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

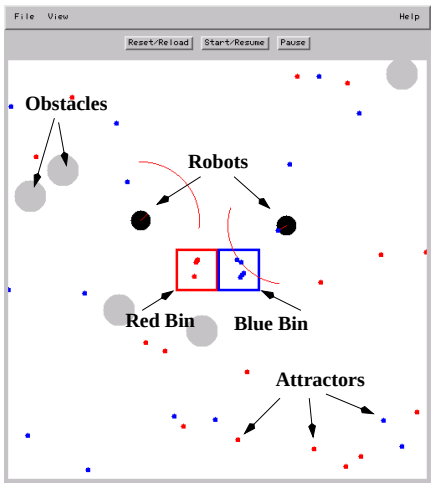

Figure 6.1: Simulation of the multi-forage task. Robots are represented as black circles; arcs indicate the visual sensing range. Obstacles are drawn as gray circles. The small discs are attractors. The robots deliver the attractors to the color-coded squares representing delivery areas.

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

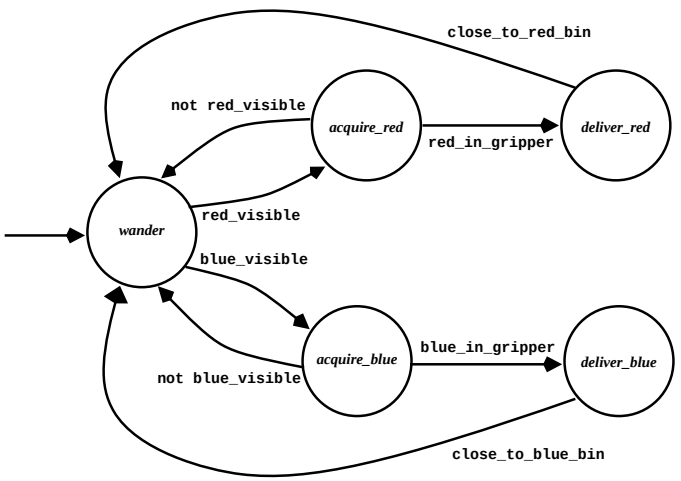

Figure 6.3: An FSA representing the homogeneous foraging strategy. This kind of agent can collect all types of attractor.

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

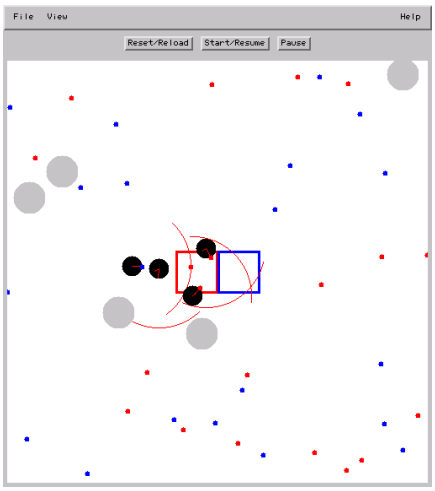

Figure 6.4: Inter-robot interference. In this simulation three robots are attempting to deliver attractors to the left (red) delivery area simultaneously. Interference reduces the rate at which agents can deliver attractors.

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-

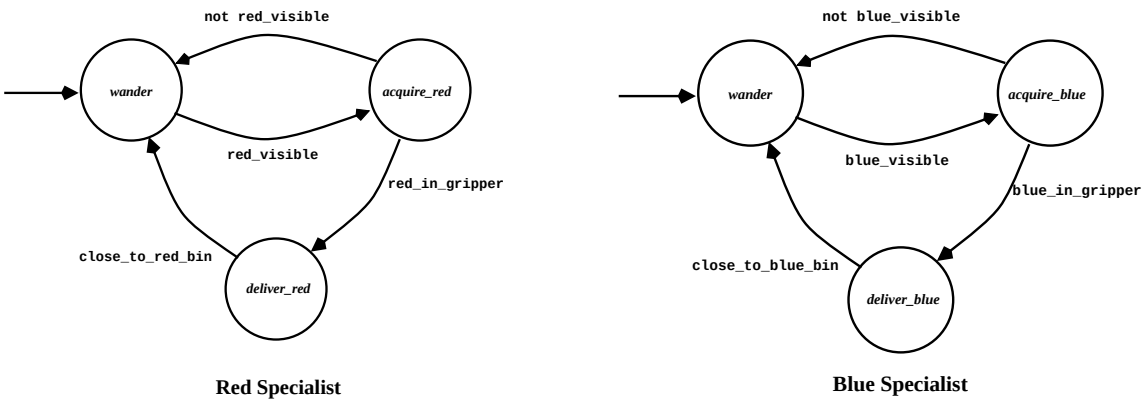

Figure 6.5: FSAs representing specialized behaviors for foraging. The FSA on the left shows the sequence behaviors are activated in for an agent specializing in collecting red attractors. The FSA on the right shows the sequence for blue specialists.

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

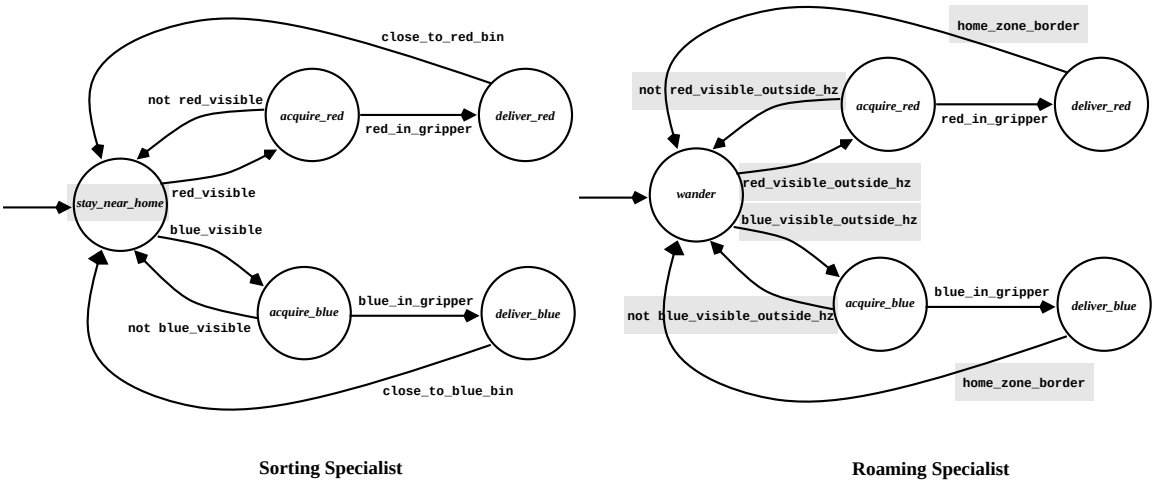

Figure 6.6: The Territorial behavioral sequences for foraging. FSA on the left shows the sequence of behaviors for an agent that remains close to the homebase, completing the delivery of the attractors. Agents using the strategy on the right search for attractors away from the home zone and deliver them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

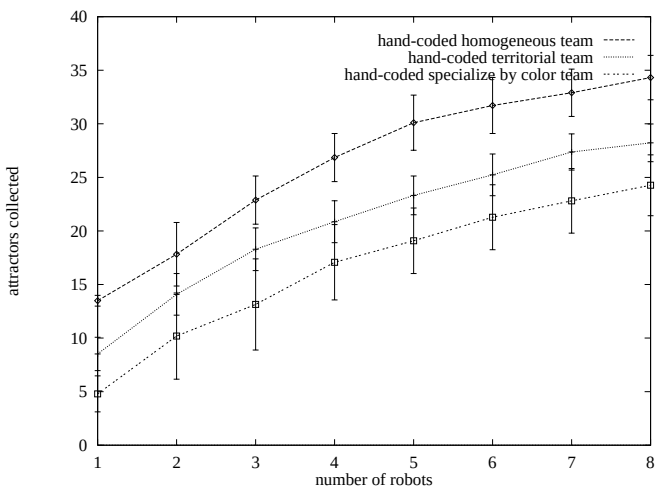

Figure 6.7: Performance of the hand-coded teams versus size of the team. Higher numbers are better; error bars indicate 95% confidence intervals. The homogeneous teams perform best in all cases.

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

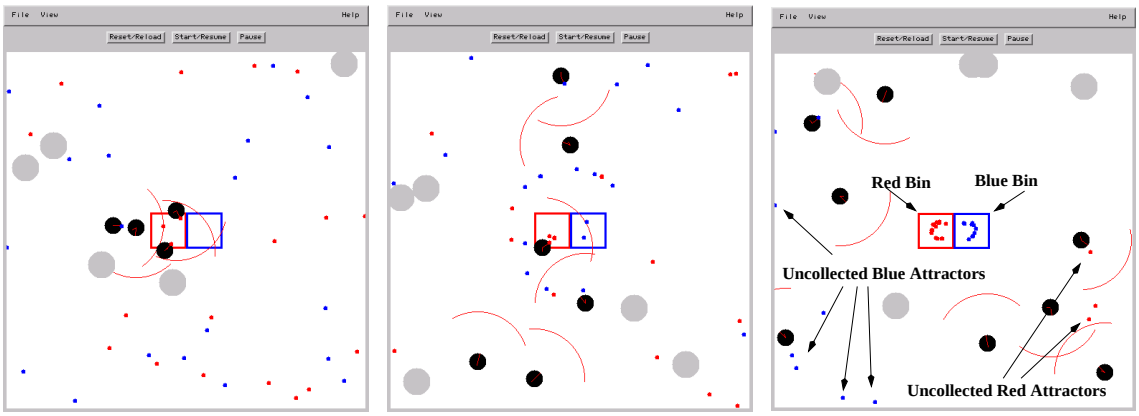

Figure 6.8: Simulations highlighting some of the factors that impact performance in hand-coded teams. From left to right: interference in a homogeneous team, an over-worked sorting robot, attractors left uncollected by agents specializing in one color of attractor.

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.

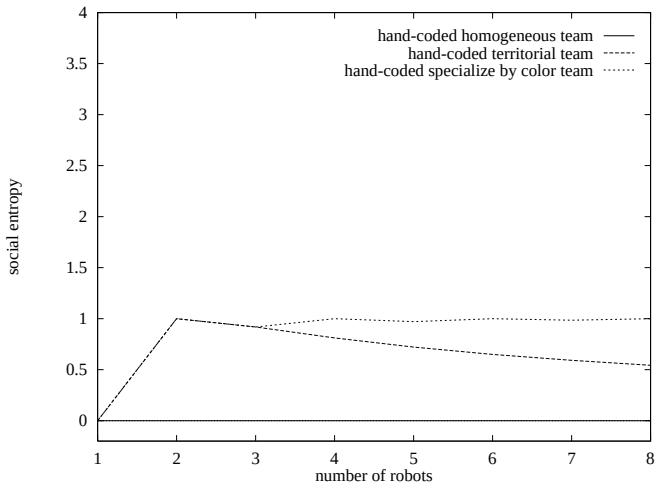

Figure 6.9: Diversity, as measured by social entropy, versus size of the team for hand-coded teams. Larger numbers indicate greater diversity.

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_{global} and R_{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_{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_{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.

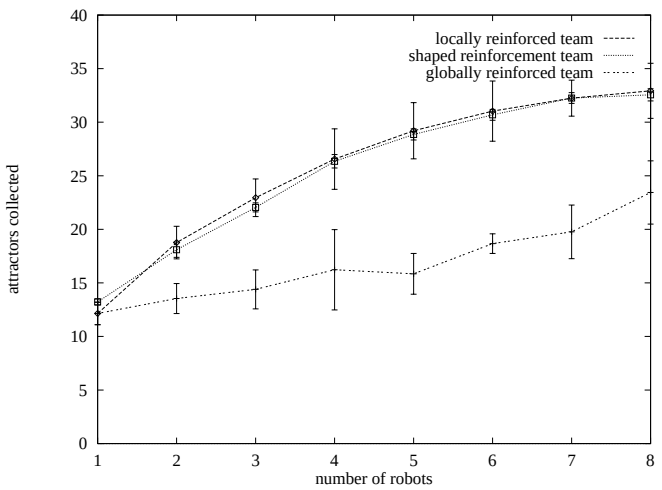

Figure 6.10: Performance for learning systems versus the number of robots on a team.

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.

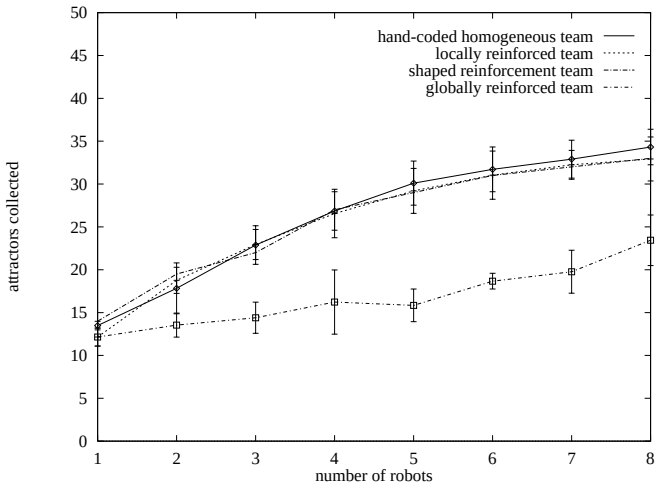

Figure 6.11: Performance of the best learning and non-learning systems combined in one graph.

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

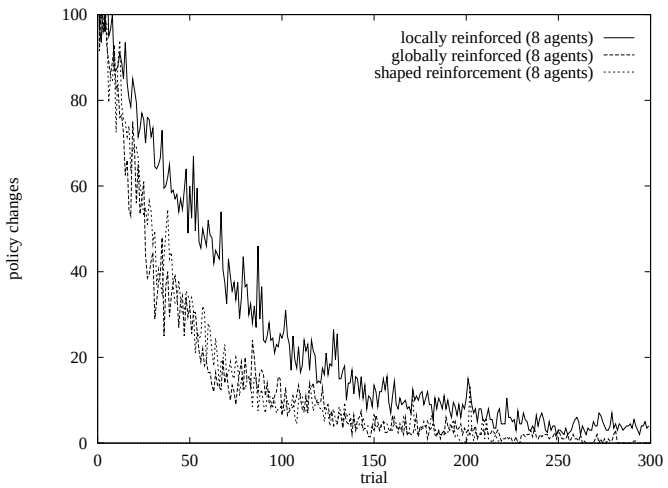

Figure 6.12: Convergence for learning systems, measured as policy changes per trial; lower numbers indicate convergence to a stable policy. Shaped reinforcement shows the best convergence properties (reaching zero after 220 trials).

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 ϵ-equivalent if their behavioral difference is

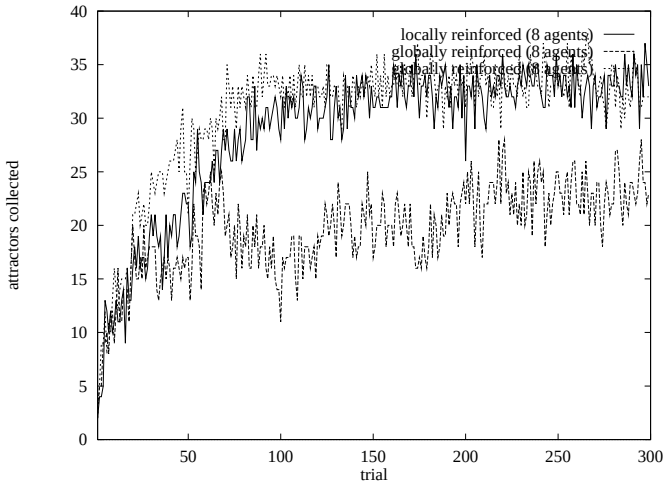

Figure 6.13: Performance versus trial number for learning systems with eight robots. These plots indicate how performance improves as the agents learn over time. Local and shaped-reinforcement teams stabilize at about the same level, with globally-reinforced teams performing worst.

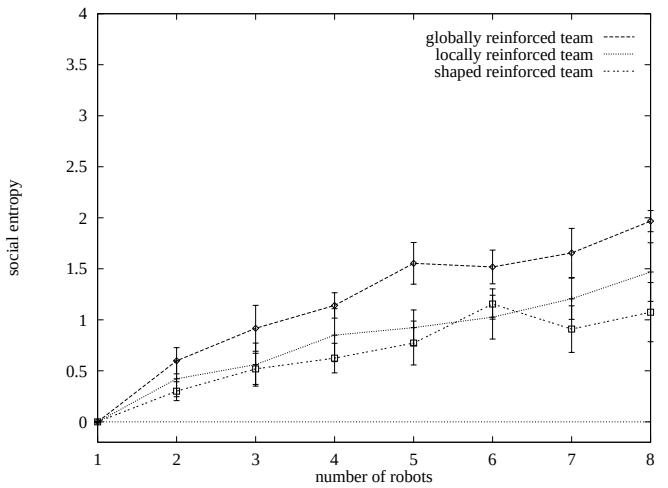

Figure 6.14: Hierarchic social entropy versus size of the team for learning teams; larger numbers indicate greater diversity, error bars indicate 95% confidence intervals. Shaped-reinforcement generates the least diverse teams, while globally-reinforced teams are the most diverse. For some robot team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

less than or equal to ϵ. 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 ϵ, 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 ϵ) 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.

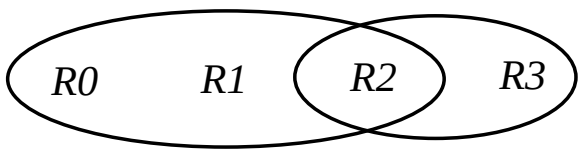

Figure 6.16: Division of the example team into castes based on behavioral difference. In this case $\epsilon = 0.2$.

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.

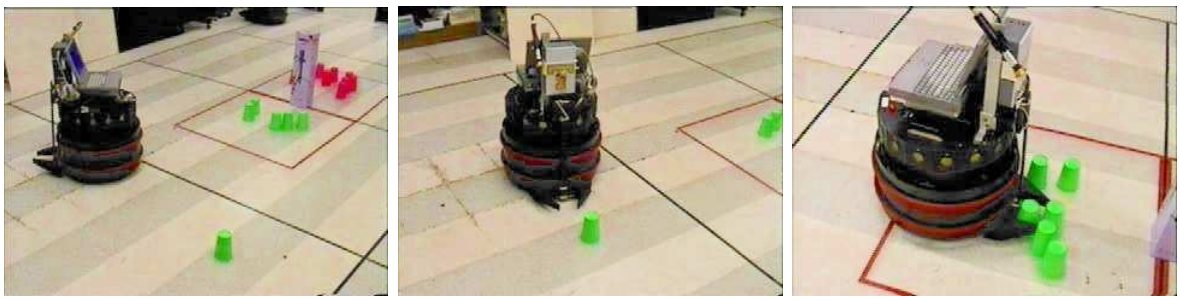

Figure 6.17: Nomad 150 robot equipped with passive gripper demonstrates three foraging behaviors. From left to right: *wander*, *acquire_blue* and *deliver_blue*.

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

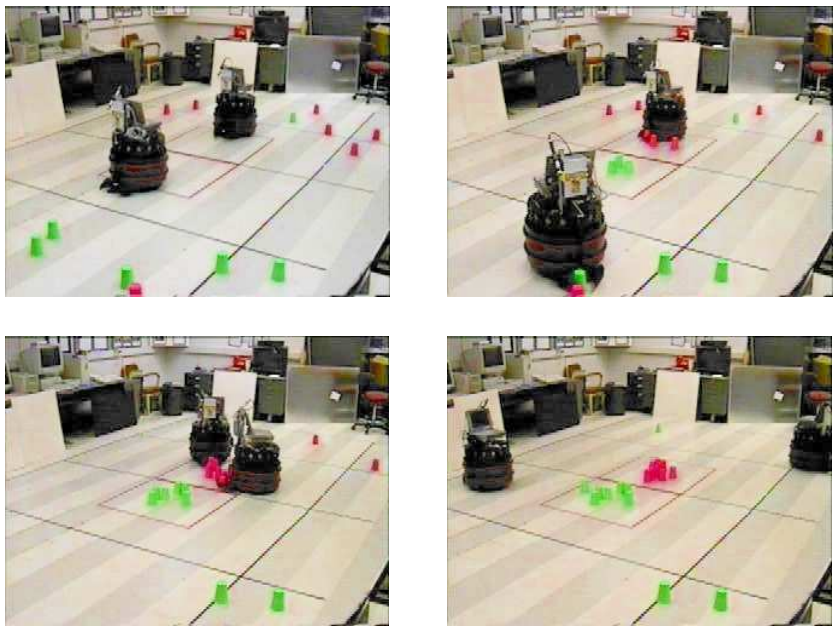

Figure 6.18: Two robots demonstrate learned foraging polices in the Mobile Robot Lab. A total of 16 attractors were collected in this 10 minute trial. Sequence is from top left to bottom right.

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.

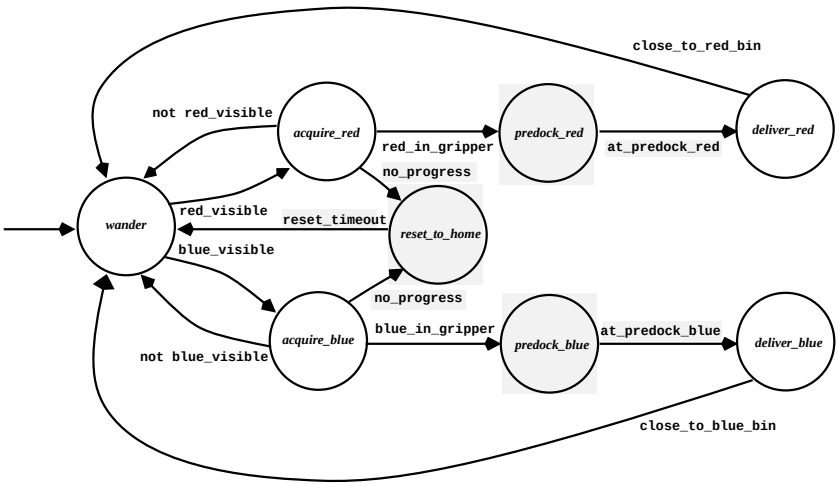

Figure 6.19: FSA representing the behavioral configuration used by Lewis and Clark at the AAAI-97 Mobile Robot Competition. Some additional behaviors and perceptual features (highlighted in gray shading) were added to address specific differences between the contest task and the general multi-foraging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

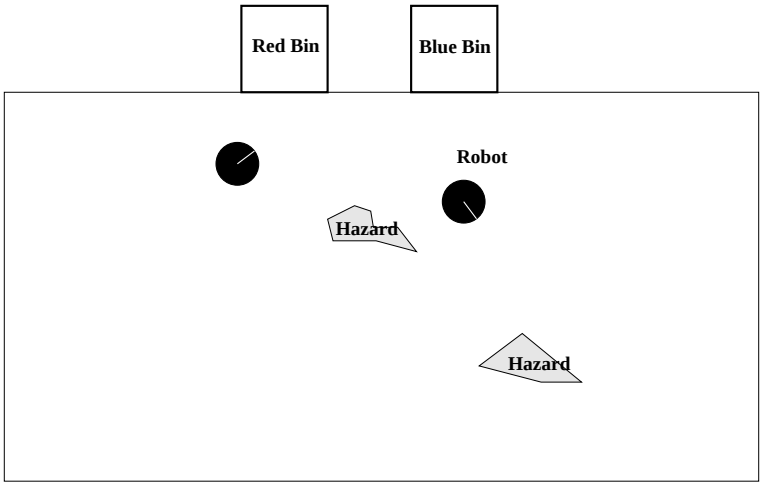

Figure 6.21: Diagram of laboratory arrangement for mobile robot experiments in the "Find Life on Mars" task.

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.