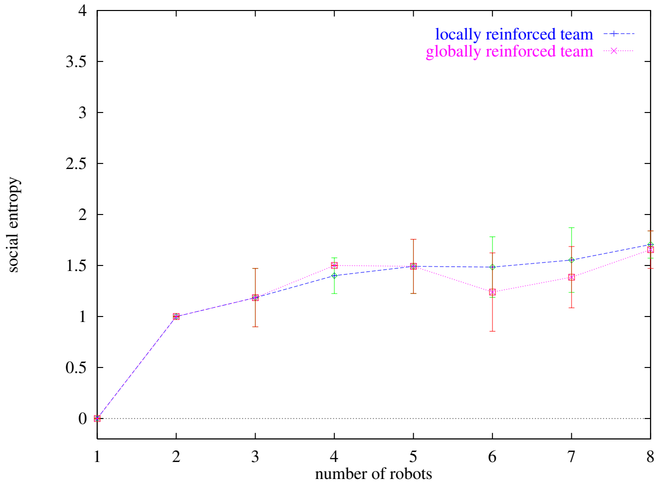



**Figure 8.11**: Hierarchic social entropy of learning navigational teams. Error bars indicate 95% confidence intervals.

After the training phase, robots are evaluated for behavioral diversity by examining their policies. Diversity is measured after the learning phase is complete using hierarchic social entropy (Chapter 5). For teams of five robots, entropy can range from a minimum of 0.0 (all agents are identical) to 2.32 (all agents are different). The graph in Figure 8.11 plots diversity for the learning navigational teams as the number of robots varies from one to eight. The data indicate no significant difference in diversity between the teams trained using local rewards and those trained using global rewards.

## 8.6 Discussion

The results of experiments in this task raise several challenging questions. First, why doesn't the selection of local versus global rewards impact diversity as it does in other tasks?

In contrast to the other tasks examined in this dissertation, this one is not internally competitive (`COMP_INT`). In soccer, for instance, a greedy forward might deny other agents opportunities to score. In this task however, a "selfish" agent seeking to maximize its own reward will not penalize other other agents on the team. It is likely that agents striving to maximize a local reward in this task would behave in the same way as agents striving to maximize a global reward. This is why we see little or no difference in performance and diversity between teams using local and global rewards.

It is also surprising that teams using a diverse set of formation behaviors can perform better than those using the best homogeneous strategy. How can this be? It would seem that formation can only work if the agents agree on the same formation geometry. The answer is that the agents learn to exploit each other's behavior to speed themselves across the terrain. Figure 8.12 illustrates.

In this example the agent at the bottom of the figure attempts to maintain a *line* formation with respect to the other robot. At the same time the robot at the top tries to maintain a *diamond* formation with respect to the bottom robot. The top robot can never reach its formation position because as it attempts to move there it pulls the other agent along with it. The resulting interaction is very much like a "carrot on a stick" for both robots. The **maintain_formation** behavior contributes a forward vector to the motion of both agents, thus speeding them up.

Note that the agents only benefit from this effect when they select differing formation behaviors. Otherwise they would quickly settle into equilibrium and be driven forward only by other forces (e.g. **move_to_goal**). **The agents diversify in response to one another's behavior.**

Another strategy observed in the learning teams is for a "leader" robot to select the *no_formation* assemblage while a follower utilizes the *column* or *diamond* behavior. The leading robot moves more quickly than if it used a formation behavior because it is repelled slightly from the trailing agent (due to the **avoid-robot** motor schema) and not pulled back by a formation force. The trailing robot in turn is "pulled" forward by its **maintain_formation** motor schema.

The resulting team behaviors do not provide the regular geometric arrangements one would expect in robot "formations." In most systems, the group breaks into pairs of agents that move across the field in an irregular formation like that in Figure 8.12. Note that regular geometry is not a performance criteria, nor is it part of either of the reward functions. There is really no reason to expect it in the resulting multi-robot teams.

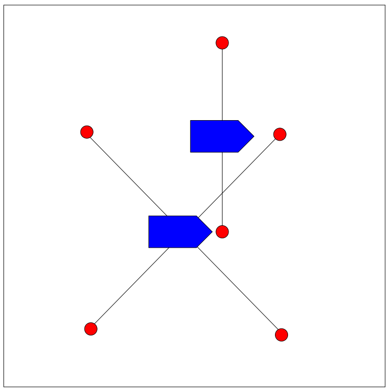


**Figure 8.12**: Agents exploit each other's formation behavior to move more quickly.

Finally, it must be pointed out that the experimental approach used may have contributed to the extent of diversity in the systems. Although each team was trained in a different random environment, the environments were not re-randomized between trials. It is likely that the robots adapted to the specific environment they were trained in. This bias could be removed by rearranging the obstacles at the start of each trial.

## 8.7 Summary

Both fixed and learning teams were evaluated in their ability to navigate quickly across an obstacle field. The experiments utilize a new scalable and flexible strategy for multi-robot formation and cooperative movement. Behaviors for four types of formation geometries were developed and evaluated: *diamond*, *line*, *column* and *no_formation*.

The results for teams following fixed homogeneous policies agreed with results from earlier work [BA95b]. In particular, *column* formations are best for navigation across cluttered terrain. Line formations perform worst.

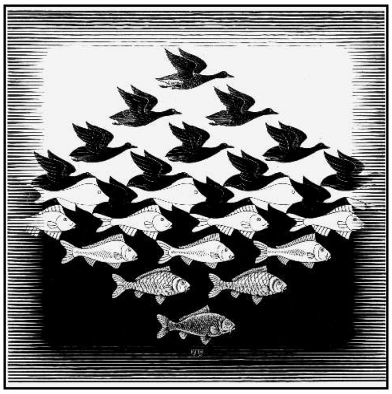
In separate experiments, robots were trained to navigate using local and global reward strategies. In contrast to the other task domains examined in this work, the performance and diversity of teams was not significantly impacted by the choice of reward. The key results for this task are:

- team performance is about the same for local and global rewards,
- both types of rewards lead to diversification in the robot teams,
- the learning robots find ways to exploit each others' behavior in order to move more quickly across the terrain.

Different types of rewards do not impact diversity or performance because the task is not internally competitive (`COMP_INT`). Agents striving to maximize individual performance do not penalize each other, and will also tend to maximize team performance.

# Chapter 9

# Conclusion



This work is based on the idea that behavioral diversity should be evaluated as a *result* rather than an initial condition of multi-robot experiments. Previously, researchers configured robot teams as homogeneous or heterogeneous *a priori*, then compared performance of the resulting teams [FM97, GM97, Par94]. That approach does not support the study of behavioral diversity as an emergent property in multi-robot teams.

Defining behavioral diversity as an independent rather than dependent variable enables the examination of heterogeneity from an ecological point of view. How and when does diversity arise in robot teams interacting with each other and their environment? This work provides the methodology and quantitative measures necessary for this new type of investigation.

The specific contributions of this work are

- **a methodology** for building and evaluating learning behavior-based robot teams (Chapter 3),
- **a taxonomy of multi-robot reinforcement functions** (Chapter 4),
- **a classification of multi-robot tasks** (Chapter 4),
- **a quantitative measure of behavioral diversity** in multi-robot teams (Chapter 5),
- **a quantitative measure of behavioral difference** between individual robots (Chapter 5),
- **evaluation procedures** for multi-robot task performance (Chapter 3), and
- **a significant body of experimental results** illustrating the use of these tools in three multi-robot task domains (Chapters 6, 7 and 8).

This chapter reviews the key contributions of the research, provides an analysis of the experimental data and concludes with a discussion of promising directions for future work.

## 9.1 Methodology

Principled research in any field requires adherence to a methodological framework. Over the last decade the Mobile Robot Laboratory at Georgia Tech has evolved and refined a successful approach to behavior-based robot design and implementation. Key components of the method are the use of simulation for experimentation and behavioral prototyping along with verification of the results on mobile robots.

The framework was extended significantly in this research. First, a formal view of multi-robot task is adopted; multi-robot tasks are classified according to how performance is measured in them. This enables a principled description and exploration of the multi-robot task space. Second, a classification of reward functions is employed. This classification defines an experimental space for investigating the impact of reward on multi-robot systems. To support these experiments, motor schema control and reinforcement learning are integrated using a new object-oriented system for behavioral specification. Finally, new evaluation metrics necessary for the measurement of diversity in multi-robot teams are developed and employed in the analysis of experimental data.

The design and implementation of multi-robot systems and their experimental evaluation is carried out in the following steps:

1. **Task and performance metric specification:** This step defines performance, one of the dependent variables of experimentation.

2. **Behavioral design:** In this phase, a library of behaviors are developed for solving the task. Both hand-coded and learning systems are built using the behavioral components.

3. **Reinforcement function specification:** A goal of the research is to explore how different reinforcement functions impact performance and diversity in learning systems. In each task domain, several reinforcement functions are employed, with primary focus on the comparison of local and global rewards.

4. **Simulation:** The behaviors and learning systems are prototyped and tested in simulation.

5. **Implementation on mobile robots:** Performance of the simulated system is validated on mobile robots. If inconsistencies are discovered the simulation environment is refined to more closely approximate mobile robot performance.

6. **Data collection:** Multiple runs (thousands, usually) are conducted in simulation, and when possible, on mobile robots. The experimental space is explored by varying the independent variables (e.g. number of robots and/or the reward function).

7. **Analysis:** The data are analyzed using the performance metric the measures of diversity presented in Chapter 5.

Each step in the methodology is covered in detail in Chapter 3. The remaining contributions of the research each play an important role in the application of this methodology: quantitative measures of diversity and behavioral difference are used to evaluate the multiagent systems developed using the methodology; classifications of task and reward establish a frame of reference for analyzing the experimental results.

## 9.2 Classifications of task and reward

Task characterization helps answer questions regarding how the same type of reinforcement can lead to different performance and and diversity levels in different tasks. Without answering "how are soccer and foraging different?" for instance, we can't answer "why is diversity good in soccer but bad in foraging?"

A taxonomy of reward structure is also important. Experimental results (Chapters 6 and 7) show that performance and diversity in a learning team depend on the form of reinforcement used to train the robots. The results indicate that there are tradeoffs to consider in the selection of a reward function; for instance some functions provide quicker learning but slightly poorer performance. Without a characterization of the differences between reward functions, it would be impossible for a robot systems designer to consider these tradeoffs intelligently.

Prior to this research no taxonomies of task or reinforcement existed. To address this crucial gap a new system for characterizing multi-robot tasks and a taxonomy of reinforcement functions for multi-robot teams are introduced in Chapter 4. As well as aiding this investigation, these classifications are potentially useful for other researchers investigating multiagent robotic systems.

One focus of the experiments in this work is to determine whether it is better for each agent to be rewarded individually or if all agents should receive the same reinforcement. `GLOBAL` reinforcement refers to the case where a single reinforcement signal is simultaneously delivered to all agents, while `LOCAL` reinforcement rewards each agent individually. In the case of soccer, a global system would reward all team members when any one of them scores a goal. With local reinforcement only the agent that scored the goal would be rewarded. Global rewards correspond more closely with overall system performance, but they may not be appropriate for all tasks. The relative advantages of these two types of reward examined in each of the three experimental task domains.

Classifications of task and reward provide a description of the independent variables in this work. We now consider the evaluation of the key dependent variable: diversity.

## 9.3 New quantitative metrics

It is impossible to correlate heterogeneity with performance in multiagent robotic systems without a quantitative metric of diversity. Previously, diversity in multi-robot teams was evaluated on a bipolar scale with systems classified as either *heterogeneous* or *homogeneous*, depending on whether any of the agents differ [FM97, GM97, Par94]. Unfortunately, this labeling doesn't tell us much about the *extent* of diversity in heterogeneous teams. Heterogeneity is better viewed on a sliding scale providing for quantitative comparisons. Such a metric enables the investigation of issues like the impact of diversity on performance, and conversely, the impact of other task factors on diversity.

*Social entropy*, inspired by Shannon's information entropy [Sha49], is introduced as a measure of diversity in robot teams (Chapter 5). It captures important components of the meaning of diversity, including the number and size of groups in a society. In order to evaluate the diversity of a team, however, a way to categorize or differentiate the behavior of individuals is also required. To address this, a measure of *behavioral difference* that provides for agent categorization is also developed. Difference refers to disparity between two specific agents, while diversity is a measure of the entire society.

Diversity may not always be desirable. In fact, experimental results presented in Chapter 6 show that for at least one multi-robot task (multi-foraging) homogeneous robot teams perform better than diverse teams. The aim of this work is to discover when diversity is important and which conditions give rise to it in learning teams. Social entropy provides the objective quantitative metric required for a principled investigation of these issues.

This research is focused specifically on diversity in teams of mechanically similar agents that use reinforcement learning to develop behavioral policies. Evaluation of diversity in teams of mechanically similar robots is challenging because when agents differ, they differ only in their behavior. Behavior is an especially interesting dimension of diversity in learning systems since as they learn, agents effectively choose between a hetero- or homogeneous society. The metrics developed in this work will help researchers investigate the origin and benefits of diversity in these learning systems.

The next section describes how the new metrics are applied in the evaluation of learning multi-robot teams.

## 9.4 Experiments

The relationships between task, reward, performance and diversity are explored through experiments in three task domains: multi-foraging, soccer and cooperative movement. The motivation for experiments in a variety of domains is to investigate how differences in task impact the utility of diversity, and to see whether the usefulness of a particular reward structure depends on the task. Results are reviewed at a high level here. Details for each domain are reported in Chapters 6, 7 and 8.

Following the methodology introduced in Chapter 3, non-learning, or fixed, team strategies are developed for each task. A schema-based reactive control system is used for robot programming in each domain. In this approach, the agent is provided several pre-programmed behavioral assemblages that correspond to steps in achieving the task. Binary perceptual features (also referred to as perceptual triggers) are used to sequence the robot through steps in achieving the task. Selection of the appropriate behavior, given the situation, may be hand-coded or discovered by the robot through reinforcement learning.

While the focus of the dissertation is diversity in *learning* robot systems, human-coded non-learning teams play an important role as well. The performance of human-designed teams establishes a baseline for comparison with results from learning systems. Also, human-coded strategies provide additional data points regarding the relationship between diversity and performance in each task. Learning teams are provided the same behavioral assemblages from which to build a task-achieving strategy, but they must discover appropriate sequences of behavior through interaction with the environment.

The space of tasks and reward strategies explored experimentally is summarized in Table 9.1. Local and global reward strategies were evaluated in each task. Additional learning strategies were tested in the multi-foraging and RoboCup soccer tasks. In all three tasks, at least one of the learning strategies led to teams that perform as well as, and in some cases better than the best human-coded strategies.

The extent to which diversity is beneficial depends on the task. Table 9.2 summarizes these results. A link between diversity and performance is found in all three tasks. In the soccer and cooperative movement experiments the best performing systems are also the most diverse. In multi-foraging, however, diversity is strongly negatively correlated with performance: the best teams are homogeneous.

The utility of global versus local rewards also depends on the task. In soccer experiments global rewards are found to work best. Just the opposite is true in foraging; local rewards lead to the best-performing multi-robot teams. Finally, in the cooperative movement task local and global rewards generate systems that perform equally well. These results are also summarized in Table 9.2.

**Table 9.1**: Summary of the task and reward space explored in robot team experiments. Experiments for all three tasks were run in simulation. Foraging experiments were also conducted on mobile robots. The reward strategies are classified according to the taxonomy introduced in Chapter 4. In addition to the features listed, all rewards are also `INTERNAL_SOURCE`.

| Task | Fixed Strategies | Learning Strategies |
|---|---|---|
| **multi-foraging** | homogeneous | $R_{\text{local}}$ |
| | territorial | `LOCAL PERFORMANCE DELAYED DISCRETE` |
| | specialize-by-color | $R_{\text{global}}$ |
| | | `GLOBAL PERFORMANCE DELAYED DISCRETE` |
| | | $R_{\text{shaped}}$ |
| | | `LOCAL HEURISTIC IMMEDIATE CONTINUOUS` |
| **soccer** | | |
| **simplified** | control team | $R_{\text{local}}$ |
| | | `LOCAL PERFORMANCE DELAYED DISCRETE` |
| | | $R_{\text{global}}$ |
| | | `GLOBAL PERFORMANCE DELAYED DISCRETE` |
| **RoboCup** | control team | $R_{\text{touch}}$ |
| | BriSpec by McNamara | `COMB_LOCALITY PERFORMANCE DELAYED CONTINUOUS` |
| | DTeam by Johnson | $R_{\text{global}}$ ($R_{\text{touch}}$ with $\gamma = 1$) |
| | Kechze by Lyons *et al* | `GLOBAL PERFORMANCE DELAYED CONTINUOUS` |
| **cooperative movement** | homogeneous diamond | $R_{\text{local}}$ |
| | homogeneous line | `LOCAL PERFORMANCE IMMEDIATE CONTINUOUS` |
| | homogeneous column | $R_{\text{global}}$ |
| | | `GLOBAL PERFORMANCE IMMEDIATE CONTINUOUS` |

**Table 9.2**: Description of each task according to the classification introduced in Chapter 4 and a summary of the key results in each task.

| | **Multi-foraging** | **Simplified Soccer** | **RoboCup Soccer** | **Coop. Movement** |
|---|---|---|---|---|
| **Classification** | `TIME_LIM` | `TIME_UNLIM` | `TIME_LIM` | `TIME_MIN` |
| | `RESOURCE_LIM` | | | |
| | `OBJECT_BASED` | `OBJECT_BASED` | `OBJECT_BASED` | `ROBOT_BASED` |
| | | `COMP_EXT` | `COMP_EXT` | |
| | `COMP_INT` | `COMP_INT` | `COMP_INT` | |
| | `SINGLE_AGENT` | `MULTI_AGENT` | `MULTI_AGENT` | `MULTI_AGENT` |
| | `SENSOR_LIM` | `SENSOR_COMPLETE` | `SENSOR_COMPLETE` | `SENSOR_LIM` |
| **diversity/perf. correlated?** | yes, negatively | yes, positively | yes, positively | yes, positively |
| **best reward(s)** | $R_{\text{local}}$ | $R_{\text{global}}$ | $R_{\text{global}}$ | $R_{\text{global}}$ |
| | $R_{\text{shaped}}$ | | | $R_{\text{local}}$ |

## 9.5 Discussion of results

The long-range goal for the research begun here is a general model explaining the relationships between task, reward, agent diversity and performance in multi-robot teams. From a practical perspective this model will help robot system designers select appropriate learning strategies for robot teams according to the task for which they are designed. From a more philosophical point of view, the model might help explain how and why diversity arises in natural and artificial societies.

This dissertation provides the framework for the model by identifying and defining the independent (task and reward) and dependent (performance and diversity) variables. Mathematical relations between the variables will be derived as results are gathered from more points in the experimental space. At this point we can only hypothesize on the basis of results gathered thus far:

First, in all three domains examined here global rewards generate the most diverse societies. In soccer and foraging the globally rewarded teams are significantly more diverse than other systems. In the cooperative movement task global and local rewards generate about the same degree of diversity in robot teams. It makes sense that global rewards lead to greater diversity because it is more likely agents will be rewarded for different behavior under global reinforcement. We can hypothesize that, in general, global rewards lead to greater diversity and that global rewards are the better choice in domains where behavioral diversity is important.

Similarly, local rewards encourage uniformity in agent behavior. There are several advantages to the use of local rewards. First, they usually require only local sensing – the extra cost of communication is not incurred. Also, in some tasks (e.g. soccer and foraging) local rewards provide more rapid learning (policy convergence). In the foraging task, where homogeneous teams provide the best behavior, we also find that local rewards generate the best performing teams. It is likely that local rewards are the best choice in tasks where homogeneous behavior is preferred.

These observations beg the question "when is diverse behavior preferred?" The results of this work show that diverse teams perform best in soccer and cooperative movement tasks. Foraging, however, seems to be a decidedly homogeneous task. The task classifications provide a clue: soccer and cooperative movement are both classified as `MULTI_AGENT`, while foraging is a `SINGLE_AGENT` task (this distinction is based on whether an individual agent could reasonably perform the task alone). We can speculate that homogeneous teams excel in SINGLE_AGENT tasks.

## 9.6 Future directions

Important future work includes the application of these tools in new and different multi-robot task domains. The author hopes other researchers will adopt the measures of behavioral difference and robot team diversity introduced here in the evaluation of new multi-robot systems. This will provide additional data points in the multiagent task/reward space and help us derive the relations between task, reward, diversity and performance more precisely.

Another important direction for future research is the extension of these tools to a broader range of robotic systems. The behavioral difference metric, for instance, is limited to the comparison of deterministic policies. Can we compare the behavior of agents coded in FSAs or more complex representations? It may also prove useful to extend and refine the classifications of task and reward. Perhaps a broader range of tasks can be described or more subtle distinctions can be drawn between them.

Finally, can the results of this research be applied in other fields? Researchers in behavior-based robotics often draw inspiration from biology and psychology; perhaps roboticists can provide tools for the sociobiologist. It is tempting, for instance, to draw parallels between robotic tasks and rewards and their counterparts in human and animal societies. Local and global reinforcement for robots, for example, bears a strong resemblance to the capitalist and socialist economies of human society. It would be presumptuous to suppose such comparisons are valid now, but as the research and theory mature we may gain insights into the origins and benefits of diversity in natural as well as artificial social systems.

# Appendix A

# Motor Schema Formulations and Gain Values

A motor schema-based approach to robot behavioral design is used in this work [Ark89]. Individual motor schemas, or primitive behaviors, express separate goals or constraints for a task. As an example, important schemas for a navigational task would include **avoid_static_obstacles** and **move_to_goal**. Since schemas are independent, they can run concurrently, providing parallelism and speed. Sensor input is processed by perceptual schemas embedded in the motor behaviors. Perceptual processing is minimal and provides just the information pertinent to the motor schema. For instance, a **find_obstacles** perceptual schema which provides a list of sensed obstacles is embedded in the **avoid_static_obstacles** motor schema.

The concurrently running motor schemas are integrated as follows. First, each produces a vector indicating the direction the robot should move to satisfy that schema's goal or constraint. The magnitude of the vector indicates the importance of achieving it. It is not so critical, for instance, to avoid an obstacle if it is distant, but crucial if close by. The magnitude of the **avoid_static_obstacles** vector is correspondingly small for distant obstacles and large for close ones. The importance of motor schemas relative to each other is indicated by a gain value for each one.

This appendix reports the methods by which each of the individual primitive motor schemas used in this research compute their component vectors. The gain values and parameters for schemas used in each behavioral assemblage are also listed.

## A.1 Foraging behaviors

Details regarding the high-level sequencing of behavioral assemblages for foraging are covered in Chapter 6. The motor schemas used to build the behavioral assemblages for foraging are described below. For each behavioral assemblage, several schemas are instantiated at once with appropriate parameter and gain values. The parameter and gain values used experimentally are listed in Table A.1.

- **noise**

  **description:** generates movement in a pseudo-random direction.
  **parameter:** $P$, persistence, the time in seconds between each change in direction.
  **mathematical formulation:** the vector is computed as follows:

$$\begin{aligned} V_{\text{direction}} &= \text{pseudo-random direction} \\ &\quad \text{between 0 and } 2\pi \\ V_{\text{magnitude}} &= 1 \end{aligned}$$

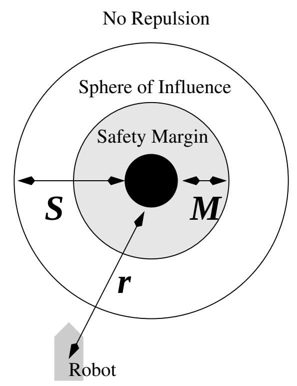



**Figure A.1**: Parameters used in the calculation of **avoid** motor schema vectors. The object to be avoided is represented as a black circle at the center of the illustration.

- **avoid_static_obstacles**

  **description:** repulsion from detected obstacles. The magnitude of repulsion varies with distance from each obstacle (Figure A.1). When beyond the sphere of influence ($S$), no repulsion is generated. Within the sphere of influence, repulsion increases linearly until the robot reaches the safety margin. When the robot is within the safety margin, the magnitude of repulsion is $\infty$.
  **parameters:** $S$, the sphere of influence beyond which detected obstacles have no effect; $M$, safety margin.

**mathematical formulation:** a separate vector is computed for each detected obstacle as follows, where $r$ is the distance from the center of the robot to closest point on the obstacle:

$$V_{\text{direction}} = \text{along a line from the center of the obstacle to the robot, moving away from obstacle}$$

$$V_{\text{magnitude}} = \begin{cases} 0 & \text{for } r > S \\ \frac{S-r}{S-M} & \text{for } M < r \leq S \\ \infty & \text{for } r \leq M \end{cases}$$

The overall **avoid_static_obstacles** vector is computed by summing the individual vectors calculated for each obstacle.

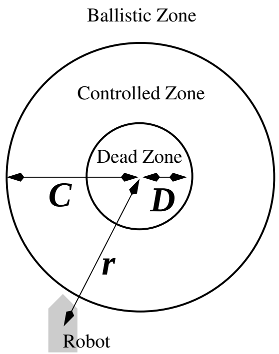



**Figure A.2**: Parameters used in the calculation of **move_to** motor schema vectors.

- **stay_near_homebase**
  **move_to_red_attractor**
  **move_to_blue_attractor**
  **move_to_red_bin**
  **move_to_blue_bin**

**description:** attraction to a detected object or goal location. The magnitude of attraction varies with distance from the goal. Figure A.2 illustrates three zones, defined by distance from the goal, used for magnitude computation. The radii of these zones are parameters of the schema. Outside the *controlled zone* attraction is set at a fixed maximum (1.0). Within the controlled zone attraction decreases linearly from 1.0 to 0.0 at the boundary of the *dead zone*. Inside the dead zone the magnitude is 0.0.

**parameters:** $C$, radius of the controlled zone; $D$, radius of the dead zone.

**mathematical formulation:** the vector is computed as follows, where $r$ is the distance from the center of the robot to the goal location:

$$V_{\text{direction}} = \begin{array}{l}\text{along a line from the robot}\\ \text{to the goal, moving to the goal}\end{array}$$

$$V_{\text{magnitude}} = \begin{cases} 1 & \text{for } r > C \\ \frac{r-D}{C-D} & \text{for } D < r \leq C \\ 0 & \text{for } r \leq D \end{cases}$$

- **avoid_robots**

**description:** repulsion from detected robots. The magnitude of repulsion varies with distance from each robot. When beyond the sphere of influence ($S$), no repulsion is generated. Within the sphere of influence, repulsion increases linearly until the robot reaches the safety margin. When the robot is within the safety margin, the magnitude of repulsion is $\infty$.

**parameters:** $S$, the sphere of influence beyond which detected robots have no effect; $M$, safety margin.

**mathematical formulation:** same as described for **avoid_static_obstacles**. A separate vector is computed for each detected robot and the vectors are summed to provide the overall value.

- **avoid_home_zone**

**description:** repulsion from the delivery area. Used to move agents away from this area during the search phase. The magnitude of repulsion varies with distance from the homebase. When far from the homebase, no repulsion is generated. As the robot approaches the area, repulsion increases linearly until the robot reaches the center of the zone.

**parameters:** $S$, the sphere of influence beyond which repulsion is zero.

**mathematical formulation:** the repulsion is computed as follows, where $r$ is the distance from the center of the robot to the center of the homebase:

$$V_{\text{direction}} = \begin{array}{l}\text{along a line from the homebase}\\ \text{to the robot, moving away from the base}\end{array}$$

$$V_{\text{magnitude}} = \begin{cases} 0 & \text{for } r > S \\ \frac{S-r}{S} & \text{for } r \leq S \end{cases}$$

- **avoid_delivered_attractors**

**description:** a localized repulsion centered on the delivery area with a small sphere of influence. Used to keep robots from pushing over previously delivered attractors.

**parameters:** $S$, the sphere of influence beyond which repulsion is zero.

**mathematical formulation:** the repulsion is computed as in the same manner as **avoid_home_zone**.

## A.2 Soccer behaviors

Details regarding the high-level sequencing of behavioral assemblages for soccer are covered in Chapter 7. The motor schemas used to build the behavioral assemblages for soccer are described below. For each behavioral assemblage, several schemas are instantiated at once with appropriate parameter and gain values. Experiments were conducted in two slightly different domains (simplified and RoboCup soccer). Behaviors used in both domains are identical except the **avoid_teammates** schema was not used in the simplified domain. The parameter and gain values used experimentally are listed in Table A.2.

- **move_to_kickspot**

  **description:** attractive force to draw the robot to a point one-half of a robot radius behind the ball. If the robot bumps the ball from that location, the ball is propelled in the direction of the opponent's goal. Parameters used in this motor schema are illustrated in Figure A.2.

  **parameters:** $C$, radius of the controlled zone; $D$, radius of the dead zone.

  **mathematical formulation:** the vector is computed as described for the **move_to** schemas used in foraging.

- **move_to_halfway_point**

  **description:** attractive force to draw the robot to a point halfway between the ball and the defended goal. Parameters used in this motor schema are illustrated in Figure A.2.

  **parameters:** $C$, radius of the controlled zone; $D$, radius of the dead zone.

  **mathematical formulation:** the vector is computed as described for the **move_to** schemas used in foraging.

- **move_to_defended_goal**

  **description:** attraction to the defended goal. A large dead zone centered on the goal area permits the robot to roam freely if it is near the goal. Parameters used in this motor schema are illustrated in Figure A.2.

  **parameters:** $C$, radius of the controlled zone; $D$, radius of the dead zone.

**Table A.1**: Motor schema parameter values and gains used in the behavioral assemblages for foraging.

| **behavioral assemblage** | **motor schemas** | **gain values** |
|---|---|---|
| *wander* | **noise** | 1.0 |
| search the environment for attractors | $P = 5.0$sec | |
| | **avoid_static_obstacles** | 1.0 |
| | $S = 1.5$m $M = 0.1$m | |
| | **avoid_robots** | 1.0 |
| | $S = 1.5$m $M = 0.1$m | |
| | **avoid_home_zone** | 2.0 |
| | $S = 4.0$m | |
| *stay_near_home* | **stay_near_homebase** | 1.0 |
| search the home zone for attractors | $C = 4.0$m, $D = 3.0$m | |
| | **avoid_static_obstacles** | 1.0 |
| | $S = 1.5$m $M = 0.1$m | |
| | **avoid_robots** | 1.0 |
| | $S = 1.5$m $M = 0.1$m | |
| | **noise** | 1.0 |
| | $P = 5.0$sec | |
| | **avoid_delivered_attractors** | 1.0 |
| | $S = 1.0$m | |
| *acquire_blue* | **move_to_{red\|blue}_attractor** | 1.0 |
| *acquire_red* | $C = 0.4$m, $D = 0.0$m | |
| acquire the closest red or blue attractor | **avoid_static_obstacles** | 1.0 |
| | $S = 1.5$m $M = 0.1$m | |
| | **avoid_robots** | 1.0 |
| | $S = 1.5$m $M = 0.1$m | |
| | **noise** | 0.2 |
| | $P = 5.0$sec | |
| | **avoid_delivered_attractors** | 1.0 |
| | $S = 1.0$m | |
| *deliver_red* | **move_to_{red\|blue}_bin** | 1.0 |
| *deliver_blue* | $C = 0.4$m, $D = 0.0$m | |
| go to the red or blue bin | **avoid_static_obstacles** | 1.0 |
| | $S = 1.5$m $M = 0.1$m | |
| | **avoid_robots** | 1.0 |
| | $S = 1.5$m $M = 0.1$m | |
| | **noise** | 0.2 |
| | $P = 5.0$sec | |
| | **avoid_delivered_attractors** | 1.0 |
| | $S = 1.0$m | |

**mathematical formulation:** the vector is computed as described for the **move_to** schemas used in foraging.

- **avoid_teammates**

  **description:** repulsion from robots on the same soccer team. The magnitude of repulsion varies with distance from each robot. When beyond the sphere of influence ($S$), no repulsion is generated. Within the sphere of influence, repulsion increases linearly until the robot reaches the safety margin. When the robot is within the safety margin, the magnitude of repulsion is $\infty$.

  **parameters:** $S$, the sphere of influence beyond which detected robots have no effect; $M$, safety margin.

  **mathematical formulation:** same as described for **avoid_static_obstacles** in the foraging section. A separate vector is computed for each detected robot and the vectors are summed to provide the overall value.

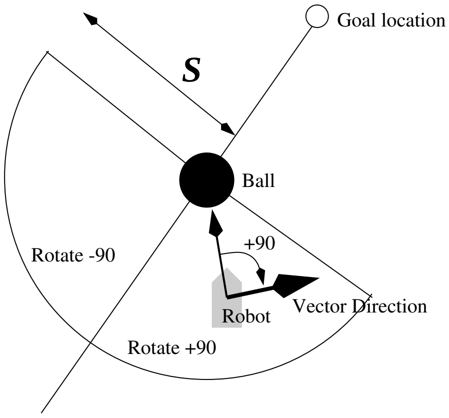



**Figure A.3**: Parameters used in the calculation of the **swirl_ball** motor schema vector.

- **swirl_ball**

  **description:** a dodging behavior used to keep robots from colliding with the ball as they attempt to move behind it. The direction of the vector is perpendicular to a line drawn from the robot through the ball. The magnitude of the vector depends on distance to the ball. When beyond the sphere of influence ($S$), the magnitude is zero. Within the sphere of influence, magnitude increases linearly until the robot reaches the ball. If the robot is past the ball, the magnitude is zero.

  **parameter:** $S$, the sphere of influence beyond which vector magnitude is zero.

  **mathematical formulation:** the vector is computed as follows, where $r$ is the distance from the center of the robot to the center of the ball:

$$V_{\text{direction}} = \text{a line from the robot to the ball}$$

rotated either +90 or −90
degrees such that the rotation
sweeps through the goal

$$V_{\text{magnitude}} = \begin{cases} 0 & \text{for } r > S \text{ or robot is past the ball} \\ \frac{S-r}{S} & \text{for } r \leq S \end{cases}$$

**Table A.2**: The **avoid_teammates** schema was not used in the simplified soccer domain experiments.

| behavioral assemblage | motor schemas | gain values |
|---|---|---|
| *move_to_ball* | **move_to_kickspot**<br>$C = 0.0\text{m}$ $D = 0.0$<br>**avoid_teammates**<br>$S = 1.0\text{m}$ $M = 0.1\text{m}$ | 1.0<br><br>0.8 (RoboCup)<br>0.0 (simplified) |
| *get_behind_ball*<br>search the home zone for attractors | **move_to_halfway_point**<br>$C = 0.0\text{m}$ $D = 0.0$<br>**swirl_ball**<br>$S = 0.5\text{m}$<br>**avoid_teammates**<br>$S = 1.0\text{m}$ $M = 0.1\text{m}$ | 1.0<br><br>1.2<br><br>0.8 (RoboCup)<br>0.0 (simplified) |
| *move_to_backfield* | **move_to_defended_goal**<br>$C = 0.3\text{m}$, $D = 0.2\text{m}$<br>**move_to_kickspot**<br>$S = 0.0\text{m}$ $M = 0.0\text{m}$ | 1.2<br><br>1.2 |

## A.3 Formation behaviors

The overall navigational strategy is similar to the approach developed in earlier research [BA95b]: several motor schemas, **move_to_goal**, **avoid_static_obstacles**, **avoid_robots** and **maintain_formation** implement the overall behavior for a robot to move to a goal location while avoiding obstacles, collisions with other robots and remaining in formation. An additional low-gain attractive force, **move_to_unit_center**, draws all of the robots together.

Formation maintenance is accomplished in two steps: first, a perceptual process, **detect_formation_position**, determines the robot's proper position in formation based on current sensor data; second, the motor process **maintain_formation**, generates motor commands to direct the robot toward the correct location. The perceptual process used by robots for determining their position in a formation is discussed in Chapter 8.

The motor schemas used to build the navigational behavioral assemblages for cooperative movement are described below. The *diamond*, *line* and *column* assemblages are identical except for the perceptual process used to determine the robot's desired position in formation. The parameter and gain values used experimentally are listed in Table A.3.

- **move_to_goal**

  **description:** attractive force to draw the robot to a goal location. The goal is positioned 1000 meters beyond the finish line. The robots never actually reach the goal in experimental trials because each trial terminates when the robots cross the finish line. Parameters used in this motor schema are illustrated in Figure A.2.

  **parameters:** $C$, radius of the controlled zone; $D$, radius of the dead zone.

  **mathematical formulation:** the vector is computed as described for the **move_to** schemas used in foraging.

- **maintain_formation**

  **description:** attractive force to draw the robot into the proper formation position. Parameters used in this motor schema are illustrated in Figure A.2.

  **parameters:** $C$, radius of the controlled zone; $D$, radius of the dead zone.

  **mathematical formulation:** the vector is computed as described for the **move_to** schemas used in foraging.

- **move_to_unit_center**

  **description:** a low-gain attractive force, added to draw all of the robots together. Parameters used in this motor schema are illustrated in Figure A.2.

  **parameters:** $C$, radius of the controlled zone; $D$, radius of the dead zone.

  **mathematical formulation:** the vector is computed as described for the **move_to** schemas used in foraging.

- **avoid_static_obstacles** same as described for foraging.
- **avoid_robots** same as described for foraging.
- **noise** same as described for foraging.

**Table A.3:**

| behavioral assemblage | motor schemas | gain values |
|---|---|---|
| *diamond* | **avoid_static_obstacles** | 1.1 |
| *line* | $S = 2.0$m $M = 0.1$ | |
| *column* | **avoid_robots** | 1.1 |
| | $S = 2.0$m $M = 0.1$m | |
| | **move_to_goal** | 0.7 |
| | $C = 0.0$m, $D = 0.0$m | |
| | **maintain_formation** | 1.3 |
| | $C = 1.0$m $D = 0.0$m | |
| | **move_to_unit_center** | 0.6 |
| | $C = 3.0$m, $D = 2.0$m | |
| | **noise** | 0.1 |
| | $P = 5.0$sec | |
| *no_formation* | **avoid_static_obstacles** | 1.1 |
| | $S = 2.0$m $M = 0.1$ | |
| | **avoid_robots** | 1.1 |
| | $S = 2.0$m $M = 0.1$m | |
| | **move_to_goal** | 0.7 |
| | $C = 0.0$m, $D = 0.0$m | |
| | **move_to_unit_center** | 0.6 |
| | $C = 3.0$m, $D = 2.0$m | |
| | **noise** | 0.1 |
| | $P = 5.0$sec | |

# Appendix B

# Spearman's Rank-Order Correlation Test

Spearman's rank-order correlation test enables us to determine whether rank-ordered data is correlated. The key concept of non-parametric (rank-ordered) correlation is this: if we replace the value of each $x_i$ by the value of its *rank* among all the other $x_i$'s in the sample, that is $1, 2, 3, ..., N$, then the resulting list of numbers will be drawn from a perfectly known uniform distribution. Of course we do the same for the $y_i$s, replacing each value by its rank among the others in the sample. Let $R_i$ be the rank of $x_i$ among the other $x$s, $S_i$ be the rank of $y_i$ among the other $y$s, ties being assigned a mid-rank. The rank-order correlation is defined to be the linear correlation coefficient of the ranks:

$$r_s = \frac{\sum_i (R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\sum_i (R_i - \bar{R})^2}\sqrt{\sum_i (S_i - \bar{S})^2}} \tag{B.1}$$

A value of $r_s = 1$ indicates complete positive correlation, $r_s = -1$ indicates complete negative correlation and $r_s = 0$ indicates no correlation of the data.

The statistical significance of $r_s$ is tested by computing the probability that $x$ and $y$ are uncorrelated and the the ranking occurred by chance (the null hypothesis):

$$p = \operatorname{erfc}\left(\frac{|r_s|\sqrt{N}}{\sqrt{2}}\right) \tag{B.2}$$

where $\operatorname{erfc}(x)$ is the complementary error function [PTVF88]. Values of $p$ less than 0.05 indicate the correlation is statistically significant.

The *Numerical Recipes in C* software package was used to compute the values of Spearman's coefficient reported in this dissertation [PTVF88].

# Bibliography

[AAMR88] C. Atkeson, E. Aboaf, J. McIntyre, and D. Reinkensmeyer. Model-based robot learning. Technical Report AIM-1024, MIT, 1988.

[AB97] R.C. Arkin and T.R. Balch. AuRA: principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.

[ABN93] R.C. Arkin, T. Balch, and E. Nitz. Communication of behavioral state in multi-agent retrieval tasks. In *Proceedings 1993 IEEE Conference on Robotics and Automation*, Atlanta, GA, 1993.

[Ali97] K. Ali. Phd thesis proposal. College of Computing, Georgia Institute of Technology, 1997.

[AM94] R.C. Arkin and D. MacKenzie. Temporal coordination of perceptual algorithms for mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 10(3):276–286, June 1994.

[Ark89] R.C. Arkin. Motor schema-based mobile robot navigation. *International Journal of Robotics Research*, 8(4):92–112, 1989.

[Ark92] R.C. Arkin. Cooperation without communication: Multi-agent schema based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[BA93] T. Balch and R.C. Arkin. Avoiding the past: a simple but effective strategy for reactive navigation. In *IEEE Conference on Robotics and Automation*, pp 678–685, May 1993. Atlanta, Georgia.

[BA95a] T. Balch and R.C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1), 1995.

[BA95b] T. Balch and R.C. Arkin. Motor schema-based formation control for multiagent robot teams. In *First International Conference on Multiagent Systems*, June 1995. San Francisco.

[BA99] T. Balch and R.C. Arkin. Behavior-based formation control for multiagent robot teams. *IEEE Transactions on Robots and Automation*, to appear.

[Bai90] K. Bailey. *Social Entropy Theory.* State University of New York Press, Albany, 1990.

[Bal] T. Balch. JavaBots. `www.cc.gatech.edu/~tucker/JavaBots`.

[Bal97a] T. Balch. Clay: Integrating motor schemas and reinforcement learning. College of Computing Technical Report GIT-CC-97-11, Georgia Institute of Technology, Atlanta, Georgia, March 1997.

[Bal97b] T. Balch. Social entropy: a new metric for learning multi-robot teams. In *Proc. 10th International FLAIRS Conference (FLAIRS-97)*, May 1997.

[Bal97c] T. Balch. Learning roles: Behavioral diversity in robot teams. In *AAAI-97 Workshop on Multiagent Learning*, Providence, R.I., 1997.

[Bal98] T. Balch. Integrating robotics research with JavaBots. In *AAAI-98 Spring Symposium*, March 1998.

[Bal98a] T. Balch. Personal communication. Georgia Institute of Technology, December 1998.

[BBC+95] T. Balch, G. Boone, T. Collins, H. Forbes, D. MacKenzie, and J. Santamaría. Io, Ganymede and Callisto - a multiagent robot trash-collecting team. *AI Magazine*, 16(2):39–51, 1995.

[Bel57] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, N.J., 1957.

[Bev90] H. Bevan. On the continuity between information theory entropy and statistical diversity. *Elektrontechnik und Informationstechnik*, 107(12):601–604, 1990.

[Boo97] G. Boone. Efficient reinforcement learning: Model-based acrobot control. In *IEEE International Conference on Robotics and Automation*, pp 229–234, Albuquerque, NM, 1997.

[Bro86] R. Brooks. A robust layered control system for a mobile robot. *IEEE Jour. of Robotics and Auto.*, RA-2(1):14, 1986.

[BSW90] A. Barto, R. Sutton, and C. Watkins. Learning and sequential decision making. In Michael Gabriel and John Moore, editors, *Learning and computational neuroscience : Foundations of adaptive networks*. M.I.T. Press, Cambridge, Mass, 1990.

[CAR92] R.J. Clark, R.C. Arkin, and A. Ram. Learning momentum: On-line performance enhancement for reactive systems. In *IEEE Conf. on Robotics and Automation*, pp 111–116, May 1992. Nice, France.

[CG93] C. Connolly and R. Grupen. On the applications of harmonic functions to robotics. *Journal of Robotic Systems*, 10(7):931–936, 1993.

[CGH96] D. J. Cook, P. Gmytrasiewicz, and L.B. Holder. Decision-theoretic cooperative sensor planning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1013–23, 1996.

[Com97] RoboCup Committee. Robocup real robot league regulations. `http://www.robocup.org`, 1997.

[CS96] P. Cheeseman and J. Stutz. Bayesian Classification (AutoClass): Theory and Results. Chapter in *Advances in Knowledge Discovery and Data Mining*, Kluwer, 1996.

[CSB65] J.M. Cullen, E. Shaw, and H.A. Baldwin. Methods for measuring the three-dimensional structure of fish schools. *Animal Behavior*, 13:534–543, 1965.

[Dem92] L. Demetrius. The thermodynamics of evolution. *Physica A*, 189(3-4):417–436, November 1992.

[DF94] A. Drogoul and J. Ferber. From tom thumb to the dockers: Some experiments with foraging robots. In *From Animals to Animats: Proc.*

*2nd International Conference on the Simulation of Adaptive Behavior*, pp 451–459, Honolulu, HI, 1994.

[DJW93] G. Dudek, E. Jenkin, and D. Wilkes. A taxonomy for swarm robots. In *Proc. 1993 IEEE International Conference on Intelligent Robots and Systems*, pp 441–447, 1993.

[FM97] M. Fontan and M. Mataric. A study of territoriality: The role of critical mass in adaptive task division. In *From Animals to Animats 4: Proceedings of the Fourth International Conference of Simulation of Adaptive Behavior*, pp 553–561, 1997.

[For92] U.S. Air Force. *Air Combat Command Manual 3-3.* Department of the Air Force, Washington, D.C., 1992.

[GM97] D. Goldberg and M. Mataric. Interference as a tool for designing and evaluating multi-robot controllers. In *Proceedings, AAAI-97*, pp 637–642, July 1997.

[HW90] B. Holldobler and E. Wilson. *The Ants.* Belknap Press, 1990.

[JS71] N. Jardine and R. Sibson. *Mathematical Taxonomy.* John Wiley & Sons, 1971.

[KAK+97] M. Kitano, M. Asada, Y. Kuniyoushi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proc. Autonomous Agents 97*, Marina Del Rey, California, 1997.

[KLM96] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Koz92] J. Koza. *Genetic Programming - On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, 1992.

[LCK95] M. Littman, A. Cassandra, and L. Kaelbling. Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[LDK95] M. Littman, T. Dean, and L. Kaelbling. On the complexity of solving markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, May 1995.

[Lin91] Long-Ji Lin. Programming robots using reinforcement learning and teaching. In *Proceedings, AAAI-91*, 1991.

[Lin92] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, August 1992.

[Lin93] Long-Ji Lin. Hierachical learning of robot skills by reinforcement. In *International Conference on Neural Networks*, 1993.

[LVW83] D. Lurie, J. Valls, and J. Wagensberg. Thermodynamic approach to biomass distribution in ecological systems. *Bulletin of Mathematical Biology*, 45(5):869–872, 1983.

[LW80] D. Lurie and J. Wagensberg. Information theory and ecological diversity. In L. Garrido, editor, *Systems Far from Equilibrium*, pp 290–303, Springer-Verlag, Berlin, 1980.

[MAC97] D. MacKenzie, R. Arkin, and J. Cameron. Multiagent mission specification and execution. *Autonomous Robots*, 4(1):29–52, 1997.

[Mag88] A.E. Magurran. *Ecological Diversity and Its Measurement.* Princeton University Press, 1988.

[Mat92] M. Matarić. Designing emergent behaviors: From local interactions to collective intelligence. In *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats 2*, pp 432–441, 1992.

[Mat94] M. Matarić. Learning to behave socially. In *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3*, 1994.

[Mat97] M. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, January 1997.

[May98] M.M. Mayoral. Renyi's entropy as an index of diversity in simple-stage cluster sampling. *Information Sciences*, 105(1-4):101–114, March 1998.

[MB90] P. Maes and R. Brooks. Learning to coordinate behaviors. In *Proceedings, AAAI-90*, pp 796–802, 1990.

[MC92] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, pp 311–365, 1992.

[MCA95] D.C. MacKenzie, J.M. Cameron, and R.C. Arkin. Specification and execution of multiagent missions. In *IEEE International Workshop on Intelligent Robots and Systems (IROS '95)*, 1995.

[MGL92] I. Martinez, M.A. Gil, and M.T. Lopez. Analysis of mutual information measures in cluster sampling. *Applied Mathematics and Computation*, 52(2-3):389–402, December 1992.

[MM92] P. Meyer and S. McIntosh. The USA Today index of ethnic diversity. *International Journal of Public Opinion Research*, page 56, Spring 1992.

[MW89] Merriam-Webster. *Webster's ninth new collegiate dictionary.* Merriam-Webster, 1989.

[Nil84] N. Nilsson. Shakey the robot. SRI International Tech. Note 323, 1984.

[NMH96] I. Noda, H. Matsubara, and K. Hiraki. Learning cooperative behavior in multi-agent environment: a case study of choice of play-plans in soccer. In *Proc. of 4th Pacific Rim International Conference on Artificial Intelligence*, pp 570–579, 1996. Cairns, Autralia.

[PAR92] M. Pearce, R.C. Arkin, and A. Ram. The learning of reactive control parameters through genetic algorithms. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp 130–137, 1992. Raleigh, NC.

[Par93] L. Parker. Designing control laws for cooperative agent teams. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pp 582–587. IEEE, 1993.

[Par94] L. Parker. *Heterogeneous Multi-Robot Cooperation.* PhD thesis, M.I.T., 1994.

[PPM92] L. Pardo, J.A. Pardo, and M.L. Menendez. Unified (r,s)-entropy as an index of diversity. *Journal of the Franklin Institute*, 329(5):907–921, September 1992.

[PTVF88] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1988.

[RS93] A. Ram and J.C. Santamaría. Multistrategy learning in reactive control. *Informatica*, 17(4):347–369, 1993.

[Sah94] M.K. Sahota. Reactive deliberation: An architecture for real-time intelligent control in dynamic environments. In *Proceedings, AAAI-94*, pp 1303–1308. AAAI, 1994.

[Sha49] C. E. Shannon. *The Mathematical Theory of Communication.* University of Illinois Press, 1949.

[SS73] P. Sneath and R. Sokal. *Numerical Taxonomy.* W. H. Freeman and Company, San Francisco, 1973.

[SSR98] J. Santamaria, R. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6(2):163–217, 1998.

[Sto98] P. Stone. Personal communication. Carnegie Mellon University, April 1998.

[Sut91] R. Sutton. DYNA, an Integrated Architecture for Learning, Planning and Reacting. In *Working Notes of the AAAI Spring Symposium on Integrated Intelligent Architectures*, March 1991.

[SV98] P. Stone and M. Veloso. Layered approach to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence*, 12(2-3):165–188, 1998.

[SWS98] R. P. Sałustowicz, M. A. Wiering, and J. Schmidhuber. Learning team strategies: Soccer case studies. *Machine Learning*, 1998. To appear.

[Tes92] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–277, 1992.

[TVR96] J. Tsitsiklis and B Van Roy. An analysis of temporal-difference learning with function approximation. Technical Report LIDS-P-2322, M.I.T. Laboratory for Information and Decision Systems, 1996.

[Veh87] S. L. Veherencamp. Individual, kin, and group selection. In P. Marler and J.G. Vandenbergh, editors, *Handbook of Behavioral Neurobiology, Volume 3: Social Behavior and Communication*, pp 354–382, Plenum Press, New York, 1987.

[WD92] C. Watkins and P. Dayan. Technical note: Q learning. *Machine Learning*, 8:279–292, 1992.

[Wil92] E.O. Wilson. *The Diversity of Life*. Norton, 1992.

[YS93] H. Yanco and L. Stein. An adaptive communication protocol for cooperating mobile robots. In *From Animals to Animats: Proc. 2nd International Conference on the Simulation of Adaptive Behavior*, pp 478–485, Boston, 1993.