

Chapter 2

Problem Description

This chapter describes the problem setting investigated in this book. Rather than considering unified teams of agents designed by a single designer, we consider controlling a single agent on a newly created team. The following sections describe the problem in more depth as well as the evaluation framework used to measure performance on this problem. In addition, this chapter investigates an approach for describing the different dimensions of ad hoc team problems.

2.1 Ad Hoc Teamwork

Robots are becoming cheaper and more durable and are therefore being deployed in more environments for longer periods of time. As robots continue to proliferate in this way, many of them will encounter and interact with a variety of other kinds of robots. In many cases, these interacting robots will share a set of common goals, in which case it will be desirable for them to cooperate with each other. In order to effectively perform in new environments and with changing teammates, they should observe their teammates and adapt to achieve their shared goals. For example, after a disaster, it is helpful to use robots to search the site and rescue survivors. However, the robots may come from a variety of sources and may not be designed to cooperate with each other, such as in the response to the 2011 Tohoku earthquake and tsunami [2–5]. These robots were used to investigate the Fukushima Daiichi Nuclear Power Station, clear a fishing port, and find victims trapped underwater. These robots were remotely controlled and therefore derived any cooperation from their human operators. Robots that operate autonomously will have to be designed for cooperation as they will not have human operators providing cooperation. If these autonomous robots are not pre-programmed to cooperate, they may not share information about which areas

This chapter contains material from the publication: [1].

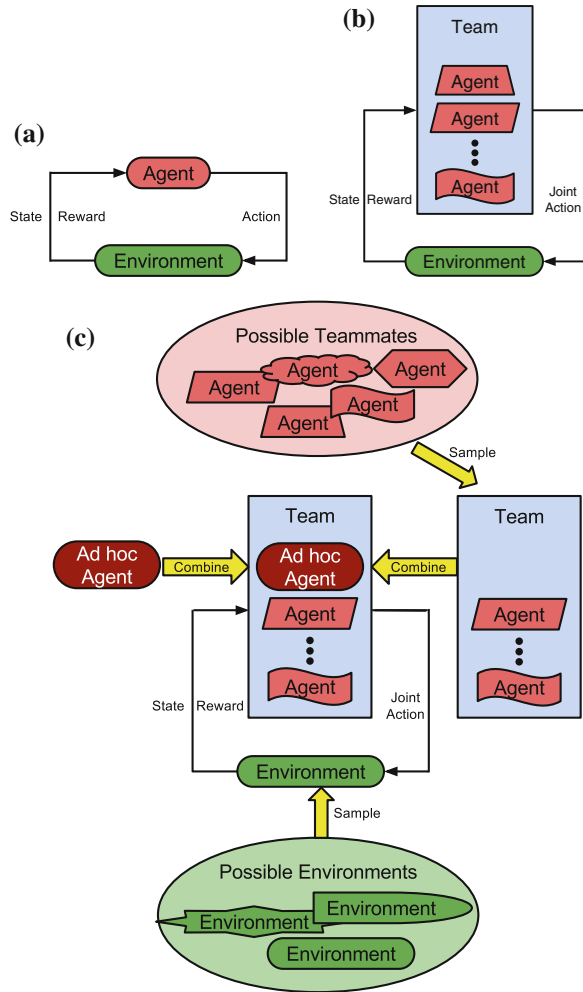
have been searched; or worse, they may unintentionally impede their teammates' efforts to rescue survivors. Therefore, in the future, it is desirable for robots to be designed to observe their teammates and adapt to them, forming a cohesive team that quickly searches the area and rescues the survivors.

This idea epitomizes the spirit of *ad hoc teamwork*. In ad hoc teamwork settings, agents encounter a variety of teammates and try to accomplish a shared goal. Ideally, agents designed for ad hoc teamwork try to quickly learn about their teammates and figure out how they should try to fit into the team. Agents that reason about ad hoc teamwork will be robust to changes in teammates and the environment. They must be adaptive and resourceful, learning how to accomplish the team's goals.

In this book, the word “agent” refers to an entity that repeatedly senses its environment and takes actions that affect this environment. Robots are examples of agents, as are software agents that bid for advertisements. As a shorthand, the terms ad hoc team agent and ad hoc agent are used in this book to refer to an agent that reasons about ad hoc teamwork. The environment includes the dynamics of the world the agent interacts with, as well as defining the observations received by the agent. In addition, the ad hoc agent will have to interact with teammate agents that are attempting to accomplish the same goals as the ad hoc agent. This book considers ad hoc agents that explicitly reason about the behaviors of their teammates separately from the environment because this factoring significantly reduces the complexity of the learning problem. Previous work has largely assumed that all agents in the domain will act as a unified team and are designed to work with their specific teammates [6–9]. On the other hand, this book will focus on creating a single agent that cooperates with teammates coming from a variety of sources without directly altering the behavior of these teammates. This agent will need to adapt to these different teammates and learn to cooperate with them on the fly.

The differences of this book from prior work are presented visually in Fig. 2.1. One existing area of research into how agents should behave is reinforcement learning (RL). Generally, RL problems revolve around a single agent learning by interacting with its environment. In RL problems, agents receive sparse feedback about the quality of sequences of actions. Generally, RL algorithms model other agents as part of the environment and try to learn the best policy for the single agent given this environment. In addition, RL algorithms usually learn from scratch in each new environment, ignoring information coming from previous environments. However, there is a growing body of work on applying transfer learning to RL to allow agents to reuse prior experiences [10]. Figure 2.1a shows the standard RL view of an agent interacting with its environment. Figure 2.1b represents a common multiagent view of a unified team interacting with the environment where the agents model their teammates as being separate from the environment. In this case, the team is designed before being deployed to cooperate with these specific agents to interact with a fixed environment. However, these agents rely on knowing their teammates and usually require an explicit communication and/or coordination protocol to be shared among the whole team [11, 12]. On the other hand, this book will focus on ad hoc teams drawn from a set of possible teammates, where the team tackles a variety of possible environments as shown in Fig. 2.1c. In this case, the teammates are not programmed

Fig. 2.1 Foci of agent based research. **a** A view of a single agent interacting with its environment used by many reinforcement learning algorithms. **b** A standard view of a unified team interacting with the environment. **c** The ad hoc teamwork setting in which an agent cooperates with an ad hoc team of agents to accomplish shared goals in a given environment where the teammates and the environment are each drawn from diverse sets at the beginning of an episode



to cooperate with this specific ad hoc agent, and they must be treated as fixed and given. Instead, this research focuses on enabling the ad hoc agent to cooperate with a variety of teammates in a range of possible environments.

In order to be responsive to different teammates and environments, a fully general ad hoc agent needs two general classes of capabilities: (1) the ability to learn how to act in an environment to maximize reward, and (2) the ability to reason about teamwork and learn about its teammates. Previous work in reinforcement learning has largely focused on how an agent should learn about the dynamics of the environment [13, 14]. Therefore, this book will leverage such past research about (1) and expand this work in the new direction of (2), reasoning about the team and social knowledge required for effective teamwork.

Ad hoc teamwork problems can be encountered in a variety of real world scenarios. As described in the example above, in search and rescue scenarios, robots from different developers need to cooperate quickly. Furthermore, as more robots enter society, their interactions will increase. In the near future, personal assistant robots may need to interact with other service robots to accomplish their tasks. In addition, the introduction of autonomous cars opens up an interesting area for ad hoc teamwork: cooperating with human drivers. Cars on the road have the shared goal of reaching their destinations quickly and safely, and they need to cooperate with the other cars in order to accomplish these goals. These agents have very limited observations of the other cars, and therefore must adapt quickly.

Another area where ad hoc teamwork comes into play is when robots need to accomplish tasks in workplace settings with human teammates. These settings include manufacturing jobs, where new robots are now able to work more closely with humans, and using robots in warehouses for moving products. The robots are likely to interact with a variety of humans, and therefore need to adapt quickly to these new teammates. While the robots and humans share a common goal, communication between them is limited; humans cannot quickly and fully specify their intentions to the level used in existing multiagent coordination algorithms. Therefore, it is desirable for the robots to reason about ad hoc teamwork.

Another interesting application of ad hoc teamwork is in the area of games. Game-playing agents interact with humans and need to adapt to them with only limited observations. These interactions are incredibly complex, and existing approaches rely heavily on heuristic approaches with only limited adaptations [15–17]. Reasoning about ad hoc teamwork would allow virtual agents in video games to adapt to their human teammates.

2.2 Evaluation Framework

In an ad hoc team, agents need to be able to cooperate with a variety of previously unseen teammates. Rather than developing protocols for coordinating an entire team, ad hoc team research focuses on developing agents that cooperate with teammates in the absence of such explicit protocols. Therefore, we consider a single agent cooperating with teammates that may or may not adapt to its behavior. In this scenario, we can only develop algorithms for the ad hoc team agent, without having any direct control over the other teammates.

However, directly measuring teamwork is difficult. In many cases, the only easily measurable aspect is the overall performance of the team, which makes it difficult to assign credit to each agent. By placing an agent on a variety of teams and measuring those teams' performances, we can estimate how good the agent is at teamwork.

Therefore, we introduce an algorithm that evaluates an ad hoc team agent while considering the teammates and domains it may encounter. This framework is specified in Algorithm 2.1 and visually presented in Fig. 2.2. According to this framework, the performance of the ad hoc team agent a depends on the distribution of problem

Algorithm 2.1 Ad hoc agent evaluation

1: **function** Evaluate:

inputs:
 a

▷ the ad hoc agent

 A

▷ the set of possible teammate agents

 D

▷ the set of possible domains

outputs:
 $\frac{r}{n}$

▷ the average performance (reward)

params:
 s_{min}

▷ the minimal acceptable performance of a team

 n

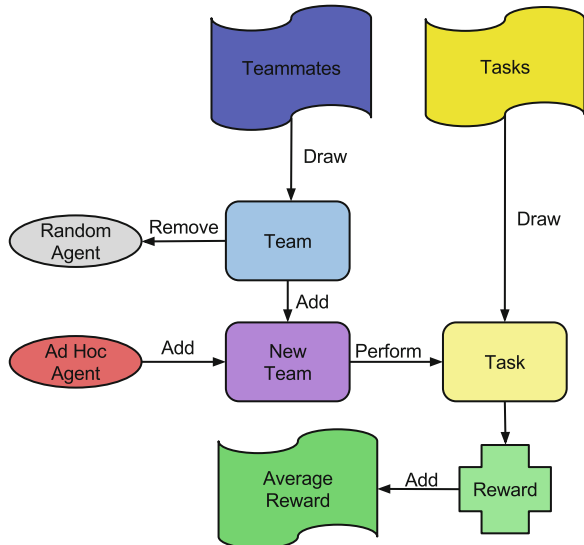
▷ the number of iterations

2: Initialize: $r = 0$ 3: **for** $i = 1$ to n **do**4: Sample a task d from D 5: Randomly draw a subset of agents B , from A such that $E[s(B, d)] \geq s_{min}$ 6: Randomly select one agent $b \in B$ 7: Create the new team $C = \{a\} \cup B \setminus \{b\}$ 8: $r = r + s(C, d)$ 9: **return** $\frac{r}{n}$

10: If $\text{Evaluate}(a_0, A, D) > \text{Evaluate}(a_1, A, D)$ and the difference is significant, we can conclude that a_0 is a better ad hoc team agent than a_1 in domain d over the set of possible teammates A .

domains D and the distribution of possible teammates A that it will cooperate with. For the team B cooperating to execute the task d , $s(B, d)$ is a scalar score representing their effectiveness, where higher scores indicate better performance. The algorithm takes a sampling approach to average the agent's performance across a range of

Fig. 2.2 A visual representation of the evaluation algorithm given in Algorithm 2.1



possible tasks and teammates to capture the idea that a good ad hoc team player ought to be robust to a wide variety of teamwork scenarios. We use s_{min} as a minimum acceptable reward for the team to be evaluated, because the ad hoc team agent may be unable to accomplish a task if its teammates are too ineffective, regardless of its own abilities. It is mainly used to reduce the number of samples required to evaluate the ad hoc agents and reduces the noise in the comparisons. Metrics other than the sum of the rewards can be used depending on the domain, such as the worst-case performance.

2.3 Dimensions of Ad Hoc Team Problems

Section 2.2 specified the framework for evaluating ad hoc team agents, but this evaluation depends on the specific domain and teammates that the ad hoc agent may encounter. This section identifies three dimensions of ad hoc teamwork settings that can be used to describe these domains and teammates. We hypothesize that domains with similar values along these dimensions can be tackled by similar algorithms, while domains with very different values will need different algorithms for good performance. For this book, we use these dimensions as a way as classifying problems, but a promising area for future work is to apply these dimensions to predict which algorithms will be effective on different problems.

There are many possible ways that ad hoc team domains can vary, such as the size of the task's state space and the stochasticity of the domain. But, for differentiating among the algorithms in the existing literature, we find the following three to be the most informative.

1. **Team Knowledge:** Does the ad hoc agent know what its teammates' actions will be for a given state, before interacting with them?
2. **Environment Knowledge:** Does the ad hoc agent know the transition and reward distribution given a joint action and state before interacting with the environment?
3. **Reactivity of teammates:** How much does the ad hoc agent's actions affect those of its teammates?

These dimensions affect the difficulty of planning in the domain in addition to how much an ad hoc agent needs to explore the environment and its teammates. When an ad hoc agent has good knowledge, it can plan without considering exploration, but when it has incomplete knowledge, it must reason about the cost and benefits of exploration. The exploration-exploitation problem has been studied previously, but adding in the need to explore the teammates' behaviors and the ability to affect them considerably alters this tradeoff. Sections 2.3.1–2.3.3 provide further details about each of these dimensions, how they are measured, and why they are important for ad hoc teamwork.

To better illustrate the dimensions, we introduce a simple domain to evaluate across each of the dimensions. The domain is described here, and it will be revisited in the discussion of each dimension (Sections 2.3.1–2.3.3).

MatchActions: *This domain is a typical coordination game with two agents, each of which has two actions. If they select the same action, both receive a reward of r_i , where r_i is randomly selected from $\{0.5, 0.75, 1.0\}$ for $i \in 1, 2$, but fixed for the episode. On the other hand, if both of the agents select different actions, they receive a reward of 0. In addition, both agents can observe their teammates' previous actions. The ad hoc agent knows that its teammate is following one of two behaviors:*

- **FirstAction:** *the teammate always chooses the first action*
- **BestResponse:** *the teammate chooses the same action as the ad hoc agent did previously*

Therefore, the state can be represented as the previous action taken by the ad hoc agent, called s_0 if the ad hoc agent chose the first action, and s_1 otherwise.

2.3.1 Team Knowledge

The ad hoc agent's knowledge about its teammates' behaviors gives insight into the difficulty of planning in the domain. The agent's knowledge can range from knowing the complete behaviors of its teammates to knowing nothing about them. Settings with partial information are especially relevant, because in many real world problems, the exact behavior of a teammate may not be known, but some reasonable guidelines of their behaviors exist. For example, when playing soccer, one can usually assume that a teammate will not intentionally pass to the other team or shoot at the wrong goal. If the behaviors are completely known, the agent can reason fully about the team's actions, while if the behaviors are unknown, the agent must learn about them and adapt to find a good behavior.

To estimate the ad hoc agent's knowledge about its teammates' behaviors, we compare the actions the ad hoc agent expects them to take and the ground truth of what actions they take. Specifically, we compare the expected distribution of teammate actions to the true distribution that the teammates follow. To compute the difference between the distributions, we use the Jensen-Shannon divergence measure, which was chosen because it is a smoothed, symmetric variant of the popular Kullback-Leibler divergence measure. Specifically, we denote the Jensen-Shannon divergence by JS where

$$JS(P, Q) = \frac{1}{2}(KL(P, M) + KL(Q, M))$$

and $M = \frac{1}{2}(P + Q)$. The Kullback-Leibler divergence is given by

$$KL(P, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

When the ad hoc agent has no information about a teammate's action, we assume that it uses the uniform distribution to represent its actions. Therefore, we define the knowledge measure as

$$K(T, P) = \begin{cases} 1 & \text{if } JS(T, P) = 0 \\ 1 - \frac{JS(T, P)}{JS(T, U)} & \text{if } JS(T, P) < JS(T, U) \\ -\frac{JS(P, U)}{JS(U, \text{Point})} & \text{otherwise} \end{cases} \quad (2.1)$$

where T is the true distribution, P is the predicted distribution, U is the uniform distribution, and Point is a distribution with all weight on one point (e.g. $[1, 0, 0, \dots]$). By this definition, $K(T, T) = 1$, so the knowledge is complete if the ad hoc agent knows the true distribution. $K(T, U) = 0$, representing when the ad hoc agent has no knowledge and relies on the uniform distribution. Finally, if the predicted distribution is less accurate than the uniform distribution, then $K(T, P)$ is negative, with a minimum value of -1 . This measure captures the range $[0, 1]$ smoothly, but can still be used for the range $[-1, 0]$.¹ However, we generally expect the prediction to be a higher entropy distribution than the true distribution as the ad hoc agent ought to correctly model its uncertainty in its teammates' behaviors rather than being confident and wrong, which keeps the measure in the range $[0, 1]$.

We define the ad hoc agent's knowledge about its teammates' behaviors as the average over the teammates and world states, specifically

$$\text{TeamK} = \frac{\sum_{s=1}^n \sum_{t=1}^k K(\text{TrueAction}_t(s), \text{PredAction}_t(s))}{nk}$$

where $1 \leq s \leq n$ is the state, $1 \leq t \leq k$ specifies a teammate, $\text{TrueAction}_t(s)$ is the ground truth action distribution for teammate t for state s , and $\text{PredAction}_t(s)$ is the action distribution that the ad hoc agent predicts teammate t to select for state s . Thus, if the ad hoc agent has better information about its teammates' behaviors, the distance between the distributions will be smaller and TeamK will be higher.

Let us now calculate the TeamK for the MatchActions domain. The ad hoc agent has uniform beliefs over its teammate following either the FirstAction or BestResponse behaviors. However, the teammate is actually following the BestResponse behavior. With these beliefs, in s_0 , the ad hoc agent expects that its teammate will always chose a_0 , so $\text{PredAction}_{s_0} = [1, 0]$. In s_1 , the ad hoc agent thinks that the teammate will choose a_0 with probability 0.5 and a_1 with probability 0.5, while it actually chooses a_1 with probability 1. Thus,

¹One slight anomaly of this measure is that when T is the uniform distribution (e.g. $[0.5, 0.5]$), K is either 1 when P is exactly correct at $[0.5, 0.5]$ or negative. For all other values of T , K smoothly spans the range $[-1, 1]$.

$$\text{TeamK} = \frac{K([1, 0], [1, 0]) + K([0, 1], [\frac{1}{2}, \frac{1}{2}])}{2} = \frac{0 + 1}{2} = 0.5$$

This value indicates that the ad hoc agent is somewhat knowledgeable about its teammate's actions as it predicts its teammate's actions half the time better than random guessing.

2.3.2 Environmental Knowledge

Another informative dimension is how much knowledge the ad hoc agent has about the effects of a joint action given a state, for example the transition and reward functions. If the ad hoc agent has complete knowledge about the environment, it can plan about what actions it should select more simply than if it must also consider unknown effects of actions. However, if it has incomplete knowledge, it must explore its actions and face the standard problem of balancing exploring the environment versus exploiting its current knowledge.

Similarly to teammate knowledge, we formally define the ad hoc agent's knowledge about the environment's transitions as

$$\text{TransK} = \frac{1}{nm} \sum_{s=1}^n \sum_{j=1}^m K(\text{TrueTrans}(s, j), \text{PredTrans}(s, j))$$

where $1 \leq s \leq n$ is the state, $1 \leq j \leq m$ is a joint action, K is taken from Eq. (2.1), $\text{TrueTrans}(s, j)$ is the ground truth transition distribution from state s given joint action j , and PredTrans is the ad hoc agent's expected transition distribution. If the agent has no information about the transitions, we assume that $\text{PredTrans}(s, j)$ is the uniform distribution. Intuitively, if the ad hoc agent knows more about the transition function, then the distance between TrueTrans and PredTrans will be smaller and as a result TransK will be higher. We define the agent's knowledge about the environmental rewards similarly

$$\text{RewardK} = \frac{1}{nm} \sum_{s=1}^n \sum_{j=1}^m K(\text{TrueReward}(s, j), \text{PredReward}(s, j))$$

We define the environmental knowledge as a 2-dimensional value given by $\text{EnvK} = (\text{TransK}, \text{RewardK})$.

Revisiting the MatchActions domain, the ad hoc agent knows the true transition function, as it only depends on the ad hoc agent's previous action, so $\text{TransK} = 1$. However, it only knows that the payoff for each action is uniformly drawn from $\{0.5, 0.75, 1.0\}$ and the reward is 0 if the agents' actions do not match. There are 8 possible cases to count over, coming from 2 states, 2 actions for the ad hoc agent, and 2 for its teammate, but the cases fall into 2 sets based on whether the actions

match, each set covering 4 cases. In addition, it does not matter which value each matched action actually takes, so we can simplify the calculation. If the agents take the different actions, the reward is correctly known to be 0. Note that there are four reward values possible: $\{0, 0.5, 0.75, 1.0\}$. Therefore, the knowledge in this case is $K([1, 0, 0, 0], [1, 0, 0, 0]) = 1$. On the other hand, if they take the same actions, the ad hoc agent is unsure which of the three rewards $\{0.5, 0.75, 1.0\}$ it will receive, so the knowledge in this case is $K([0, 1, 0, 0], [0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}]) = 0.164$. This leads to

$$\begin{aligned} \text{RewardK} &= \frac{4 * K([1, 0, 0, 0], [1, 0, 0, 0]) + 4 * K([0, 1, 0, 0], [0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}])}{8} \\ &= \frac{4 * 1 + 4 * 0.164}{8} \\ &= 0.582 \end{aligned}$$

Thus, $\text{EnvK} = (1, 0.582)$. As the agent observes these payoffs, it can refine its knowledge, but we are evaluating these properties prior to the ad hoc agent interacting with its environment.

2.3.3 Teammate Reactivity

The optimal behavior for the ad hoc agent also depends on how much its teammates react to its actions. If its teammates' actions do not depend on the ad hoc agent at all, the ad hoc agent can simply choose its actions to maximize the team reward, as if it were a single agent problem. Considering the actions of its teammates separately from that of the environment may still help computation by factoring the domain. However, if the teammates' actions depend strongly on the ad hoc agent's actions, the ad hoc agent's reasoning should consider what its teammates' reactions will be. If the ad hoc agent is modeling its teammates and its teammates are modeling the ad hoc agent, the problem can become recursive, as is directly addressed by Vidal and Durfee's Recursive Modeling Method [18].

A formal measure of the teammate reactivity needs to capture how different the teammates' actions will be when the ad hoc agent chooses different actions. We measure the distance between the resulting distributions of the teammate joint actions, using the pairwise Jensen-Shannon divergence measures. However, it is desirable for the distance to be 1 when the distributions have no overlap, so we use a normalizing constant of $\log 2$. Thus, we define the *reactivity* of a domain in state s as

$$\text{Reactivity}(s) = \frac{1}{m(m-1) \log 2} \sum_{a=1}^m \sum_{a'=1}^m \text{JS}(T(s, a), T(s, a'))$$

where JS is the Jensen-Shannon divergence measure, $1 \leq a, a' \leq m$ is the actions available to the ad hoc agent, and $T(s, a)$ is the distribution of the teammates' joint

actions given the state s and the ad hoc agent's action, a . We use $m - 1$ in the denominator because we exclude the case where $a = a'$; in the numerator, the JS measure will be 0 in this case. For the overall reactivity of the domain, we average over the states, resulting in $\text{Reactivity} = \frac{1}{n} \sum_{s=1}^n \text{Reactivity}(s)$. It is possible to consider how an action affects the teammates' actions further in the future, but we restrict our focus to one step reactivity for this book. Note that all of the sums in this formulation can be converted to integrals for continuous states or actions. This formulation is similar to the empowerment measure used by Jung et al. [19], but we consider the ad hoc agent's ability to change the actions of its teammates rather than the environment state.

Let us once again explore this dimension in the context of the MatchActions domain. Although the ad hoc agent is unsure of its teammate's behavior, the teammate is truly playing the BestResponse behavior. Thus, its actions are entirely dependent on the ad hoc agent's actions, so $\text{Reactivity} = 1$. If instead the teammate played BestResponse with probability $\frac{9}{10}$ and FirstAction with probability $\frac{1}{10}$, then we would get

$$\text{Reactivity} = \frac{\text{JS}([1, 0], [\frac{1}{10}, \frac{9}{10}]) + \text{JS}([\frac{1}{10}, \frac{9}{10}], [1, 0])}{2 \log 2} = 0.758$$

Therefore, we can conclude that the agent would still be very reactive, though not as reactive as the BestResponse agent.

2.3.4 Applying the Dimensions

In theory, calculating the dimensions over every possible state is a promising approach. However, as the size of the state space grows, this approach rapidly becomes computationally ineffective. Therefore, it is desirable to approximate the values along each dimension. Specifically, we approximate these values by randomly sampling states and teammates and summing over these samples to calculate approximate values for each of the dimensions. In addition, in continuous state spaces, the summations in the dimension definitions become integrals in the continuous case, but we continue to sample states in these scenarios. Furthermore, the distributions become continuous, but the JS measure can operate over continuous distributions. Specifically, we approximate the JS measure using Monte Carlo sampling in this book. The domains used in this book are described in Sect. 3.2, where we discuss the values of each domain along these dimensions.

2.4 Chapter Summary

This chapter introduces the type of situations this book focuses on: *ad hoc team* problems. In ad hoc teams, agents must adapt to new and unknown teammates without prior coordination, possibly without any explicit communication channels. In addition, this chapter introduces the evaluation framework used to evaluate ad hoc agents. This evaluation framework relies on sampling teams and tasks and then replacing an agent on the team with the ad hoc agent. The resulting team performs the task and receives a reward based on its performance, which is combined with results with other teams and tasks. Finally, this chapter describes 3 dimensions for categorizing ad hoc team problems that indicate which approaches are expected to be effective. These dimensions are: (1) team knowledge, (2) environment knowledge, and (3) team reactivity. This chapter provides the framework for how the rest of the book investigates ad hoc teamwork scenarios. The next chapter will provide an introduction to the algorithms that this book builds upon as well as a description of the domains used to evaluate the proposed algorithm.

References

1. Barrett, Samuel, and Peter Stone. 2012. An analysis framework for ad hoc teamwork tasks. In *Proceedings of the eleventh international conference on autonomous agents and multiagent systems (AAMAS)*, June 2012.
2. Huang, Ya-Wen, Y. Sasaki, Y. Harakawa, E.F. Fukushima, and S. Hirose. 2011. Operation of underwater rescue Robot anchor diver III during the 2011 Tohoku earthquake and tsunami. In *OCEANS 2011*, Sept 2011, 1–6.
3. Murphy, R.R., K.L. Dreger, S. Newsome, J. Rodocker, E. Steimle, T. Kimura, K. Makabe, F. Matsuno, S. Tadokoro, and K. Kon. 2011. Use of remotely operated marine vehicles at Minamisanriku and Rikuzentakata Japan for disaster recovery. In *2011 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, Nov 2011, 19–25.
4. Nagatani, K., S. Kiribayashi, Y. Okada, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, and Y. Hada. 2011. Redesign of rescue mobile Robot Quince. In *2011 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, Nov 2011, 13–18.
5. Richardson, D.K. 2011. Robots to the rescue? *Engineering Technology* 6(4): 52–54.
6. Decker, Keith S., and Victor R. Lesser. Designing a family of coordination algorithms. In *International conference on multi-agent systems (ICMAS)*, June 1995, 73–80.
7. Grosz, B., and S. Kraus. 1996. Collaborative plans for complex group actions. *Artificial Intelligence (AIJ)* 86: 269–368.
8. Stone, Peter, and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8(3): 345–383.
9. Tambe, Milind. 1997. Towards flexible teamwork. *Journal of Artificial Intelligence Research (JAIR)* 7: 81–124.
10. Taylor, Matthew E., and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research (JMLR)* 10(1): 1633–1685.
11. Lauer, Martin, and Martin Riedmiller. 2000. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the seventeenth international conference on machine learning (ICML)*. Morgan Kaufmann, 535–542.

12. Xuan, Ping, Victor Lesser, and Shlomo Zilberstein. 2001. Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the fifth international conference on autonomous agents (AGENTS)*.
13. Kalyanakrishnan, Shivaram, and Peter Stone. 2011. Characterizing reinforcement learning methods through parameterized learning problems. *Machine Learning (MLJ)* 84(1–2): 205–247.
14. Sutton, Richard S., and Andrew G. Barto. 1998. *Reinforcement learning: An introduction*. Cambridge: MIT Press.
15. Millington, Ian, and John Funge. 2009. *Artificial intelligence for games*. 2nd ed. San Francisco: Morgan Kaufmann Publishers Inc.
16. Smed, Jouni, and Harri Hakonen. 2006. *Algorithms and networking for computer games*. Wiley.
17. Buckland, Mat, and Mark Collins. 2002. *AI techniques for game programming*. Premier Press.
18. Vidal, Jose M., and Edmund H. Durfee. 1995. Recursive agent modeling using limited rationality. In *International conference on multi-agent systems (ICMAS)*.
19. Jung, T., D. Polani, and P. Stone. 2010. Empowerment for continuous agent-environment systems. Technical Report AI-10-03, The University of Texas at Austin Computer Science Department.

<http://www.springer.com/978-3-319-18068-7>

Making Friends on the Fly: Advances in Ad Hoc
Teamwork

Barrett, S.

2015, XX, 144 p. 26 illus., 7 illus. in color., Hardcover

ISBN: 978-3-319-18068-7