To appear in *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (IJCAI 2007), January 2007.

Learning and Multiagent Reasoning for Autonomous Agents

IJCAI-07 Computers and Thought Paper

Peter Stone
Department of Computer Sciences
The University of Texas at Austin
pstone@cs.utexas.edu

Abstract

One goal of Artificial Intelligence is to enable the creation of robust, fully autonomous agents that can coexist with us in the real world. Such agents will need to be able to learn, both in order to correct and circumvent their inevitable imperfections, and to keep up with a dynamically changing world. They will also need to be able to interact with one another, whether they share common goals, they pursue independent goals, or their goals are in direct conflict. This paper presents current research directions in machine learning, multiagent reasoning, and robotics, and advocates their unification within concrete application domains. Ideally, new theoretical results in each separate area will inform practical implementations while innovations from concrete multiagent applications will drive new theoretical pursuits, and together these synergistic research approaches will lead us towards the goal of fully autonomous agents.

1 Introduction

Much of the past research in Artificial Intelligence has focused either on high-level reasoning from abstract, ungrounded representations or on interpreting raw sensor data towards building grounded representations. However, neither of these foci taken alone is sufficient for deploying practical, real-world AI systems. In recent years, an active contingent within the field has focused on creating complete autonomous agents: those that sense their environment, engage in high-level cognitive decision-making, and then execute their actions in the environment.

As the field progresses in this direction, individual autonomous agents, either in software or physically embodied, are becoming more and more capable and prevalent. Multiagent systems consisting of homogeneous or similar agents are also becoming better understood. However, to successfully interact in the real world, agents must be able to reason about their interactions with heterogeneous agents of widely varying properties and capabilities.

Some of these other agents will have similar or identical goals, in which case there may be some advantage to explicitly coordinating their activities. On the other hand, some of the other agents will have opposing or uncorrelated goals. In both cases, when dealing with other agents, it is beneficial from the points of view of robustness and flexibility for an agent to be able to learn or adapt how it interacts with them.

As such, learning, interaction, and their combination are key necessary capabilities on our path towards creating robust autonomous agents. To enable these capabilities, we need to combine 1) basic, algorithmic research on machine learning and multiagent systems with 2) application-oriented research threads aimed at studying complete agents in specific, complex environments, with the ultimate goal of drawing general lessons from the specific implementations.

This paper illustrates this research methodology, drawing heavily on concrete examples from my own research, and suggests research directions for the future aimed at creating fully autonomous agents, including robots, capable of learning and interacting. It represents my own perception of the important and interesting research topics in the field as of 2007. It is not intended to be comprehensive – there are certainly other AI areas that are active, interesting, and likely to be fruitful. My ambition for the paper is that it will inspire some of my current and future colleagues to take an interest in the problems presented here, and perhaps that it will inspire others to formulate opposing arguments that entirely different topics are more worthy of pursuit than those advocated here.

The remainder of this paper is organized as follows. First, in Sections 2 and 3, learning and multiagent reasoning are considered separately. Next, in Section 4, related issues on the path towards robust *physical* agents (robots) that can learn and interact are addressed.

Although Sections 2–4 treat agent components, the essence of creating *complete* autonomous agents is putting all the pieces together. Indeed, Daphne Koller organized her 2001 Computers and Thought lecture around the notion of three conceptual bridges connecting representation, reasoning, and learning as a way of mitigating her observation that:

> In AI, as in many communities, we have the tendency to divide a problem into well-defined pieces, and make progress on each one. But as we make progress, the problems tend to move away from each other. [Koller, 2001]

In Section 5, I argue that creating fully functional agents in complex application environments is another excellent way to combat such fragmentation. Finally, Section 6 concludes.

2 Learning

Machine Learning has been an active area within AI for many years. Indeed Tom Mitchell's Computers and Thought paper more than two decades ago refers to "a resurgence of interest in machine learning" due to advances in general problem solving and knowledge-based expert systems [Mitchell, 1983]. Since then, supervised learning methods, for both classification and regression, have matured to the point of enabling a general purpose toolkit that can be used productively by experts and novices alike [Witten and Frank, 1999]. Similarly, unsupervised learning algorithms for data clustering have advanced nicely. But from the point of view of autonomous agents, it is the relatively recent development of reinforcement learning (RL) algorithms, designed to learn action selection from delayed reward in sequential decision making problems, that is most significant.[1]

Unlike classical supervised and unsupervised learning where the learner must be supplied with training data so that it can learn a function from features to the target set, the premise of RL matches the agent paradigm exactly: the learner gathers its own training data by interacting with the environment so that it can learn a policy mapping states to actions. An RL agent repeatedly takes actions that both move it to a new state in its environment and lead to some immediate reward signal. The learner must explicitly tradeoff between exploration and exploitation in an effort to maximize the long-term reward that it will receive.

One common approach to reinforcement learning relies on the concept of *value functions*, which indicate, for a particular policy, the long-term value of a given state or state-action pair. *Temporal difference* (TD) methods [Sutton, 1988], which combine principles of dynamic programming with statistical sampling, use the immediate rewards received by the agent to incrementally improve both the agent's policy and the estimated value function for that policy. Hence, TD methods enable an agent to learn during its "lifetime" from its individual experience interacting with the environment.[2]

For small problems, the value function can be represented as a table. However, the large, probabilistic domains which arise in the real world usually require coupling TD methods with a *function approximator*, which represents the mapping from state-action pairs to values via a more concise, parameterized function and uses supervised learning methods to set its parameters. Many different methods of function approximation have been used successfully, including CMACs, radial basis functions, and neural networks [Sutton and Barto, 1998]. However, using function approximators requires making crucial representational decisions (e.g. the number of hidden units and initial weights of a neural network). Poor design choices can result in estimates that diverge from the optimal value function [Baird, 1995] and agents that perform poorly. Even for reinforcement learning algorithms with guaranteed convergence [Baird and Moore, 1999; Lagoudakis and Parr, 2003], achieving high performance in practice requires finding an appropriate representation for the function approximator. As Lagoudakis and Parr observe,

> The crucial factor for a successful approximate algorithm is the choice of the parametric approximation architecture(s) and the choice of the projection (parameter adjustment) method. [Lagoudakis and Parr, 2003]

Nonetheless, representational choices are typically made manually, based only on the designer's intuition.

Despite an early emphasis in the field on "tabula rasa" learning, it is becoming increasingly accepted that the process of designing agents for complex, real-world domains will always require some manual input of this sort. The key to success will be limiting the requirements for human knowledge to tasks and representations for which humans have good intuitions. As referred to above, the existence of a general purpose toolkit that can be used without expert knowledge of the underlying algorithms suggests that supervised learning methods are mature enough for use in robust, complex systems. That is not yet the case for RL. Despite the elegant theory that accompanies TD methods, most notably that Q-learning converges to an optimal value function if every state is visited infinitely often [Watkins, 1989], and despite a limited number of successes that have been reported in large-scale domains [Tesauro, 1994; Crites and Barto, 1996; Stone *et al.*, 2005], crucial, somewhat unintuitive decisions about representations need to be made based on a deep understanding of the underlying algorithms and application domain.

The remainder of this section outlines directions for research that will help current reinforcement learning, and other machine learning, methods scale up to the point that they can become core components of fully autonomous agents in real-world tasks. Sections 2.1 and 2.2 deal with automatically adjusting knowledge representations used for learning. Sections 2.3 and 2.4 address ways in which humans can provide intuitive knowledge to learning agents, either by providing a subtask decomposition or by suggesting related tasks for knowledge transfer.

2.1 Adaptive Function Approximation

In order to address the observation of Lagoudakis and Parr above, Whiteson and Stone [2006] automate the search for effective function approximators for RL agents by applying optimization techniques to the representation problem. In that research, we propose using evolutionary methods [Goldberg, 1989] for optimizing the representation because of their demonstrated ability to discover effective representations [Gruau *et al.*, 1996; Stanley and Miikkulainen, 2002]. Synthesizing evolutionary and TD methods results in a new approach called *evolutionary function approximation*, which automatically selects function approximator representations that enable efficient individual learning. Thus, this method

[1] This is not to say that classical learning is irrelevant to agents. An autonomous agent may well incorporate learned predictions as a part of its world model. For example an autonomous bidding agent may use regression to predict the future closing price of an auction, and then use this information to evaluate its potential bids [Stone *et al.*, 2003].

[2] TD methods contrast with policy search methods which learn based on the results of holistic policy evaluations rather than from individual action effects.

evolves individuals that are better able to *learn*. This biologically intuitive combination has been applied to computational systems in the past [Hinton and Nowlan, 1987; Ackley and Littman, 1991; Boers *et al.*, 1995; French and Messinger, 1994; Gruau and Whitley, 1993; Nolfi *et al.*, 1994] but never, to our knowledge, to aid the discovery of good TD function approximators.

Specifically, we use NeuroEvolution of Augmenting Topologies (NEAT) [Stanley and Miikkulainen, 2002] to select neural network function approximators for Q-learning [Watkins, 1989], a popular TD method. The resulting algorithm, called NEAT+Q, uses NEAT to evolve topologies and initial weights of neural networks that are better able to learn, via backpropagation, to represent the value estimates provided by Q-learning.

In experimental evaluation, we test Q-learning with a series of manually designed neural networks and compare the results to NEAT+Q and regular NEAT, which learns direct representations of policies. The results demonstrate that evolutionary function approximation can significantly improve the performance of TD methods, thus providing a much-needed practical approach to selecting TD function approximators, automating a critical design step that is typically performed manually.

Though that research takes a step towards automating the choice of representations for learning, there is still much room for future work, including extending it to use different policy search methods such as PEGASUS [Ng and Russell, 2000] and policy gradient methods [Sutton *et al.*, 2000]; and perhaps more importantly, optimizing function approximators other than neural networks, such as CMACs and radial basis functions. Research on feature selection to adapt the inputs to the representation [Fawcett, 1993; Whiteson *et al.*, 2005] is also in this space of adaptive representations.

2.2 Learned Abstractions

Another approach to adjusting problem representation is *state abstraction*, which maps two distinct ground states to a single abstract state if an agent should treat the ground states in exactly the same way. The agent can still learn optimal behavior if the environment satisfies a particular condition, namely that each action has the same abstract outcome (next state and reward) for all primitive states that are mapped to the same abstract state: ground states that are grouped together must share the same local behavior in the abstract state space [Dean and Givan, 1997; Ravindran and Barto, 2003]. However, this cited research only applies in a planning context, in which the domain model is given, or if the user manually determines that the condition holds and supplies the corresponding state abstraction to the RL algorithm.

Jong and Stone [2005] propose an alternative condition for state abstraction that is more conducive to automatic discovery. Intuitively, if an agent can behave optimally while ignoring a certain variable of the state representation, then it might be able to learn successfully while ignoring that state variable. Recognizing that discovering such qualitative structure tends to require more time than learning an optimal behavior policy [Thrun and Schwartz, 1995], this approach suggests a knowledge-transfer framework, in which we analyze learned policies in one environment to discover abstractions that might improve learning in similar environments.

We introduce an efficient algorithm that discovers local regions of the state space in which the same action would be optimal regardless of the value of a given set of state variables. For example, you would take the same route to work in New York no matter what the weather is in London. This algorithm depends on determining what set of actions is optimal in each state; we give two statistical tests for this criterion that trade off computational and sample complexity. We then generalize this learned structure to a similar environment, where an agent can ignore each set of state variables in the corresponding learned region of the state space.

We must take care when we apply our discovered abstractions, since the criteria we use in discovery are strictly weaker than those given in other work on safe state abstraction [Li *et al.*, 2006]. Transferring abstractions from one domain to another may also introduce generalization error. To preserve convergence to an optimal policy, we encapsulate our state abstractions in *temporal abstractions*, or *options*, which construe sequences of primitive actions as constituting a single abstract action [Sutton *et al.*, 1999]. In contrast to previous work with temporal abstraction, we discover abstract actions intended just to simplify the state representation, not to achieve a certain goal state. RL agents equipped with these abstract actions thus learn when to apply state abstraction the same way they learn when to execute any other action.

The fact that abstractions are the building blocks for hierarchical RL suggests the recursive application of our abstraction discovery technique to create hierarchies of temporal abstractions that explicitly facilitate state abstractions, as in MAXQ task decompositions [Dietterich, 2000]. This possibility highlights the need for robust testing of optimal actions, since each application of our method adds new potentially optimal actions to the agent. Learning MAXQ hierarchies based on our method is a natural direction for future research.

2.3 Layered Learning

Hierarchies in general are powerful tools for decomposing complex control tasks into manageable subtasks. As a case in point, mammalian biology is a composition of hierarchically organized components, each able to perform specialized subtasks. These components span many levels of behavior ranging from individual cells to complex organs, culminating in the complete organism. Even at the purely behavioral level, organisms have distinct subsystems, including reflexes, the visual system, etc. It is difficult to imagine a monolithic entity that would be capable of the range and complexity of behaviors that mammals exhibit.

As covered in the previous section, initial steps have been made towards learning hierarchies automatically in relatively simple domains. However, in complex tasks, the hierarchies still need to be defined manually [Brooks, 1986; Gat, 1998]. Layered learning [Stone and Veloso, 1998; 2000a] is a hierarchical paradigm that similarly requires a given task decomposition, but that then relies on *learning* the various subtasks necessary for achieving the complete high-level goal. Layered learning is a bottom-up paradigm by which low-level behaviors (those closer to the environmental inputs) are trained prior to high-level behaviors.

The layered learning approach is somewhat reminiscent of Rodney Brooks' subsumption architecture as summarized in his Computers and Thought paper [Brooks, 1991]. The subsumption architecture layers control modules, allowing high-level controllers to override lower-lever ones. Each control level is capable of controlling a robot on its own up to a specified level of functionality. In order to focus on learning and to move quickly to high-level behaviors, layered learning abandons the commitment to have every layer be completely able to control a robot. Instead, many situation-specific (but as general as possible) behaviors are learned which are then managed by higher-level behaviors. Nevertheless, the idea of building higher levels of functionality on top of lower levels is retained.

Table 1 summarizes the principles of the layered learning paradigm which are described in detail in this section.

1. A mapping directly from inputs to outputs is not tractably learnable.
2. A hierarchical task decomposition is given.
3. Machine learning exploits data to train and/or adapt. Learning occurs separately at each level.
4. The output of learning in one layer feeds into the next layer.

Table 1: The key principles of layered learning.

Principle 1

Layered learning is designed for domains that are too complex for learning a mapping directly from the input to the output representation. Instead, the layered learning approach consists of breaking a problem down into several task layers. At each layer, a concept needs to be acquired. A machine learning (ML) algorithm abstracts and solves the local concept-learning task.

Principle 2

Layered learning uses a bottom-up incremental approach to hierarchical task decomposition. Starting with low-level subtasks, the process of creating new ML subtasks continues until the high-level tasks, that deal with the full domain complexity, are reached. The appropriate learning granularity and subtasks to be learned are determined as a function of the specific domain. The task decomposition in layered learning is not automated. Instead, the layers are defined by the ML opportunities in the domain.

Principle 3

Machine learning is used as a central part of layered learning to exploit data in order to *train* and/or *adapt* the overall system. ML is useful for training functions that are difficult to fine-tune manually. It is useful for adaptation when the task details are not completely known in advance or when they may change dynamically. Like the task decomposition itself, the choice of machine learning method depends on the subtask.

Principle 4

The key defining characteristic of layered learning is that each learned layer directly affects the learning at the next layer. A learned subtask can affect the subsequent layer by:

- constructing the set of training examples;
- providing the features used for learning; and/or
- pruning the output set.

Layered learning was originally applied in a complex, multi-agent learning task, namely simulated robot soccer in the RoboCup soccer server [Noda *et al.*, 1998]. An extension that allows for concurrent training of multiple layers was also implemented in simulation [Whiteson and Stone, 2003]. As described below in Section 4.1, layered learning has recently been applied successfully on physical robots. In all these cases, the subtask decomposition is supplied manually and if relatively intuitive to construct. Nonetheless, discovering ways to automate this decomposition, perhaps by leveraging the abstraction discovery work described in Section 2.2, is an important future goal.

2.4 Transfer Learning

A particularly topical area of AI research in 2007 is transfer learning: leveraging learned knowledge on a *source task* to improve learning on a related, but different, *target task*. Transfer learning can pertain to classical learning, but it is particularly appropriate for learning agents that are meant to persist over time, changing flexibly among tasks and environments. Rather than having to learn each new task from scratch, the goal is to enable an agent to take advantage of its past experience to speed up new learning.

For a reinforcement learning agent, there are several ways in which the source and target may differ in a transfer problem. For example, source and target tasks with the following differences have been studied in the literature:

- Transition function: Effects of agents' actions differ [Selfridge *et al.*, 1985]
- Reward structure: Agents have different goals [Singh, 1992]
- State space: Agents' environments differ [Konidaris and Barto, 2006; Fernandez and Veloso, 2006]
- Initial state: Agents start in different locations over time [Asada *et al.*, 1994]
- Actions: Agents have different available actions [Maclin *et al.*, 2005; Taylor *et al.*, 2005; Soni and Singh, 2006]
- State variables: Agents' state descriptions differ [Maclin *et al.*, 2005; Taylor *et al.*, 2005; Soni and Singh, 2006]

In the most general case, the source and target can differ in all of these ways. For such cases, Taylor *et al.* [2005] introduce a method for transferring the learned value function in a source RL task to seed learning in the target. The key technical challenge is mapping a value function in one representation to a meaningful value function in another, typically larger, representation, despite the fact that state-action values are inherently task-specific.

Past research confirms that if two tasks are closely related the learned policy from one task can be used to provide a good initial policy for the second task. For example, Selfridge *et al.* [1985] showed that the 1-D pole balancing task could be made harder over time by shortening the length of the pole and increasing its mass; when the learner was first trained on a longer and lighter pole it could more quickly learn to succeed in the more difficult task with the modified transition function. In this way, the learner is able to refine an initial policy for a given task.

We consider the more general case where tasks are related but distinct in that their state and/or action spaces differ. To use the source policy π_S as the initial policy for a TD learner in the target task, we must transform the value function so that it can be directly applied to the new state and action space. We introduce the notion of a behavior transfer functional $\rho(\pi_S) = \pi_T$ that will allow us to apply a policy in the target task. The policy transform functional, ρ, needs to modify the source policy and its associated value function so that it accepts the states in the target task as inputs and allows for the actions in the target task to be outputs, as depicted in Figure 1.

Defining ρ to do this modification in such a way that π_T is a good starting point for learning in the target is the key technical challenge to enable general behavior transfer. Current results indicate that such ρ's do exist, at least for some tasks [Taylor *et al.*, 2005]. However, automating the discovery of the inter-task mapping between the state variables and actions in the source and target tasks remains an open challenge, as does automatically selecting the source and target tasks themselves.

3 Multiagent Reasoning

In addition to learning, a second essential capability of robust, fully autonomous agents is the ability to interact with other agents: multiagent reasoning. As argued in the introduction, to successfully interact in the real world, agents must be able to reason about their interactions with heterogeneous agents of widely varying properties and capabilities. Once we have a single complete agent that is able to operate autonomously for extended periods of time in the real world, it is inevitable that soon after we will have many. And they will need to interact with one another.

Though more recent to our field than Machine Learning, in the past decade Multiagent Systems (MAS) has begun to come to the forefront. As with any methodology, two important questions about MAS are:

- What advantages does it offer over the alternatives? and
- In what circumstances is it useful?

It would be foolish to claim that MAS should be used when designing all complex systems. Like any approach, there are some situations for which it is particularly appropriate, and others for which it is not. In a survey of the field, Stone and Veloso [2000b] summarized the circumstances in which MAS

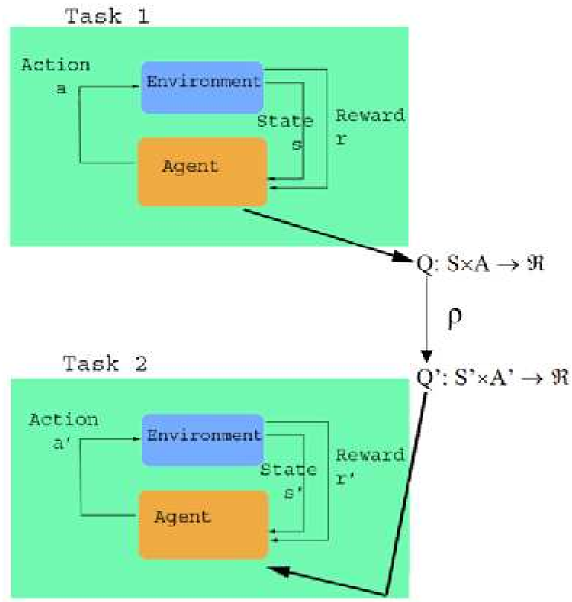

Figure 1: ρ is a functional which transforms a value function Q from one task so that it is applicable in a second task that has different state and action spaces.

is appropriate.[3] For example, some domains *require* MAS, such as those in which agents represent people or organizations with different (possibly conflicting) goals and proprietary information; having multiple agents may speed up computation via parallelization; MAS can provide robustness via redundancy; MAS may be more scalable as a result of modularity; and they can be useful for their elucidation of fundamental problems in the social sciences and life sciences [Cao *et al.*, 1997], including intelligence itself [Decker, 1987]. Regarding this last point, as Weiß [1996] put it: "Intelligence is deeply and inevitably coupled with interaction." In fact, it has been proposed that the best way to develop intelligent machines might be to start by creating "social" machines [Dautenhahn, 1995]. This theory is based on the socio-biological theory that primate intelligence first evolved because of the need to deal with social interactions [Minsky, 1988].

Multiagent systems differ from single-agent systems in that several agents exist that model each other's goals and actions. In the fully general multiagent scenario, there may be direct interaction among agents (communication). Although this interaction could be viewed as environmental stimuli, we present inter-agent communication as being separate from the environment.

From an individual agent's perspective, multiagent systems differ from single-agent systems most significantly in that the environment's dynamics can be affected by other agents. In addition to the uncertainty that may be inherent in the domain, other agents intentionally affect the environment in unpredictable ways. Thus, all multiagent systems can be viewed as having dynamic environments. Figure 2 illustrates the view

[3]Other takes on the same issue appear elsewhere [Bond and Gasser, 1988; Sycara, 1998].

that each agent is both part of the environment and modeled as a separate entity. There may be any number of agents, with different degrees of heterogeneity and with or without the ability to communicate directly.

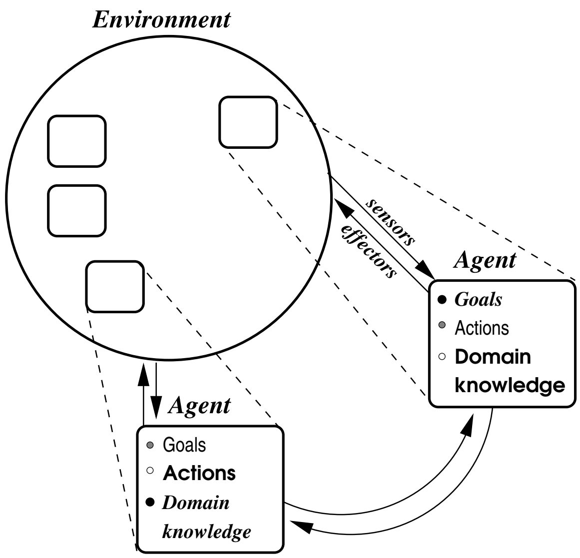

Figure 2: The fully general multiagent scenario. Agents model each other's goals, actions, and domain knowledge, which may differ as indicated by the different fonts. They may also interact directly (communicate) as indicated by the arrows between the agents.

Most MAS research assumes that the protocol for interaction among agents is fixed from the outset. Perhaps they will interact via a well-defined communication language as represented by the arrows between the agents in Figure 2, or perhaps they will interact just through observation of one another's actions as is often the case in adversarial domains. However robust agents should also be able to cope with flexible forms of interaction. In Section 2, the focus was on research areas of Machine Learning that pertain to changing the representation of learned knowledge (e.g. function approximators, abstractions, and inter-task mappings for transfer) more so than the learning algorithms themselves. An analog in MAS is considering that the patterns and very rules of interaction among agents may be able to change over time. This section considers cases in which the rules of interaction can change (Sections 3.1 and 3.2) as well as cases in which the behaviors or capabilities of other agents change or are not known in advance so that adaptive agent modeling is useful or necessary (Section 3.3).

3.1 Adaptive Mechanism Design

The first such case we consider is adaptive mechanism design, which lies in the space considered by Tuomas Sandholm's Computers and Thought lecture, namely agent-based electronic commerce [Sandholm, 2003]. This area falls at the intersection of Computer Science and Economics.

Recent years have seen the emergence of numerous auction platforms that cater to a variety of markets such as business-to-business procurement and consumer-to-consumer transactions. Many different types of auction *mechanisms* defining the rules of exchange may be used for such purposes. Varying parameters of the auction mechanism, such as auctioneer fees, minimum bid increments, and reserve prices, can lead to widely differing results depending on factors such as bidder strategies and product types. *Mechanism design* is the challenge of optimizing auction parameters so as to maximize an objective function, such as auctioneer revenue.

Mechanism design has traditionally been largely an analytic process. Assumptions such as full rationality are made about bidders, and the resulting properties of the mechanism are analyzed in this context [Parkes, 2001]. Even in large-scale real-world auction settings such as the FCC Spectrum auctions, game theorists have convened prior to the auction to determine the best mechanism to satisfy a set of objectives. Historically, this process has been incremental, requiring several live iterations to iron out wrinkles, and the results have been mixed [Cramton, 1997; Weber, 1997]. An important component of this incremental design process involves reevaluating the assumptions made about bidders in light of auction outcomes. These assumptions pertain to bidders' intrinsic properties and to the manner by which these properties are manifested in bidding strategies.

Pardoe *et al.* [2006] address this problem by considering an *adaptive mechanism* that changes in response to observed bidder behavior through the use of a learning algorithm. Our view of adaptive mechanism design is illustrated in Figure 3. A parameterized mechanism is defined such that the seller can use an adaptive algorithm to revise parameters in response to observed results of previous auctions, choosing the most promising parameters to be used in future auctions. Upon execution, the parameterized mechanism clears one or more auctions involving a population of bidders with various, generally unknown, bidding strategies. The results of the auction are then taken as input to the adaptive algorithm as it revises the mechanism parameters in an effort to maximize an objective function such as seller revenue. Any number of continuous or discrete auction parameters may be considered, such as reserve prices, auctioneer fees, minimum bid increments, and whether the close is hard or soft [Wurman *et al.*, 2001].

The bidders in Figure 3 may use a variety of different bidding strategies, including heuristic, analytic, and learning-based approaches. For the latter to make sense, the same bidders must interact repeatedly with the mechanism, leading to a potential co-evolutionary scenario in which the bidders and mechanism continue to adapt in response to each other [Phelps *et al.*, 2002]. However, our approach does not depend on repeated interactions with the same bidders. The only required assumption about the bidders is that their behavior is somewhat consistent (e.g. bidders associated with a particular industry tend to bid similarly) for a sufficient period of time to allow for prediction of auction results as a function of the mechanism, at least in expectation. The main contribution of our work in this area is the presentation of a *metalearning* technique with which information about potential bidder behavior can be used to guide the selection of the method of adaptation and significantly improve auctioneer revenue.

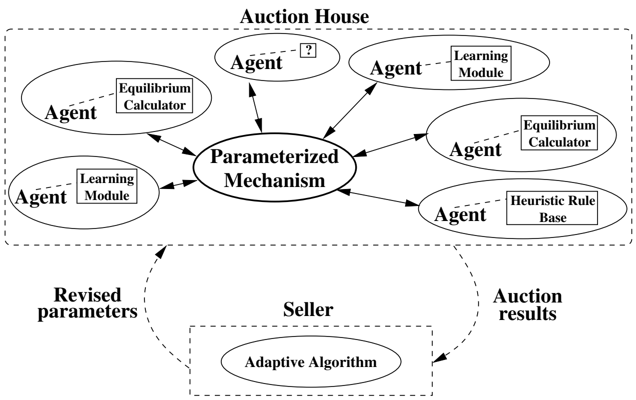

Figure 3: A high-level illustration of the concept of adaptive mechanisms. From the point of view of the seller, the bidder behaviors are unknown aspects of the environment.

There are several directions in which this work could be extended. Many auction parameters are available for tuning, ranging from bidding rules to clearing policies. The problem becomes more challenging in the face of multidimensional parameterizations. The choice of metalearning algorithm itself is a possible area for improvement as well. And perhaps most interestingly, the effects of including some adaptive bidders in the economies that are treated by adaptive mechanisms are currently unknown.

3.2 General Game Playing

As presented in the previous section, the idea behind adaptive mechanism design is that an agent may actively change the rules of market interaction for bidding agents as time goes on. For such an approach to be feasible in practice, 1) there will need to be a way to specify the current rules of interaction to the participating agents, and 2) the agents must be able to figure out for themselves how to bid in any mechanism that is so specified. Challenges similar to both of these have been recently addressed in the context of general game playing.

Creating programs that can play games such as chess, checkers, and backgammon, at a high level has long been a challenge and benchmark for AI. While several game-playing systems developed in the past, such as Deep Blue [Campbell *et al.*, 2002], Chinook [Schaeffer *et al.*, 1992], and TD-gammon [Tesauro, 1994] have demonstrated competitive play against human players, such systems are limited in that they play only one particular game and they typically must be supplied with game-specific knowledge. While their performance is impressive, it is difficult to determine if their success is due to the particular game-playing technique or due to the human game analysis.

A *general* game playing agent [Pell, 1993] must be able to take as input a description of a game's rules and proceed to play without any human input, despite having had no prior experience in that game. Doing so requires the integration of several AI components, including theorem proving, feature discovery, heuristic search, and potentially learning.

Kuhlmann *et al.* [2006a] present a complete and fully autonomous general game playing agent designed to participate in the first AAAI General Game Playing (GGP) Competition which was held at AAAI 2005 [Genesereth and Love, 2005]. In that setting, a restricted class of games is considered, namely discrete state, deterministic, perfect information games. The games can be single or multi-player and they can be turn-taking or simultaneous decision.

Our main contribution in this setting is a novel method for automatically constructing effective search heuristics based on the formal game description. The agent analyzes the game description to automatically detect relevant features, such as the number of white pieces on a board, to encode within heuristic evaluation functions for use in iterative-deepening alpha-beta search. Banerjee and Stone [2007] have also shown the applicability of RL transfer learning in this setting.

Research on GGP is still in its very early stages, with current agents performing far worse than agents with domain-specific evaluation functions. Though perhaps GGP agents will never rival special-purpose game players, there are many open research directions pertaining to automatically generating heuristic evaluation functions that may help close the gap.

3.3 Agent Modeling

For fully autonomous agents in multiagent environments, the ability to predict the behavior of other agents in the environment can be crucial to one's own performance. Specifically, knowing the likely actions of other agents can influence an agent's expected distribution over future world states, and thus inform its planning of future actions.

In an adversarial environment, this predicted behavior of other agents is referred to as an *opponent model*. Opponent models are particularly useful if they include some identification of potential patterns or weaknesses on the part of the opponent. For example, a chess grandmaster may study past games of a future opponent so as to determine how best to play away from that opponent's strengths.

In multiagent adversarial settings, in which the adversary consists of a *team* of opponents, it can be useful to explicitly model the opponent as engaging in team activities. For example, Tambe [1996] presents a simulated air-combat scenario in which an individual's behavior can indicate the commencement of a team "pincer" maneuver that requires multiple participants, thus enabling the prediction of other opponents' future actions as well.

One setting in which opponent modeling research has been conducted is the RoboCup simulation coach competition. RoboCup is an international research initiative that uses the game of soccer as a testbed to advance the state of the art in AI and robotics [Kitano *et al.*, 1997a]. In most RoboCup soccer leagues the goal is to create complete teams of agents that can succeed at the task of winning soccer games. Though opponent modeling can play a part in this task, it is often the case that opponents cannot be observed prior to playing against them (at least not by the agents themselves). Even when they can be observed, opponent modeling challenges are easily overshadowed by challenges such as vision, localization, locomotion, individual ball manipulation, and teamwork.

In contrast, the goal of the simulation coach competition is to focus entirely on opponent modeling. This focus is accomplished by 1) providing entrants with recordings of the opponents' past play that is understandable by the coach agent; 2) providing each entrant with an identical team of fully competent player agents; and 3) restricting the actions available to *advice* regarding how the team should alter its playing style to fit a particular opponent.

In this context, Riley *et al.* [2002] approach advice-giving as an action-prediction problem. Both offensive and defensive models are generated using the C4.5 [Quinlan, 1993] decision tree learning algorithm. Their work also stresses the importance of learned formation advice. Subsequently, Kuhlmann et al. [2005; 2006b] decompose the problem similarly, but using different model representations and advice-generation procedures.

In other work, Riley and Veloso [2002] use Bayesian modeling to predict opponent movement during set plays. The model is used to generate adaptive plans to counter the opponent's plays. In addition, Riley and Veloso [2000] have tried to model high-level adversarial behavior by classifying opponent actions as belonging to one of a set of predefined behavioral classes. Their system could classify fixed duration *windows* of behavior using a set of sequence-invariant action features.

Opponent team modeling has also been studied in military-like scenarios. In addition to Tambe's work mentioned above [Tambe, 1996], Sukthankar and Sycara [2005] use HMMs to monitor and classify *human* team behavior in a MOUT (military operations in urban terrain) scenario, especially focusing on sequential team behaviors.

As the number and variety of agents increases, methods for agent modeling will only become more central to enabling agent autonomy. Whether through recursive modeling methods based on deep knowledge of the other agents' internal states [Vidal and Durfee, 1995]; strictly via observation of the other agents' actions [Huber and Durfee, 1995]; or somewhere in between, there will continue to be a need for methods that enable prediction of other agents' future actions.

4 Robotics

The general topics of learning and multiagent reasoning are relevant to autonomous agents of all kinds, including software agents. But to the extent that one goal of AI is to enable the creation of intelligent *physical* agents, or robots, that can coexist with us in the real world, it is important to consider to what extent these topics must be tailored to facilitate learning, interacting robots.

Compared to other machine learning scenarios such as classification or action learning by an individual agent in simulation, multiagent learning on physical robots presents several formidable challenges, including the following.

Sparse Training Data: It is often prohibitively difficult to generate large amounts of data due to the maintenance required on robots, such as battery changes, hardware repairs, and, usually, constant human supervision. Thus learning methods designed for physical robots must be effective with small amounts of data.

Dynamic Environments: Robots are inherently situated in a dynamically changing environment with unpredictable sensor and actuator noise, namely the real world. When acting in teams, they must adapt to each other's changing behaviors in addition to changes in the environment. Thus learning algorithms designed for teams of physical robots must be able to adapt quickly and continually.

Due at least in part to these challenges, most research on learning robots to date has focused on individual tasks or relatively simple multi-robot tasks. However, recent successful applications of machine learning to complex dynamic environments with limited training examples, both with multiple agents in simulation and with individual robots in the real world, suggest that the time is right for a concerted effort towards further developing learning methods that can be deployed on teams of physical robots. As such, this section focuses on research pertaining to learning and multiagent reasoning on physical robots.

4.1 Learned Behaviors on Physical Robots

Ideally, a robot should be able to respond to a change in its surroundings by adapting both its low-level skills, such as its walking style, and the higher-level behaviors that depend on them. Because hand-coding is time-consuming and often leads to brittle solutions, one would like this adaptation to occur as autonomously as possible.

Unfortunately, current learning methods typically need a large amount of training data to be effective. If that data must be gathered by a robot in the real world, the amount of time required for learning could become prohibitively large. One possible way to alleviate this problem is to train behaviors first in simulation before implementing them in the real world [Davidor, 1991; Gat, 1995; Porta and Celaya, 2001]. However, especially when concerned with complex perception or manipulation tasks, we cannot assume an adequate simulator will always exist for a given robot. With no simulator, each trial requires interaction with the physical world in real time. In such cases, it is not possible to offset the costs of an inefficient learning algorithm with a faster processor. The learning algorithm must make efficient use of the information gained from each trial (i.e., it must have low sample complexity).

For this reason, until recently, most of the locomotion approaches for quadrupedal robots have focused on hand-tuning a parameterized gait. This approach has been somewhat fruitful: since the introduction of the Sony Aibo robot [Sony, 2004] in 1998, the walking speed of the Aibos has increased significantly via manual gait tuning. However, the process of hand-tuning a parameterized gait both can be time-consuming and can require a good deal of human expertise. Furthermore, a change of robot hardware and/or the surface on which it is to walk necessitates retuning.

As an alternative to hand-tuning a parameterized gait, machine learning can be used to automate the search for good parameters. Various machine learning techniques have proven to be useful in finding control policies for a wide variety of robots including helicopters [Bagnell and Schneider, 2001; Ng *et al.*, 2004], biped robots [Zhang and Vadakkepat, 2003] and Aibos [Hornby *et al.*, 1999; 2000; Kim and Uther, 2003]. Kohl and Stone [2004b] present a policy gradient learning approach for generating a fast walk on legged robots. Using this method, we have created a walk that is faster than hand-tuned gaits and was among the fastest learned gaits of its time.[4] A key feature of our approach is that the robots time *themselves* walking across a known, fixed distance, thus eliminating the need for any human supervision.

Fidelman and Stone [2007] further demonstrate that it is possible to similarly learn a higher-level fine-motor skill, again with all learning occurring directly on the robot. In particular, the Aibo is able to learn a ball-grasping skill with no human intervention other than battery changes. The learned skill significantly outperforms our best hand-tuned solution.

As the learned grasping skill relies on a learned walk, we characterize our learning implementation within the framework of layered learning, as introduced in Section 2.3. This research represents the first implementation of layered learning on a physical robot, with all training performed in the real world. Extending this approach to other robots and additional hierarchical behavior layers is a promising direction for future research.

4.2 Learned Sensor and Actuator Models

One popular approach towards achieving the goal of robust autonomy in robots is to imbue the robot with a general reasoning capability that relies on 1) a *model* of the current state of the world; and 2) a model of the effects of the robot's actions on the world. Given these models, the robot can then plan its actions so as to best achieve its goals given the current state of the world.

For such a model-based approach to be effective, the sensor and actuator models must be accurate and well-calibrated, at least in relation to one another. For example, state-of-the-art robot localization algorithms such as particle filtering [Dellaert *et al.*, 1999; Kwok *et al.*, 2003] rely on calibrated sensor and actuator (odometry) models to fuse sensory and action history information into a unified probabilistic estimate of the robot's location.

Currently, these sensor and actuator models are typically calibrated manually: sensor readings are correlated with actual measured distances to objects, and robot actuator commands are measured with a stopwatch and a tape measure. However this type of approach has three significant drawbacks. First, it is labor intensive, requiring a human operator to take the necessary measurements. Second, for sensors and actuators with many (perhaps infinitely many) possible readings or parameter settings, the measured model can only be made to coarsely approximate the complete model. Third, and perhaps most importantly, the model is necessarily tuned to a specific environment and may not apply more generally. This *brittleness* is a major motivation for the automatic model building advocated by Stronger and Stone [2006].

When considered in isolation, there is no choice but to build each individual sensor and action model manually. In practice, however, each of the robot's sensors as well as its action selection mechanism can be related through their reflection on the world state, as illustrated in Figure 4.

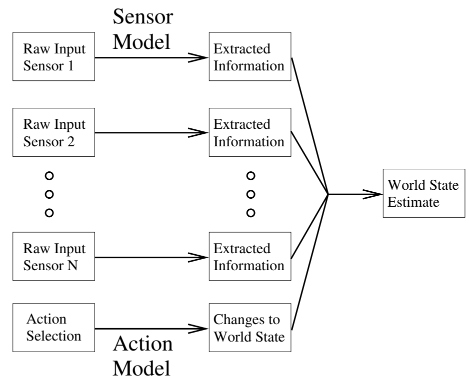

Figure 4: The flow of information on an autonomous robot. The data from each sensor and the action selections are interpreted based on the robot's action and sensor models, represented by arrows here. The resulting *Extracted Information* can then be used to inform the robot's estimate of the state of the world.

Motivated by this relationship, Stronger and Stone [2006] introduce ASAMI (Autonomous Sensor and Actuator Model Induction), a general methodology by which a robot can learn its action and sensor models *from each other*. Because the world state can be estimated from independent, redundant sources of information, the robot can use this information to learn a model of any given individual source. It learns each model by comparing the input to that model to an estimate of the world state based on all of the other information sources. Figure 5 shows (in a simplified setting) how the robot can use redundant information to learn its action and sensor models. For the sensor model, the world state estimate is first relayed back through arrow A to the "information about world state" from the sensor model. This tells us what the output of the sensor model should have been assuming the world state estimate is perfectly accurate. When this data is combined with the raw sensory input via arrow B, the result is training data for learning the sensor model. This data can be processed by supervised learning method based on the structure of the sensor model. Similarly, the world state estimate can be relayed back to the "changes to world state" from the action model (arrow C). In this case, the world state, if accurate, indicates how the world actually changed as a result of previous actions. This information can be combined with the action selections (arrow D) to train the action model.

The benefit of ASAMI is that it enables a robot to induce models of its sensors and actions without any manually labeled training data. That is, the only inputs to the learning process are the data the robot would naturally have access to: its raw sensations and its knowledge of its own action se-

[4] Since our original report on this method [Kohl and Stone, 2004a], there has been a spate of research on efficient learning algorithms for quadrupedal locomotion [Chen, 2005; Chernova and Veloso, 2004; Cohen *et al.*, 2004; Dueffert and Hoffmann, 2005; Kim and Uther, 2003; Quinlan *et al.*, 2003; 2005; Roefer *et al.*, 2005; Rofer, 2004; Rofer *et al.*, 2003].

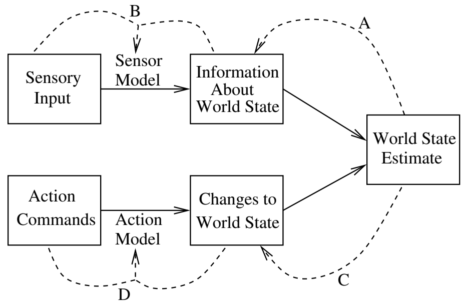

Figure 5: Dashed arrows A through D show how information can be propagated back from a redundantly informed world state estimate to calibrate the action and sensor models.

lections. This general methodology promises to increase the robustness and general applicability of autonomous robots. If a fully featured robot could automatically calibrate all of its processing modules, it would be able to navigate under a variety of environmental conditions without any human supervision. However, achieving this goal is not straightforward. Standard techniques for model induction require accurately labeled training data; in the absence of manually labeled data, the robot must combine information from its actuators and sensors to bootstrap accurate estimates of both models.

ASAMI is instantiated in a broad class of settings, namely those in which an agent navigates through a one-dimensional state space with a sensor that provides information about the state of the world and actions that determine the world state's rate of change. Examples of such settings include a robot on a track with a global positioning sensor and a velocity control; a temperature regulator; and a vehicle with a throttle whose settings correspond to accelerations. ASAMI works by simultaneously learning an action model and a sensor model, each one based on the current estimate of the other one. This bootstrapping process enables the agent to learn accurate approximations to its true action and sensor models, starting with only a very simplistic action model estimate. Furthermore, the learning process is completely autonomous and unsupervised, so that no human oversight or feedback is necessary.

ASAMI is implemented and validated in a robotic test-bed domain based on the Sony Aibo ERS-7. In experimental tests, the robot learns models from its action commands to the resultant velocities and from its visual sensor readings to the corresponding distances. The learning process takes only two and a half minutes of completely autonomous behavior.

Looking forward, we consider ASAMI to be a first step towards enabling a robot to autonomously navigate through a very high-dimensional space while learning many functions with various numbers of input and output variables. Extending ASAMI in this direction raises two main challenges. First, ASAMI currently allows a robot to learn two functions: one sensor model and one action model. With more than two models to learn, the robot will have *multiple* sources of information that can be used to train each model. This situation raises the question of how these sources can best be integrated with each other to provide effective training data for each model. Second, ASAMI currently assumes that the robot operates in a one-dimensional state space, thus restricting its learned models to being from one variable to one other variable. For many applications, it will be necessary or useful to learn functions with multiple input and output variables. In this case, straightforward polynomial regression, as used in the initial implementation, is insufficient. In principle, the ASAMI methodology should extend to other general-purpose function approximators such as neural networks and CMACs. However, additional research is needed to determine how these methods may need to be adapted for automated model learning.

4.3 Learned Vision

To operate in the real world, autonomous robots depend on their sensory information. Visual input, in the form of color images from a camera, should be an excellent and rich source of such information. But color, and images in general, have been used sparingly on mobile robots, where people have mostly focused their attention on other sensors such as tactile sensors, sonar, and laser. There are three main reasons for this reliance on other relatively low-fidelity sensors. First, most sophisticated vision algorithms require substantial amounts of computational and/or memory resources, making them infeasible to use on mobile robotic systems that demand real-time processing. Second, most vision algorithms assume a stationary or slowly moving camera and hence cannot account for the rapid non-linear camera motion that is characteristic of mobile robots. Third, the variation of illumination over the operating environment causes a nonlinear shift in color distributions that is difficult to model; mobile robots, while moving around the world, often go into places with changing illumination.

Even in the face of these formidable challenges, one factor that can be leveraged is that many mobile robot applications involve a *structured* environment with objects of unique shape and color — information that can be exploited to help overcome the problems mentioned above. Sridharan and Stone focus on designing algorithms that can work within the robot's computational and environmental constraints while making the best use of the available information.

First, we present a prototype vision system that works on-board an autonomous robot using vision as the primary source of information [Sridharan and Stone, 2005b]. This baseline system includes solutions to two main vision challenges, color segmentation[5] and object recognition, that can run within the robot's hardware constraints. The system is robust to the presence of jerky non-linear camera motion and noisy images. However, the baseline system relies on *manually* labeled training data and operates in *constant* (and reasonably uniform) illumination conditions.

Second, we use the *structure* inherent in the environment to eliminate the need for manual labeling. The robot uses the knowledge of the location of unique objects in its world to autonomously learn the desired colors required for color segmentation, thereby eliminating the time-consuming and brit-

[5] Color segmentation is the problem of mapping 3-dimensional integral pixel values $\in [0, 255]^3$ to color labels such as red, orange, or blue.

tle manual calibration process. This information is also used by the robot to navigate around its environment [Sridharan and Stone, 2005a].

Third, we enable the robot to perform efficiently even in the presence of illumination variations, a challenging vision problem because of the corresponding non-linear shift in color distributions: the very same pixel values corresponding to a color in one illumination may correspond to a completely different color in another illumination. As the robot navigates in its moderately structured world, it autonomously detects and adapts to the changes in illumination conditions [Sridharan and Stone, 2005c].

Fourth, we unify these results by enabling a robot 1) to recognize when the illumination has changed sufficiently to require a completely new color map rather than using one of the existing ones; and 2) to plan its own action sequence for learning the new color map on-line [Sridharan and Stone, 2007].

All of the algorithms above run in real-time on a physical Aibo ERS-7 robot enabling it to operate autonomously in an uncontrolled environment with changing illumination over an extended period of time. In the end, the robot is able to detect changes in illumination robustly and efficiently, *without prior knowledge of the different illumination conditions*. When the robot detects an illumination condition that it has already learned, it smoothly transitions to using the corresponding color map.

One direction of future work is to have the robot adapt to minor illumination changes by suitably modifying specific color distributions. The ultimate research goal along this path is to develop efficient algorithms for a mobile robot to function autonomously under uncontrolled natural lighting conditions.

4.4 Communication Connectivity

Many applications of distributed autonomous robotic systems can benefit from, or even may require, the team of robots staying within communication connectivity. For example, consider the problem of multirobot surveillance [Parker, 2002; Ahmadi and Stone, 2006b], in which a team of robots must collaboratively patrol a given area. If any two robots can directly communicate at all times, the robots can coordinate for efficient behavior. This condition holds trivially in environments that are smaller than the robots' communication ranges. However in larger environments, the robots must actively maintain physical locations such that any two robots can communicate, possibly through a series of other robots. Otherwise, the robots may lose track of each other's activities and become miscoordinated. Furthermore, since robots are relatively unreliable and may need to change tasks (for example if a robot is suddenly called by a human user to perform some other task), in a stable multirobot surveillance system, if one of the robots leaves or crashes, the rest should still be able to communicate. Some examples of other tasks that could benefit from any pair of robots being able to communicate with each other, are space and underwater exploration, search and rescue, and cleaning robots.

We say that robot R_1 is *connected* to robot R_2 if there is a series of robots, each within communication range of the previous, which can pass a message from R_1 to R_2. In order for the team to stay reliably connected, it must be the case that every robot is connected to each other robot either directly or via *two* distinct paths that do not share any robots in common. We call this property *biconnectivity*: the removal of any one robot from the system does not disconnect the remaining robots from each other.

Ahmadi and Stone [2006a] study this problem of enabling robots to remain connected in the face of robot failures. We address this problem by dividing it into three main steps: 1) checking whether a team of robots is *currently* biconnected, 2) maintaining biconnectivity should a robot be removed from (or added to) the team, and 3) constructing a biconnected multi-robot structure from scratch. To be applicable for teams of autonomous robots, all algorithms must be fully distributed.

Our work to date addresses Step 1 under the assumption that robots have constant and identical communication ranges. This assumption applies in the case of homogeneous robot teams (or at least teams with homogeneous transmitters) such that the range is not dependent on a robot's battery level. This assumption allows us to assume the connection graph among robots is undirected: if robot A can send a message to robot B, then the reverse is also true. The heterogeneous case, along with Steps 2 and 3 are natural directions for future work in this area.

5 Applications

Sections 2–4 treated learning and multiagent reasoning as components of autonomous agents. But as pointed out in the introduction courtesy of Koller's quote, doing so has the risk of fragmenting the field. Her reaction to this risk was to provide conceptual bridges among three different AI topics. But another way to address the risk is to build applications that require the practical unification of the various topics into a complete agent, in our case one that learns, interacts, and perhaps acts in the real world.

This notion of starting research from applications is complementary to Stuart Russell's metaphor in his Computers and Thought paper. He wrote:

> Theoreticians can produce the AI equivalent of bricks, beams, and mortar with which AI architects can build the equivalent of cathedrals. [Russell, 1995]

On the other hand architects may, by creating new artifacts that exhibit previously unseen properties, lead theoreticians to the possibility of, or even need for, new bricks and beams. In other words, one valuable way to advance the field is to study complete agents in specific, complex domains, with the ultimate goal of drawing general lessons from the specific implementations. From this point of view, theoreticians and architects share a complementary, bi-directional dependency on one another. Theory paves the way for practice and also vice versa: it is not a one-way road.

Consistent with this view, Russell writes that "AI is a field defined by its problems, not its methods" [Russell, 1995]. In the remainder of this section, I briefly describe four such problems that have been useful to me in my pursuit of agents that can learn and interact with one another: robot soccer, autonomous bidding agents, autonomic computing, and autonomous vehicles. The first and last are physical domains with real robots, while the other two focus on software agents. The second and third are also interesting for the connections they provide between AI and other research areas, namely economics and computer systems respectively.[6] Most importantly, all four are well-suited for the integration of machine learning and multiagent reasoning, including opportunities for leveraging the research directions emphasized in Sections 2 and 3 such as adaptive and hierarchical representations, layered learning, transfer learning, adaptive interaction protocols, and agent modeling.

5.1 Robot Soccer

The original motivating domain for my research on multiagent learning was robot soccer. Robot soccer pits two teams of independently-controlled agents against each other. Originated by Mackworth [1993], it has been gaining popularity over the past decade, with several international "RoboCup" competitions taking place [Kitano, 1998; Asada and Kitano, 1999; Veloso *et al.*, 2000; Stone *et al.*, 2001a; Birk *et al.*, 2002; Kaminka *et al.*, 2003; Polani *et al.*, 2004; Nardi *et al.*, 2005; Noda *et al.*, 2006]. It was the subject of an official IJCAI-97 Challenge [Kitano *et al.*, 1997b] and poses a long-term grand challenge to the field, namely the creation of a team of humanoid robots that can beat the best human soccer team on a real soccer field by the year 2050. RoboCup's explicit goal is to encourage research in the fields of AI and robotics, with a particular emphasis on collaborative and adversarial reasoning among autonomous agents in dynamic multiagent environments. One advantage of the domain is that robot soccer can be used to evaluate different AI techniques in a direct manner: teams implemented with different algorithms can play against each other.

Currently, RoboCup includes five robot soccer leagues (simulation, small-size, mid-size, four-legged, and humanoid), with each one emphasizing slightly different research issues depending on the physical properties of the robots. The commonalities across the leagues are that they are run in *dynamic, real-time, distributed*, *multiagent* environments with both *teammates* and *adversaries*. In general, there is *hidden state*, meaning that each agent has only a partial world view. The agents also have *noisy sensors and actuators*, meaning that they do not perceive the world exactly as it is, nor can they affect the world exactly as intended. In addition, the perception and action cycles are *asynchronous*, prohibiting the traditional AI paradigm of using perceptual input to trigger actions. *Communication* opportunities are limited; and the agents must make their decisions in *real-time*. These italicized domain characteristics combine to make robotic soccer a realistic and challenging domain.

Much of the research described in Sections 2–4 has been directly motivated by the challenges posed by the various RoboCup leagues.

[6]Russell also specifically portrays the "influx of new methods from other fields" and the embracing of AI methods by other fields as positive developments [Russell, 1995].

5.2 Autonomous Bidding Agents

In contrast to robot soccer, in which all the agents are either fully collaborative teammates or fully adversarial opponents, research on autonomous bidding agents presents an opportunity to consider agents whose goals are independent: an economic agent seeks to maximize its own profit regardless of the effects on the profits of other agents.

Like RoboCup, autonomous bidding has been the subject of annual competitions for several years. The first Trading Agent Competition (TAC) was held in 2000 [Wellman *et al.*, 2001] with the goal of providing a benchmark problem in the complex and rapidly advancing domain of e-marketplaces [Eisenberg, 2000] and motivating researchers to apply unique approaches to a common task.

One key feature of TAC is that it requires autonomous bidding agents to buy and sell *multiple interacting goods* in auctions of different types. Another key feature is that participating agents compete against each other in preliminary rounds consisting of many games leading up to the finals. Thus, developers change strategies in response to each other's agents in a sort of escalating arms race. Leading into the day of the finals, a wide variety of aggregate economic scenarios is generally possible. A successful agent needs to be able to perform well in any of these possible circumstances.

Current TAC domains include a travel scenario in which agents procure flights, hotels, and entertainment tickets for clients with various travel preferences; and a supply chain management scenario in which agents manage a PC manufacturing process, purchasing components from suppliers, deciding what computers to produce, and bidding for customer orders. The next TAC is planned to focus on adaptive mechanism design, as described in Section 3.1.

In these domains, the multiagent learning opportunities tend to pertain to modeling the aggregate effect of the other agents on the economy, either from extensive offline data [Stone *et al.*, 2003], or in response to short-term changes in the other agents' strategies [Stone *et al.*, 2001b; Pardoe and Stone, 2006]. Note that even though it is a domain for strictly software agents, autonomous bidding is very much a real-world problem, with potential high-stakes economic impact.

5.3 Autonomic Computing

At the intersection of AI and computer systems research, autonomic computing [Kephart and Chess, 2003] has the potential to address shortcomings in today's systems and to enable future systems. Autonomic computing refers to a broad set of strategies to reduce the amount of complexity exposed to human operators of computing systems. Autonomic systems need to intelligently make complex decisions based on large amounts of uncertain, heterogeneous information. The area of machine learning has made significant progress in developing methods that automate the construction of such complex decision-making systems by inducing robust models directly from relevant empirical data.

Many recent papers have identified systems problems that can benefit from machine learning [Wildstrom *et al.*, 2005; Chen *et al.*, 2004; Liblit *et al.*, 2003; 2005; Mesnier *et al.*, 2004; Fern *et al.*, 2004; Kolter and Maloof, 2004;

Chang *et al.*, 2004; Fox *et al.*, 2004; Gomez *et al.*, 2001; Walsh *et al.*, 2004; Murray *et al.*, 2005; Newsome *et al.*, 2005]. My own experience, and the experience of others, shows that machine learning cannot be integrated into systems as a simple black box. Rather, to achieve the goals of autonomic computing we will need to achieve a much tighter coupling between systems and machine learning in which system designs are adapted to facilitate machine-learning-based control, and in which machine learning techniques are advanced to meet the demands of large-scale systems.

Autonomic computing may be a candidate for the next successful application of layered learning. For entire systems to be able to self-diagnose failures and repair themselves, there will need to be learning components at multiple levels, including the OS, databases, and networking modules. Thus a paradigm like layered learning may be essential to keeping the entire system operating smoothly.

In addition to learning, autonomic commuting is also fundamentally a multiagent problem. In today's highly interconnected world, computers are generally networked together such that failures or upgrades on one system can have internet-wide effects, for instance with respect to packet routing or grid services.

Like autonomous bidding agents, autonomic computing is a software agents domain with high potential for real-world impact. I believe that autonomic computing will become an increasingly important domain for testing, deployment, and development of machine learning and multiagent reasoning advances in the years to come.

5.4 Autonomous Vehicles

Enabling cars to drive autonomously in cities is currently technologically feasible, and will likely be economically feasible within the next 5–10 years. Indeed, General Motors has announced that it plans to release a nearly autonomous vehicle under its European "Opel" brand. The 2008 Opel Vectra will be able to drive itself at speeds up to 60 miles per hour, even in heavy traffic.

Such autonomous vehicles will change the way we think about transportation, for example enabling people to concentrate on other activities while "driving," and enabling minors and the elderly to be transported on their own. As a result, once there's a single autonomous vehicle, there will likely be many more, and every major automobile company will need to respond.

The successful DARPA Grand Challenge [DARPA, 2006] has shown that current AI can produce autonomous, embodied agents capable of navigating the Mojave Desert. While certainly no small feat, traversing a barren desert devoid of pedestrians, narrow lanes, and multitudes of other fast-moving vehicles solves at best half the problem. As Gary Bradski, a researcher at Intel Corp. said following the successful completion of the 2005 Grand Challenge by "Stanley," a modified Volkswagen Touareg [Montemerlo *et al.*, 2006], "Now we need to teach them how to drive in traffic" [Johnson, 2005].

While autonomous vehicles driving in traffic may seem to be a long way off, advances in AI, and more specifically, Intelligent Transportation Systems [Bishop, 2005], suggest that it may soon be a reality. Cars can already be equipped with features of autonomy such as adaptive cruise control, GPS-based route planning [Rogers *et al.*, 1999; Schonberg *et al.*, 1995], and autonomous steering [Pormerleau, 1993; Reynolds, 1999]. Some current production vehicles even sport these features. In addition to the Opel Vectra mentioned above, DaimlerBenz's Mercedes-Benz S-Class has an adaptive cruise control system that can maintain a safe following distance from the car in front of it, and will apply extra braking power if it determines that the driver is not braking hard enough. Both Toyota and BMW are currently selling vehicles that can parallel park completely autonomously, even finding a space in which to park without driver input.

Once such autonomous vehicles are *possible*, it is only a matter of time before they become affordable, and then ubiquitous. A natural question that then arises is whether the current traffic control paradigms, which are designed for human drivers, are appropriate for such autonomous drivers. Dresner and Stone [2004] have created and implemented a novel algorithm that enables cars and intersections to autonomously negotiate fine-grained *reservations* for the cars to pass through. We demonstrate that our approach can lead to more than a 100-fold decrease in delays at busy intersections when compared to standard approaches such as traffic lights. This foundational result opens the way to the investigation of multiagent learning and market-based methods for autonomous vehicle navigation, a direction of inquiry that promises several years' worth of fruitful research challenges in multiagent reasoning and machine learning.

6 Conclusion

In summary, the most exciting research topics to me are those inspired by challenging real-world problems. Furthermore, successful research results involve 1) fully implemented solutions; 2) general algorithms that transcend individual domains; and 3) theoretical explanations for, or bounds on, the effectiveness of these algorithms. From where we stand today, there is both the need and the foundation for such contributions in AI aimed at enabling fully autonomous agents, both in software and physically embodied, to learn and interact with one another.

Throughout this paper I have summarized what I see to be the most interesting and promising areas for current and future research pertaining to machine learning, multiagent reasoning, and robotics with the ultimate goal of enabling the creation of robust, fully autonomous agents that are able to learn and interact with one another. By leveraging new theoretical results to inform practical implementation, and by taking advantage of innovations from concrete multiagent applications to inspire new theory, I believe that the field can continue to make fast and exciting progress towards this goal.

Acknowledgments

The research presented in this paper would not have been possible without the active collaboration of my colleagues and students, most notably my Ph.D. advisor Manuela Veloso, and my students in the Learning Agents Research Group at The University of Texas at Austin including Mazda Ahmadi, Kurt Dresner, Peggy Fidelman, Nicholas K. Jong, Nate Kohl, Gregory Kuhlmann, David Pardoe, Mohan Sridharan, Daniel Stronger, Matthew E. Taylor, Shimon Whiteson, and Jonathan Wildstrom. This research was supported in part by NSF CAREER award IIS-0237699, DARPA grant HR0011-04-1-0035, DARPA/AFRL grant FA8750-05-2-0283, ONR YIP award N00014-04-1-0545, an IBM faculty award, and a Sloan Foundation Fellowship.

References

[Ackley and Littman, 1991] David Ackley and Michael Littman. Interactions between learning and evolution. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*. Addison-Wesley, 1991.

[Ahmadi and Stone, 2006a] Mazda Ahmadi and Peter Stone. Keeping in touch: A distributed check for biconnected structure by homogeneous robots. In *The 8th International Symposium on Distributed Autonomous Robotic Systems*, July 2006.

[Ahmadi and Stone, 2006b] Mazda Ahmadi and Peter Stone. A multi-robot system for continuous area sweeping tasks. In *Proceedings of International Conference on Robotics and Automation (ICRA), to appear.*, May 2006.

[Asada and Kitano, 1999] Minoru Asada and Hiroaki Kitano, editors. *RoboCup-98: Robot Soccer World Cup II*. Lecture Notes in Artificial Intelligence 1604. Springer Verlag, Berlin, 1999.

[Asada *et al.*, 1994] Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. Vision-based behavior acquisition for a shooting robot by using a reinforcement learning. In *Proc. of IAPR/IEEE Workshop on Visual Behaviors-1994*, pages 112–118, 1994.

[Bagnell and Schneider, 2001] J. Andrew Bagnell and Jeff Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *International Conference on Robotics and Automation*, pages 1615–1620. IEEE Press, 2001.

[Baird and Moore, 1999] L. C. Baird and A. W Moore. Gradient descent for general reinforcement learning. In Michael J. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 968–974. The MIT Press, 1999.

[Baird, 1995] L.C̃. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*. Morgan Kaufman, July 1995.

[Banerjee and Stone, 2007] Bikramjit Banerjee and Peter Stone. General game learning using knowledge transfer. In *The 20th International Joint Conference on Artificial Intelligence*, January 2007. To appear.

[Birk *et al.*, 2002] Andreas Birk, Silvia Coradeschi, and Satoshi Tadokoro, editors. *RoboCup-2001: Robot Soccer World Cup V*. Springer Verlag, Berlin, 2002.

[Bishop, 2005] Richard Bishop. *Intelligent Vehicle Technology and Trends*. Artech House, 2005.

[Boers *et al.*, 1995] E.J.W. Boers, M.V. Borst, and I.G. Sprinkhuizen-Kuyper. Evolving Artificial Neural Networks using the “Baldwin Effect”. In *Artificial Neural Nets and Genetic Algorithms, Proceedings of the International Conference in Ales, France*, 1995.

[Bond and Gasser, 1988] Alan H. Bond and Les Gasser. An analysis of problems and research in DAI. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 3–35. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Brooks, 1986] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, 1986.

[Brooks, 1991] Rodney A. Brooks. Intelligence without reason. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[Campbell *et al.*, 2002] Murray Campbell, A. Joseph Hoane Jr., and Feng Hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1–2):57–83, 2002.

[Cao *et al.*, 1997] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:7–27, 1997.

[Chang *et al.*, 2004] Y-Han Chang, Tracy Ho, and Leslie Pack Kaelbling. Mobilized ad-hoc networks: A reinforcement learning approach. In *Proceedings of the First International Conference on Autonomic Computing*, May 2004.

[Chen *et al.*, 2004] Mike Chen, Alice X. Zheng, Jim Lloyd, Michael I. Jordan, and Eric Brewer. Failure diagnosis using decision trees. In *Proceedings of the First International Conference on Autonomic Computing*, May 2004.

[Chen, 2005] Weiming Chen. Odometry calibration and gait optimisation. Technical report, The University of New South Wales, School of Computer Science and Engineering, 2005.

[Chernova and Veloso, 2004] Sonia Chernova and Manuela Veloso. An evolutionary approach to gait learning for four-legged robots. In *In Proceedings of IROS’04*, September 2004.

[Cohen *et al.*, 2004] David Cohen, Yao Hua Ooi, Paul Vernaza, and Daniel D. Lee. The University of Pennsylvania RoboCup 2004 legged soccer team, 2004. Available at URL `http://www.cis.upenn.edu/robocup/UPenn04.pdf`.

[Cramton, 1997] Peter C. Cramton. The FCC spectrum auctions: An early assessment. *Journal of Economics and Management Strategy*, 6(3):431–495, 1997.

[Crites and Barto, 1996] Robert H. Crites and Andrew G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 1017–1023, Cambridge, MA, 1996. MIT Press.

[DARPA, 2006] DARPA. The DARPA grand challenge, 2006. `http://www.darpa.mil/grandchallenge`.

[Dautenhahn, 1995] Kerstin Dautenhahn. Getting to know each other—artificial social intelligence for autonomous robots. *Robotics and Autonomous Systems*, 16:333–356, 1995.

[Davidor, 1991] Yuval Davidor. *Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization*. World Scientific Publishing Co., Inc., NJ, USA, 1991.

[Dean and Givan, 1997] Thomas Dean and Robert Givan. Model minimization in Markov decision processes. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.

[Decker, 1987] Keith S. Decker. Distributed problem solving: A survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(5):729–740, September 1987.

[Dellaert *et al.*, 1999] F Dellaert, D Fox, W Burgard, and S Thrun. Monte carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[Dietterich, 2000] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[Dresner and Stone, 2004] Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 530–537, July 2004.

[Dueffert and Hoffmann, 2005] Uwe Dueffert and Jan Hoffmann. Reliable and precise gait modeling for a quadruped robot. In *RoboCup 2005: Robot Soccer World Cup IX, Lecture Notes in Artificial Intelligence*. Springer, 2005.

[Eisenberg, 2000] Anne Eisenberg. In online auctions of the future, it'll be bot vs. bot vs. bot. *The New York Times*, 2000. August 17th.

[Fawcett, 1993] Tom Elliott Fawcett. Feature discovery for problem solving systems, PhD thesis, University of Massachusetts, Amherst, 1993.

[Fern *et al.*, 2004] Alan Fern, Robert Givan, Babak Falsafi, and T.N. Vijaykumar. Dynamic feature selection for hardware prediction, 2004. From `http://web.engr.oregonstate.edu/~afern/`.

[Fernandez and Veloso, 2006] F. Fernandez and Manuela Veloso. Learning by probabilistic reuse of past policies. In *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems*, 2006.

[Fidelman and Stone, 2007] Peggy Fidelman and Peter Stone. The chin pinch: A case study in skill learning on a legged robot. In Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorenti, and Tomoichi Takahashi, editors, *RoboCup-2006: Robot Soccer World Cup X*. Springer Verlag, Berlin, 2007. To appear.

[Fox *et al.*, 2004] Armando Fox, Emre Kiciman, David Patterson, Michael Jordan, and Randy Katz. Combining statistical monitoring and predictable recovery for self-management. In *Proceedings of 2004 Workshop on Self-Managed Systems (WOSS'04)*, October 2004.

[French and Messinger, 1994] Robert French and Adam Messinger. Genes, phenes and the Baldwin effect: Learning and evolution in a simulated population. *Artificial Life*, 4:277–282, 1994.

[Gat, 1995] Erann Gat. On the role of simulation in the study of autonomous mobile robots. In *AAAI-95 Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents.*, Stanford, CA, March 1995.

[Gat, 1998] Erann Gat. Three-layer architectures. In David Kortenkamp, R. Peter Bonasso, and Robin Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 195–210. AAAI Press, Menlo Park, CA, 1998.

[Genesereth and Love, 2005] Michael Genesereth and Nathaniel Love. General game playing: Overview of the AAAI competition. *AI Magazine*, 26(2), 2005.

[Goldberg, 1989] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1989.

[Gomez *et al.*, 2001] Faustino Gomez, Doug Burger, and Risto Miikkulainen. A neuroevolution method for dynamic resource allocation on a chip multiprocessor. In *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*, pages 2355–2361. IEEE, 2001.

[Gruau and Whitley, 1993] Frederic Gruau and Darrell Whitley. Adding learning to the cellular development of neural networks: Evolution and the Baldwin effect. *Evolutionary Computation*, 1:213–233, 1993.

[Gruau *et al.*, 1996] Frederic Gruau, Darrell Whitley, and Larry Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 81–89, 1996.

[Hinton and Nowlan, 1987] Geoffrey E. Hinton and Steven J. Nowlan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[Hornby *et al.*, 1999] G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata. Autonomous evolution of gaits with the Sony quadruped robot. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1297–1304, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann.

[Hornby *et al.*, 2000] G.S. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita. Evolving robust gaits with AIBO. In *IEEE International Conference on Robotics and Automation*, pages 3040–3045, 2000.

[Huber and Durfee, 1995] Marcus J. Huber and Edmund H. Durfee. Deciding when to commit to action during observation-based coordination. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 163–170, Menlo Park, California, June 1995. AAAI Press.

[Johnson, 2005] R. Colin Johnson. Steady pace takes DARPA race. *EE Times*, October 2005. Accessed at `http://www.eetimes.com`.

[Jong and Stone, 2005] Nicholas K. Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 752–757, August 2005.

[Kaminka *et al.*, 2003] Gal A. Kaminka, Pedro U. Lima, and Raul Rojas, editors. *RoboCup-2002: Robot Soccer World Cup VI*. Springer Verlag, Berlin, 2003.

[Kephart and Chess, 2003] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, pages 41–50, January 2003.

[Kim and Uther, 2003] Min Sub Kim and William Uther. Automatic gait optimisation for quadruped robots. In *Australasian Conference on Robotics and Automation*, Brisbane, December 2003.

[Kitano *et al.*, 1997a] Hiroaki Kitano, Yasuo Kuniyoshi, Itsuki Noda, Minoru Asada, Hitoshi Matsubara, and Eiichi Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.

[Kitano *et al.*, 1997b] Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda, and Minoru Asada. The RoboCup synthetic agent challenge 97. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 24–29, San Francisco, CA, 1997. Morgan Kaufmann.

[Kitano, 1998] Hiroaki Kitano, editor. *RoboCup-97: Robot Soccer World Cup I*. Springer Verlag, Berlin, 1998.

[Kohl and Stone, 2004a] Mate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2619–2624, 2004.

[Kohl and Stone, 2004b] Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2004.

[Koller, 2001] Daphne Koller. Representation, reasoning, learning, August 2001. IJCAI Computers and Thought Award talk.

[Kolter and Maloof, 2004] J.Z. Kolter and M.A. Maloof. Learning to detect malicious executables in the wild. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 470–478, New York, NY, 2004. ACM Press. Best Application Paper.

[Konidaris and Barto, 2006] George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd Internation Conference on Machine Learning*, pages 489–496, 2006.

[Kuhlmann *et al.*, 2005] Gregory Kuhlmann, Peter Stone, and Justin Lallinger. The UT Austin Villa 2003 champion simulator coach: A machine learning approach. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup-2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence*, pages 636–644. Springer Verlag, Berlin, 2005.

[Kuhlmann *et al.*, 2006a] Gregory Kuhlmann, Kurt Dresner, and Peter Stone. Automatic heuristic construction in a complete general game player. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 1457–62, July 2006.

[Kuhlmann *et al.*, 2006b] Gregory Kuhlmann, William B. Knox, and Peter Stone. Know thine enemy: A champion RoboCup coach agent. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 1463–68, July 2006.

[Kwok *et al.*, 2003] C. Kwok, D. Fox, and M. Meila. Adaptive realtime particle filters for robot localization. In *Proc. of the IEEE International Conference on Robotics & Automation*, 2003.

[Lagoudakis and Parr, 2003] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4(2003):1107–1149, 2003.

[Li *et al.*, 2006] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstractions for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pages 531–539, 2006.

[Liblit *et al.*, 2003] Ben Liblit, Alex Aiken, Alice X. Zheng, and Michael I. Jordan. Bug isolation via remote program sampling. In *Programming Languages Design and Implementation (PLDI)*, June 2003.

[Liblit *et al.*, 2005] B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan. Scalable statistical bug isolation. In *PLDI*, 2005.

[Mackworth, 1993] A. K. Mackworth. On seeing robots. In A. Basu and X. Li, editors, *Computer Vision: Systems, Theory, and Applications*, pages 1–13. World Scientific Press, Singapore, 1993.

[Maclin *et al.*, 2005] R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 2005.

[Mesnier *et al.*, 2004] Michael Mesnier, Eno Thereska, Gregory R. Ganger, Daniel Ellard, and Margo Seltzer. File classification in self-* storage systems. In *Proceedings of the First International Conference on Autonomic Computing*, May 2004.

[Minsky, 1988] Marvin L. Minsky. *The Society of Mind*. Simon & Schuster, 1988.

[Mitchell, 1983] Tom Mitchell. Learning and problem-solving. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, August 1983. Computers and Thought Award Paper.

[Montemerlo *et al.*, 2006] M. Montemerlo, S. Thrun, H. Dahlkamp, D. Stavens an, and S. Strohband. Winning the DARPA Grand Challenge with an AI robot. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Boston, MA, July 2006.

[Murray *et al.*, 2005] Joseph F. Murray, Gordon F. Hughes, and Kenneth Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning research*, 6:783–816, May 2005.

[Nardi *et al.*, 2005] Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors. *RoboCup-2004: Robot Soccer World Cup VIII*. Springer Verlag, Berlin, 2005.

[Newsome *et al.*, 2005] James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *The IEEE Symposium on Security and Privacy*, May 2005.

[Ng and Russell, 2000] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, 2000.

[Ng *et al.*, 2004] Andrew Y. Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry. Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2004. To Appear.

[Noda *et al.*, 1998] Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.

[Noda *et al.*, 2006] Itsuki Noda, Adam Jacoff, Ansgar Bredenfeld, and Yasutake Takahashi, editors. *RoboCup-2005: Robot Soccer World Cup IX*. Springer Verlag, Berlin, 2006.

[Nolfi *et al.*, 1994] Stefano Nolfi, Jeffery L. Elman, and Domenico Parisi. Learning and evolution in neural networks. *Adaptive Behavior*, 2:5–28, 1994.

[Pardoe and Stone, 2006] David Pardoe and Peter Stone. TacTex-2005: A champion supply chain management agent. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 1489–94, July 2006.

[Pardoe *et al.*, 2006] David Pardoe, Peter Stone, Maytal Saar-Tsechansky, and Kerem Tomak. Adaptive mechanism design: A metalearning approach. In *The Eighth International Conference on Electronic Commerce*, August 2006. To appear.

[Parker, 2002] Lynne E. Parker. Distributed algorithms for multirobot observation of multiple moving targets. *Autonomous Robots*, 12(3):231–255, 2002.

[Parkes, 2001] David C. Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, May 2001.

[Pell, 1993] Barney Pell. Strategy generation and evaluation for meta-game playing. PhD thesis, University of Cambridge, 1993.

[Phelps *et al.*, 2002] Steve Phelps, Peter Mc Burnley, Simon Parsons, and Elizabeth Sklar. Co-evolutionary auction mechanism design. In *Agent Mediated Electronic Commerce IV*, volume 2531 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2002.

[Polani *et al.*, 2004] Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors. *RoboCup-2003: Robot Soccer World Cup VII*. Springer Verlag, Berlin, 2004.

[Pormerleau, 1993] Dean A. Pormerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, 1993.

[Porta and Celaya, 2001] J. M. Porta and E. Celaya. Efficient gait generation using reinforcement learning. In *Proceedings of the Fourth International Conference on Climbing and Walking Robots*, pages 411–418, 2001.

[Quinlan *et al.*, 2003] Michael J. Quinlan, Stephen K. Chalup, and Richard H. Middleton. Techniques for improving vision and locomotion on the sony aibo robot. In *Proceedings of the 2003 Australasian Conference on Robotics and Automation*, December 2003.

[Quinlan *et al.*, 2005] Michael J. Quinlan, Steven P. Nicklin, Kenny Hong, Naomi Henderson, Stephen R. Young, Timothy G. Moore, Robin Fisher, Phavanna Douangboupha, and Stephan K. Chalup. The 2005 nubots team report. Technical report, The University of Newcastle, School of Electrical Engineering and Computer Science, 2005.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Ravindran and Barto, 2003] Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-Markov decision processes. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[Reynolds, 1999] Craig W. Reynolds. Steering behaviors for autonomous characters. In *Proceedings of the Game Developers Conference*, pages 763–782, 1999.

[Riley and Veloso, 2000] Patrick Riley and Manuela Veloso. On behavior classification in adversarial environments. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, 2000.

[Riley and Veloso, 2002] Patrick Riley and Manuela Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer Verlag, Berlin, 2002.

[Riley *et al.*, 2002] Patrick Riley, Manuela Veloso, and Gal Kaminka. An empirical study of coaching. In H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*, pages 215–224. Springer-Verlag, 2002.

[Roefer *et al.*, 2005] T. Roefer, R. Brunn, S. Czarnetzki, M. Dassler, M. Hebbel, M. Juengel, T. Kerkhof, W. Nistico, T. Oberlies, C. Rohde, M. Spranger, and C. Zarges. Germanteam 2005. In *RoboCup 2005: Robot Soccer World Cup IX, Lecture Notes in Artificial Intelligence*. Springer, 2005.

[Rofer *et al.*, 2003] T. Rofer, H.-D. Burkhard, U. Duffert, J. Hoffman, D. Gohring, M. Jungel, M. Lotzach, O. v. Stryk, R. Brunn, M. Kallnik, M. Kunz, S. Petters, M. Risler, M. Stelzer, I. Dahm, M. Wachter, K. Engel, A. Osterhues, C. Schumann, and J. Ziegler. Germanteam robocup 2003. Technical report, 2003.

[Rofer, 2004] T. Rofer. Evolutionary gait-optimization using a fitness function based on proprioception. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup-2004: Robot Soccer World Cup VIII*. Springer Verlag, Berlin, 2004.

[Rogers *et al.*, 1999] Seth Rogers, Claude-Nicolas Flechter, and Pat Langley. An adaptive interactive agent for route advice. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 198–205, Seattle, WA, USA, 1999. ACM Press.

[Russell, 1995] Stuart Russell. Rationality and intelligence. 1995. Computers and Thought Award Paper.

[Sandholm, 2003] Tuomas Sandholm. Making markets and democracy work: A story of incentives and computing. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1649–1671, 2003. Computers and Thought Award Paper.

[Schaeffer *et al.*, 1992] Jonathan Schaeffer, Joseph C. Culberson, Norman Treloar, Brent Knight, Paul Lu, and Duane Szafron. A world championship caliber checkers program. *Artificial Intelligence*, 53(2-3):273–289, 1992.

[Schonberg *et al.*, 1995] T. Schonberg, M. Ojala, J. Suomela, A. Torpo, and A. Halme. Positioning an autonomous off-road vehicle by using fused DGPS and inertial navigation. In *2nd IFAC Conference on Intelligent Autonomous Vehicles*, pages 226–231, 1995.

[Selfridge *et al.*, 1985] O. Selfridge, R. S. Sutton, and Andrew G. Barto. Training and tracking in robotics. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 670–672, 1985.

[Singh, 1992] Satinder P. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.

[Soni and Singh, 2006] Vishal Soni and Satinder Singh. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, July 2006.

[Sony, 2004] Sony. Aibo robot, 2004. `http://www.sony.net/Products/aibo`.

[Sridharan and Stone, 2005a] Mohan Sridharan and Peter Stone. Autonomous color learning on a mobile robot. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005.

[Sridharan and Stone, 2005b] Mohan Sridharan and Peter Stone. Real-time vision on a mobile robot platform. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, August 2005.

[Sridharan and Stone, 2005c] Mohan Sridharan and Peter Stone. Towards illumination invariance in the legged league. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup-2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence*, pages 196–208. Springer Verlag, Berlin, 2005.

[Sridharan and Stone, 2007] Mohan Sridharan and Peter Stone. Color learning on a mobile robot: Towards full autonomy under changing illumination. In *The 20th International Joint Conference on Artificial Intelligence*, January 2007. To appear.

[Stanley and Miikkulainen, 2002] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[Stone and Veloso, 1998] Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.

[Stone and Veloso, 2000a] Peter Stone and Manuela Veloso. Layered learning. In Ramon López de Mántaras and Enric Plaza, editors, *Machine Learning: ECML 2000 (Proceedings of the Eleventh European Conference on Machine Learning)*, pages 369–381. Springer Verlag, Barcelona,Catalonia,Spain, May/June 2000.

[Stone and Veloso, 2000b] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.

[Stone *et al.*, 2001a] Peter Stone, Tucker Balch, and Gerhard Kraetzschmar, editors. *RoboCup-2000: Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, Berlin, 2001.

[Stone *et al.*, 2001b] Peter Stone, Michael L. Littman, Satinder Singh, and Michael Kearns. ATTac-2000: An adaptive autonomous bidding agent. *Journal of Artificial Intelligence Research*, 15:189–206, June 2001.

[Stone *et al.*, 2003] Peter Stone, Robert E. Schapire, Michael L. Littman, János A. Csirik, and David McAllester. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. *Journal of Artificial Intelligence Research*, 19:209–242, 2003.

[Stone *et al.*, 2005] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[Stronger and Stone, 2006] Daniel Stronger and Peter Stone. Towards autonomous sensor and actuator model induction on a mobile robot. *Connection Science*, 18(2):97–119, 2006. Special Issue on Developmental Robotics.

[Sukthankar and Sycara, 2005] Gita Sukthankar and Katia Sycara. Automatic recognition of human team behaviors. In *Proceedings of Modeling Others from Observations (MOO), Workshop at the International Joint Conference on Artificial Intelligence (IJCAI)*. July 2005.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[Sutton *et al.*, 1999] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

[Sutton *et al.*, 2000] R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

[Sutton, 1988] Richard Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Sycara, 1998] Katia Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

[Tambe, 1996] M. Tambe. Tracking dynamic team activity. In *National Conference on Artificial Intelligence(AAAI96)*, 1996.

[Taylor *et al.*, 2005] Matthew E. Taylor, Peter Stone, and Yaxin Liu. Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005.

[Tesauro, 1994] Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[Thrun and Schwartz, 1995] Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*, 1995.

[Veloso *et al.*, 2000] Manuela Veloso, Enrico Pagello, and Hiroaki Kitano, editors. *RoboCup-99: Robot Soccer World Cup III*. Springer Verlag, Berlin, 2000.

[Vidal and Durfee, 1995] Jose M. Vidal and Edmund H. Durfee. Recursive agent modeling using limited rationality. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 376–383, Menlo Park, California, June 1995. AAAI Press.

[Walsh *et al.*, 2004] William E. Walsh, Gerald Tesauro, Jeffrey O. Kephart, and Rajarshi Das. Utility functions in autonomic systems. In *Proceedings of the First International Conference on Autonomic Computing*, May 2004.

[Watkins, 1989] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

[Weber, 1997] Robert J. Weber. Making more from less: Strategic demand reduction in the FCC spectrum auctions. *Journal of Economics and Management Strategy*, 6(3):529–548, 1997.

[Weiß, 1996] Gerhard Weiß. ECAI-96 workshop on learning in distributed artificial intelligence. Call For Papers, 1996.

[Wellman *et al.*, 2001] Michael P. Wellman, Peter R. Wurman, Kevin O'Malley, Roshan Bangera, Shou-de Lin, Daniel Reeves, and William E. Walsh. A trading agent competition. *IEEE Internet Computing*, 5(2):43–51, March/April 2001.

[Whiteson and Stone, 2003] Shimon Whiteson and Peter Stone. Concurrent layered learning. In Jeffrey S. Rosenschein, Tuomas Sandholm, Michael Wooldridge, and Makoto Yokoo, editors, *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 193–200, New York, NY, July 2003. ACM Press.

[Whiteson and Stone, 2006] Shimon Whiteson and Peter Stone. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7:877–917, May 2006.

[Whiteson *et al.*, 2005] Shimon Whiteson, Peter Stone, Kenneth O. Stanley, Risto Miikkulainen, and Nate Kohl. Automatic feature selection via neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, June 2005.

[Wildstrom *et al.*, 2005] Jonathan Wildstrom, Peter Stone, Emmett Witchel, Raymond J. Mooney, and Mike Dahlin. Towards self-configuring hardware for distributed computer systems. In *The Second International Conference on Autonomic Computing*, pages 241–249, June 2005.

[Witten and Frank, 1999] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, October 1999. `http://www.cs.waikato.ac.nz/ml/weka/`.

[Wurman *et al.*, 2001] P. R. Wurman, M. P. Wellman, and W. E. Walsh. A parameterization of the auction design space. *Journal of Games of Economic Behavior*, 35:304–338, 2001.

[Zhang and Vadakkepat, 2003] Ruixiang Zhang and Prahlad Vadakkepat. An evolutionary algorithm for trajectory based gait generation of biped robot. In *Proceedings of the International Conference on Computational Intelligence, Robotics and Autonomous Systems*, Singapore, 2003.