Proceedings of the AAAI-97 Workshop on On-line Search, Providence, Rhode Island, July 1997.

A New Heuristic Approach for Dual Control

Juan C. Santamaría and **Ashwin Ram**
Georgia Institute of Technology
Atlanta, GA 30332-0280
{carlos — ashwin}@cc.gatech.edu

Abstract

Autonomous agents engaged in a continuous interaction with an incompletely known environment face the problem of *dual control* [Fel'dbaum 1965]. Simply stated, actions are necessary not only for studying the environment, but also for making progress on the task. In other words, actions must bear a "dual" character: They must be investigators to some degree, but also directors to some degree. Because the number of variables involved in the solution of the dual control problem increases with the number of decision stages, the exact solution of the dual control problem is computationally intractable except for a few special cases. This paper provides an overview of dual control theory and proposes a heuristic approach towards obtaining a near-optimal dual control method that can be implemented. The proposed algorithm selects control actions taking into account the information contained in past observations as well as the possible information that future observations may reveal. In short, the algorithm anticipates the fact that future learning is possible and selects the control actions accordingly. The algorithm uses memory-based methods to associate long-term benefit estimates to belief states and actions, and selects the actions to execute next according to such estimates. The algorithm uses the outcome of every experience to progressively refine the long-term benefit estimates so that it can make better, improved decisions as it progresses. The algorithm is tested on a classical simulation problem.

Introduction

Autonomous agents are intelligent systems engaged in an on-going interaction with the environment in which they are embedded. They are capable of sensing and influencing the environment by making decisions sequentially during their entire "life". Thus, the problem the agent confronts is that of autonomous sequential decision-making: at any given situation the agent must decide what to action to perform next, execute the action, observe the situation that results, and repeat the whole cycle at the new situation. Additionally, after executing every action, the agent should learn from the outcome so that it can make better, improved decisions as it progresses.

Commonly, autonomous agents are designed so that they select actions based on their current knowledge level. However, their decision-making mechanism does not explicitly consider that their knowledge level may change as they perform the task. That is, an agent uses its current level of knowledge to select and execute actions that are efficient from the perspective of such imperfect knowledge. Often, some type of random exploration (e.g., Boltzmann exploration [Watkins 1989]) or exploration bonuses (e.g., [Sutton 1990]) bypasses the standard action-selection mechanism so that the agent is able to learn the effect of "unseen" actions. The agent learns to perform better each time because new incoming information is used to improve the knowledge level, which leads to better decisions. However, this is a *passive* form of learning because the agent does not explicitly select actions to gain new information or consider how each action will influence its level of knowledge in the short- or the long-term. This contrast the more efficient and robust form of *active* learning in which the agent selects actions taking into account both their efficiency with respect to the task and the influence on the current level of knowledge their execution may produce [Ram & Leake 1995].

The problem of sequential decision making under incomplete knowledge has been studied by researchers in the field of optimal control for systems with continuous state and actions ([Bertsekas 1995; Fel'dbaum 1965; Bar-Shalom 1990; Stengel 1994]) and reinforcement learning for systems with discrete states and actions ([Parr & Russell 1995; Cassandra, Kaelbling, & Littman 1994; Jaakkola, Singh, & Jordan 1995]). When the agent has incomplete knowledge, actions are necessary not only for studying the environment, but also for making progress on the task. Actions must bear a "dual" character: They must be investigators to a known degree, but also directors to a known degree. The concept of *dual control* was first introduced by Fel'dbaum to denote the control of actions that bear this dual character [Fel'dbaum 1965]. In dual control a conflict arises between the two aspects of the control action mentioned above. In fact, we can control successfully only when the properties of the environment are known accurately enough. Meanwhile the study of the environment requires time. Too "hurried" an agent will perform unjustified control actions, which will not by substantiated properly by the information obtained as a result of the study of the environment. Too "cautious" an agent will wait for an unnecessarily long time, accumulating information, and will not be able to accomplish the task to the required specifications. Thus, an agent must select actions in such a way that it can achieve a balance in the dual character of its actions. In other words, the agent has to devote some effort to *exploring* the environment and

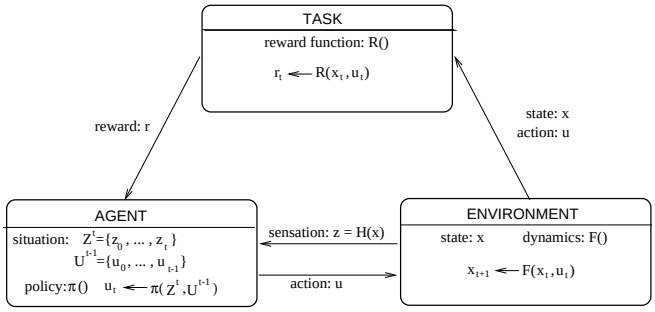

Figure 1: Autonomous Sequential Decision-Making. The framework consists of three main elements shown in boxes: the agent, the environment, and the task. The arrows indicate the flow of information among the elements.

gather new knowledge, but only to the extent that it can start *exploiting* such knowledge to proficiently perform the task.

The exact optimal solution of the dual control problem is computationally intractable. However, we propose a new heuristic solution to the problem that explicitly considers the dual nature of the control. In other words, a heuristic control algorithm that anticipates the fact that future learning is possible.

Sequential Decision Tasks

This section presents the formulation and solution of the optimal control problem for sequential decision tasks in the context of incomplete knowledge, as well as the difficulties associated with the optimal solution method. These difficulties motivate the heuristic solution method presented in the following section.

Definition

The framework for studying autonomous sequential decision-making consists of three main elements: the agent, the environment, and the task. The agent attempts to control the environment by perceiving the environment and choosing actions in a sequential fashion. The environment is a dynamic system characterized by a *state* and its *dynamics*, a function that describes the evolution of the state given the agent's actions. A task is a declarative description of the desired behavior the agent should exhibit as it interacts with the environment. A task is defined by associating a scalar value or *reward*, which is delivered to the agent every time it executes an action. The ultimate goal of the agent is to learn a strategy for selecting actions or *policy* such that the expected (possibly discounted) sum of rewards is maximized. Thus, the task defines "what" the agent should do but it is the agent's responsibility to find out "how". Figure 1 shows these three main elements and their interaction. The next subsections describe in more detail each component of the framework.

The Environment The environment is a discrete-time dynamic system described by the dynamics function

$$x_{t+1} = F(x_t, u_t) \tag{1}$$

where $x_t \in \Re^n$ and $u_t \in \Re^p$ are the state of the system and the agent's controlling action at time t, respectively. The state consists of a set of variables that completely characterize the current situation of the system and, together with the agent's subsequent decisions, fully determines the evolution of the system. The dynamics is a vectorial function that maps the current state of the system to the next state under the influence of the agent's current action.

The Task The task is characterized by a reward function, $r_{t+1} = R(x_t, u_t)$, that associates a scalar to every action the agent may execute at any given state and a performance measure given by

$$V(x_t) = E\left[\sum_{t=0}^{\infty} \gamma^t R(x_t, u_t)\right] \tag{2}$$

where γ is the *discount factor* ($0 < \gamma < 1$) and $V(x_t)$ represents the long-term benefit, or *value* of state x_t given the agent's actions $u_t, u_{t+1}, \cdots$. The ultimate goal of the agent is to learn a strategy for selecting actions (i.e., a policy) in order to maximize the performance measure (i.e., discounted sum of rewards) subject to the constraints given by Equation 1. Thus, the reward function is a declarative description of the desired behavior of the agent.

The Agent The agent implements the decision-maker in the framework. It perceives the state of the environment through the perception function

$$z_t = H(x_t) \tag{3}$$

where $z_t \in \Re^m$ is the *sensation* associated with state x_t. The agent attempts to control the environment using admissible policy functions of the feedback type

$$\begin{aligned} u_t &= \pi(Z^t, U^{t-1}) \\ Z^t &\equiv \{z_0, \cdots, z_t\} \\ U^{t-1} &\equiv \{u_0, \cdots, u_{t-1}\} \end{aligned} \tag{4}$$

where Z^t and U^{t-1} represent all the information the agent is aware of since the beginning of the task.

The policy functions described by (Equation 4) have two important characteristics. First, the control actions do not anticipate the future because any decision at a given time depends exclusively on current and/or past information. Second, the control actions at a given time take advantage of *all* the information available at that time.

The objective of the agent is to find a policy of the form given by (Equation 4) that maximizes the performance measure given by (Equation 2). Such an optimal policy if found, is able to satisfy the conflict between two apparently opposing tendencies. On one hand, decisions made with incomplete knowledge tend to decrease the performance measure. Thus, in order to reduce the decrease in the cumulative reward, the agent tends to be "cautious," a property known in the decision theory literature as "risk aversion" [DeGroot 1970]. On the other hand, since this is a multistage decision process where a new sensation is collected at each stage, the agent may be able to carry out what has been called "probing" of the environment to enhance the incomplete knowledge [Bar-Shalom 1990; Fel'dbaum 1965].

Heuristic Solution to Dual Control

This section formulates the optimal control problem for sequential decision tasks in the context of incomplete knowledge and describes a novel heuristic approach that attempts to circumvent the difficulties associated with the exact optimal solution. The derived algorithm is suboptimal but possesses the active learning characteristic because it explicitly considers the short- and long-term effects each action produces on the current state of the environment and level of knowledge in the agent. This will provide the basis for an implementation of the approach that is applicable when the agent knows that the unknown properties of the environment is one out of a set of N possible values.

The exact optimal solution to the dual control problem is extensive and involved (see, for example, [Meier 1965]); therefore we shall only outline the derivation and summarize the results below.

Two important concepts in the derivation of the optimal policy are the *information state* and the *optimal action-value function*. The information state, I_t, can be viewed as a quantity that is equivalent to all a priori knowledge of the system, and the observation process Z^t and U^{t-1} in describing the future evolution of the system [Stengel 1994]. The information state is important because it captures all the agent knows about the environment at a given point in time as well as all the predictions about the evolution of the environment the agent can infer from that knowledge base. The optimal action-value function, $Q^*(I_t, u_t)$, maps every information state and action combination to the best performance measure the agent can receive from such information state when it applies the given action and follows the optimal policy thereafter. The optimal action-value function is important because it is used to guide the search for the optimal policy. Specifically, the agent can use an explicit representation of the optimal action-value function to perform the following one-stage lookahead search

$$u_t^* = \arg\max_{u_t} E\left[Q^*(I_t, u_t)\right] \tag{5}$$

The optimal action-value function is defined as

$$\begin{aligned} Q^*(I_t, u_t) &\equiv \max_{u_{t+1}, u_{t+1}, \cdots} E\left[\sum_{k=t}^{\infty} \gamma^k R(x_k, u_k) \mid I_t\right] \\ &= E\left[R(x_t, u_t) + \gamma Q^*\left(I_{t+1}, u_{t+1}^*\right) \mid I_t\right] \end{aligned}$$

where the recursive definition follows from the principle of optimality [Bellman 1957], which can be stated as follows: at any stage, whatever the present information and past decisions, the remaining decisions must constitute an optimal policy with regard to the current information state.

The main difficulties in implementing the exact optimal control law for the dual control problem are: (1) the information state is either infinite dimensional or finite, but grows with time; (2) the optimal action-value function associated with the information state is generally not an explicit function; and (3) the computation required to find the optimal action-value function is an expensive process (see [Bar-Shalom 1990]). Thus, a reasonable suboptimal approach would be to: (1) approximate the information state such that the dimension of its space stays constant for all time; (2) approximate the optimal action-value function with a function approximator; and (3) asymptotically learn the optimal action-value function using a computationally inexpensive stochastic approximation procedure. The following subsections described in more detail each of these issues as well as the proposed dual control algorithm.

Belief State

The information state $I_t = \{Z^{t+1}, U^t\}$ is a quantity that completely characterize all the knowledge the agent has at a particular time. Since the information state captures all the information the agent is aware of since the beginning of the task, the quantity is either infinite dimensional or finite, but grows with time. A reasonable approximation of the information state should consist of a finite dimensional quantity that can, at the best of abilities, summarize the past and characterize the future evolution of the system. We shall call this quantity the *belief state* of the agent. The belief state exists inside the agent and explicitly represents the current state of the environment (or its best estimate based on past sensations) and the uncertainty or knowledge level of the unknown properties of the system. Additionally, the belief state drives the agent's action-selection mechanism. Thus, when an agent is making a decision it is considering not only the immediate reward and change in the state of the environment such decision would produce, but also the possible change in what it knows about the partially known properties of system.

To simplify the discussion, we will assume that the belief state, s_t, is composed of two separate finite dimensional quantities: an estimate of the state of the environment, $\hat{x}_t$, and a suitable representation of the knowledge level for the unknown properties, κ_t[1]. In other words,

$$s_t = \{\, \hat{x}_t, \kappa_t \,\}$$

The vector-valued variable, $\hat{x}_t$, represents the agent's best estimate of the state of the environment at stage t based on a priori knowledge and previous observations. The variable, κ_t, represents the agent's partial information about the unknown characteristics of the system at stage t and measures the amount of study of the system. For example, this variable may contain probability distributions of the unknown characteristics and, as the agent collects information, the a posteriori probability distributions gradually close to the actual characteristics of the system. The distinctive property of dual control is that the rate of change in κ_t depends on the strategy of the agent.

The belief state plays the same role as the information state. The agent uses the belief state to select and execute an action based on a given policy $u_t = \pi(s_t)$. Then, it observes the incoming sensation (z_{t+1}) and constructs the new belief state using a recursive estimation equation based on the Bayes' rule or any other recursive relation of the type

$$\begin{aligned} s_{t+1} &= \{\hat{x}_{t+1}, \kappa_{t+1}\} \\ &= \{\Psi_x(\hat{x}_t, \kappa_t, u_t, z_{t+1}), \Psi_\kappa(\kappa_t, \hat{x}_t, u_t, z_{t+1})\} \\ &= \Psi(s_t, u_t, z_{t+1}) \end{aligned} \tag{6}$$

[1] The commitment of representing the belief state as two separate quantities is not a requisite of the proposed approach. However, the discussion is easier to follow when such commitment is made because it can be clearly seen the role each of the quantities plays.

where $\Psi_x(\cdot)$ and $\Psi_\kappa(\cdot)$ denote the recursive estimation functions for the state and knowledge level respectively. The details of the estimation functions depend on the representation of the belief state and the a priori knowledge of the agent. A particular implementation will be described in detail in the next section.

Additionally, the belief state serves the one-stage lookahead search by allowing the agent to predict the long-term benefit associated with different actions. The best action is the one that maximizes the predicted long-term benefit.

In summary, the belief state is a heuristic representation of the information state using a finite and constant number of dimensions. It is a quantity that exists in the agent and, at the best of abilities, summarize the past and characterize the future evolution of the system. The agent gradually updates the belief state based on the actions it executes and the sensations it perceives using a recursive estimation procedure. The agent can use the belief state in conjunction with the optimal action-value function to select the best action by performing a one-stage lookahead search. The next subsection discusses the topic of representing value functions.

Action-Value Function Approximators

The objective of the optimal action-value function is to map any information state and action combination to the best expected performance measure the agent can receive from such information state when it executes the given action and follows the optimal policy thereafter. An explicit representation of the optimal value function is important because it enables the agent to search and select the best action at any given stage (Equation 5). However, a closed-form solution of the optimal value function is rarely available in most problems.[2] A reasonable approach for explicitly representing the optimal action-value function is to use a function approximator on the belief state. Function approximators have been used successfully in the past in other reinforcement learning tasks (e.g.,[Santamaría, Sutton, & Ram 1996; Sutton 1996; Atkeson 1993; Rummery & Niranjan 1994]). They use finite resources to represent estimates of the long-term value associated with continuous real-valued variables. Additionally, they have parameters that the agent can use to adjust the value estimates.

A function approximator for the action-value function is of the form $\hat{Q}_w(s,u)$, where s is the belief state, u is the action, and w is a set of adjustable parameters or *weights*. The function approximator provides the agent with an initial estimate of the long-term benefit associated to each belief state and action combination. Thus, the idea consists of letting the agent decide the best action to execute next by performing the one-stage lookahead search (Equation 5) with the function approximator, and then using the outcome of each action to asymptotically improve the estimates towards their optimal values. One advantage of function approximators is that they can generalize the long-term benefit estimates associated with belief states and action combinations the agent actually experiences to other regions of the belief state and action spaces. In this way, the agent can estimate the long-term benefit of belief states and actions pairs it has never experienced before. Also, function approximators are able to represent the optimal action-value function even when the variables in the belief state are continuous (i.e., real-valued variables).

The one-stage lookahead algorithm selects the best action the agent can execute next given current approximations of the belief state and action-value functions. This follows because the algorithm adheres to the principle of optimality's statement that requires that at every stage, "the remaining decisions must constitute an optimal policy with regard to the current information set." Thus, assuming the function approximator matches closely the optimal value function, the algorithm selects the action that maximizes the expected sum of the immediate reward and the (properly discounted) estimated performance measure of the belief state that would result given the current belief state. Given the limitations of the finite dimensional belief state and function approximator, no other action produces better performance measure at the given stage.

In summary, function approximators can be used to explicitly represent the optimal action-value function. They use finite resources to represent an estimate of the performance measure the agent can achieve from any given belief state. The initial estimates of the function approximator may not be the optimal ones, but the agent can use the outcome of each action it executes to asymptotically improve the estimates toward their optimal values. The agent is able to accomplish this by adjusting a set of weights associated with the function approximator. Additionally, the agent can use the one-stage lookahead algorithm to select the best action it can execute next. The next subsection discusses the topic of improving the estimates of the value function.

Computation

Finding the optimal action-value function is a computationally expensive process. A reasonable approach is use some stochastic approximation procedure (e.g., temporal difference methods [Sutton 1988]) to asymptotically improve the estimates towards their optimal values. The main idea consists of using the agent's experience to progressively learn the optimal action-value function. More specifically, the agent can use a function approximator containing an initial estimate of the value function (random estimates will suffice) and decide the best action to execute next by performing the one-stage lookahead search (Equation 5). Then, after observing the outcome of executing each action, the agent can use the stochastic approximation procedure to improve the estimate of the action-value function by adapting the weights of the function approximator. Thus, the agent can incrementally learn the optimal action-value function by continually exercising the current, non-optimal estimate of the action-value function and improving such estimate after every experience. This approach has been extensively and successfully used in reinforcement learning [Sutton 1996; Santamaría, Sutton, & Ram 1996;

[2] A notable exception are the Linear dynamics, Quadratic reward, Gaussian forced (LQG) regulators that have a quadratic optimal value function of the form $V(\hat{x}) = \hat{x}^T P \hat{x} + c$, where P is a $n \times n$ positive definite symmetric matrix and c is a positive constant (see, for example, [Stengel 1994]).

Watkins 1989; Rummery & Niranjan 1994].

According to the principle of optimality, the value function must satisfy the recursive relation (so called the Bellman's Equation, e.g. [Bertsekas 1995])

$$\hat{Q}_w(s_t, u_t) = E\left[R(x_t, u_t) + \gamma \hat{Q}_w\left(s_{t+1}, u^*_{t+1}\right) \mid s_t\right] \quad (7)$$

where s_t and u_t are the belief state and action at stage t. Stochastic approximation procedures exploit this recursive relation to create an update formula the agent can use to adapt the weights of the function approximator after every experience. More specifically, every time the agent selects action u_t and observes the next sensation z_{t+1} and reward r_{t+1}, it can verify whether Equation 7 holds or not by computing the error between the two consecutive predicted performance values; before executing the action, $\hat{Q}_w(s_t, \hat{u}^*_t)$, and after executing the action, $r_{t+1} + \gamma \hat{Q}_w(s_{t+1}, \hat{u}^*_{t+1})$. When the error is different from zero, the agent uses the update formula to adapt the weights w and improve the estimate of the action-value function at s_t.

The procedure above described can be efficiently implemented using temporal difference methods ([Sutton 1988]). Sutton defines a whole family of update formulas for temporal difference methods called TD(λ), where $0 \leq \lambda \leq 1$ is a parameter used to measure the relevance of previous predictions in the current error.

An agent improves the current estimate of $\hat{Q}_{w_t}(s_t, \hat{u}^*_t)$ by adjusting the weights after every belief state transition using TD(λ) updates. For this purpose, every time the agent selects and executes action $\hat{u}^*_t$ at belief state s_t, observes the next sensation z_{t+1} and reward r_{t+1} that results from executing that action, and constructs the next belief state $s_{t+1} = \Psi(s_t, \hat{u}^*_t, z_{t+1})$, it uses the following TD(λ) formula to update the weights of the function approximator,

$$\Delta w_t = \alpha\left(r_{t+1} + \gamma \hat{Q}_{t+1} - \hat{Q}_t\right) \sum_{k=0}^{t} (\lambda\gamma)^{t-k} \nabla_{w_k} \hat{Q}_k \quad (8)$$

where $\hat{Q}_{t+1} = \hat{Q}_{w_t}(s_{t+1}, \hat{u}^*_{t+1})$, $\hat{Q}_t = \hat{Q}_{w_t}(s_t, \hat{u}^*_t)$, $\nabla_{w_k}\hat{Q}_k$ is the gradient of $\hat{Q}(\cdot)$ with respect w and evaluated at the belief state s_k, and α is a learning rate. The interpretation of Equation 8 is as follows: $\hat{Q}_t$ and $\hat{Q}_{t+1}$ represent the current estimate of the long-term benefit (i.e., performance measure) at states s_t and s_{t+1} respectively. The term $\left(r_{t+1} + \gamma \hat{Q}_{t+1} - \hat{Q}_t\right)$ represents the error incurred by $\hat{Q}_t$ in predicting the future according to $\hat{Q}_{t+1}$. In other words, the value of belief state s_t should be equal to the immediate reward r_{t+1} plus the value of the next belief state s_{t+1} properly discounted. In case of error, the weights are proportionally modified in the direction of the gradient ∇_{w_t} in order to maximally reduce error. The discounted sum of the previous gradients ∇_{w_k} are also credited for the current error although their influence decays exponentially with λ.

The convergence of TD(λ) has been proved under different conditions and assumptions. Watkins [Watkins 1989] shows that estimates asymptotically converge to their optimal values in systems having discrete and finite state and action spaces when TD(0) is used to perform the updates.

1. **Initially**:
   Set the initial estimate of the value function: w_0
   Construct the initial belief state:
   $s_0 = \{\hat{x}_0, \kappa_0\}$
   based on the initial sensation z_0.
2. **Perform**:
   At each stage $t = 0, 1, \cdots$
   (a) **Control**:
   Select and execute the best action at stage t:
   $\hat{u}^*_t$, (Equation 5).
   (b) **Perception**:
   Observe the resulting reward and sensation
   $r_{t+1} = R(x_t, \hat{u}^*_t)$
   $z_{t+1} = H(x_{t+1})$.
   (c) **Belief State Estimation**:
   Update the belief state based on the observed outcome:
   $s_{t+1} = \Psi(s_t, \hat{u}^*_t, z_{t+1})$
   (d) **Action-Value Function Estimation**:
   Update the action-value function using TD(λ):
   w_{t+1}, (Equation 8).

Figure 2: Heuristic Solution Algorithm.

A condition for the convergence is that all states are visited and all actions are executed infinitely often. Tsitsiklis and Van Roy [Tsitsiklis & Van Roy 1996] shows that the value function associated with a given policy converges for any linear function approximator and TD(λ) updates. There is no proof showing the convergence of TD(λ) for more complex function approximators, but this has not stopped researchers for trying these methods using different classes of non-linear function approximators. Successful results have been obtained with multi-layer neural networks (e.g., [Lin 1992], [Rummery & Niranjan 1994]) and sparse coarse coding methods such as Cerebellar Model Articulation Controllers (*CMACs*) (e.g., [Santamaría, Sutton, & Ram 1996]) and memory-based function approximators (e.g., [Santamaría, Sutton, & Ram 1996]).

An efficient on-line implementation of the update rule is possible using "eligibility traces" (for details, see, [Singh & Sutton 1996]). The idea is to maintain the value of the rightmost sum in Equation 8 in a variable or *eligibility trace*, which can be easily updated at every stage using a recursive relation. The agent performs the update every time it executes an action. Thus, the method is computationally efficient since each update is computationally inexpensive. On the other hand, the data efficiency of the method can be very low since the agent requires large amounts of data (i.e., the agent must execute many actions) to make the value function estimates converge to their optimal values.

Summary

The heuristic solution to the dual control problem integrates the control (Equation5), the estimation (Equation 6, and the computation procedures (Equation 8). Figure 2 shows the algorithm.

Results

This section describes the results of using the multiple hypothesis implementation of the heuristic dual control ap-

proach in the double integrator problem. This problem consists of a linear dynamics system with quadratic costs (i.e., negative rewards) that depend on the state and action values. There is one unknown parameter in the system and the agent must learn to execute actions that help estimate the unknown parameter while minimizing the sum of costs. The function approximator used in the experiments is the instance-based and the computation method is temporal difference (Equation 8). A description of the implementation of this type of function approximator can be found in [Santamaría, Sutton, & Ram 1996].[3]

The instance-based function approximator uses cases to represent the action-value function. Each case represents a point in the belief state space and holds its associated long-term benefit estimate. The density threshold and the smoothing parameters were set to $\tau_d = 0.1$ and $\tau_k = 0.1$ respectively. The similarity metric was the Euclidean distance and the kernel function was the Gaussian. This produces cases with spherical receptive fields and blending of the value function using a small number of cases. The values of each case were updated using temporal difference (Equation 8) and implemented using eligibility traces following [Santamaría, Sutton, & Ram 1996]. The values for the free constants were $\gamma = 0.99$, $\lambda = 0.8$, $\alpha = 0.5$, and $\epsilon = 0$. The one-stage lookahead search was performed using 25 equally spaced values for the acceleration between the minimum value, $a_{\min} = -1$, and the maximum value, $a_{\max} = +1$ (i.e., $25\Delta_u = a_{\max} - a_{\min}$).

Double Integrator

The double integrator is a system with linear dynamics and two dimensional state. It represents a car of a given mass moving in a flat terrain and subject to the application of a single force (see Figure 3). The state of the system consists of the current position, p, and velocity v of the car. The action is the acceleration, a, applied to the system. The objective is to move the car from a given starting state to the origin (i.e., $p_d = 0$, $v_d = 0$) such that the sum of the rewards is maximized. The reward function is a negative quadratic function of the difference between the current and desired position and the acceleration applied, $r_{t+1} = -(p_t^2 + a_t^2)$. The reward function penalizes the agent more heavily when the distance between the current and desired states is large and also when the action applied is large. This type of reward function is widely used in robotic applications because it specifies policies that drive the system to the desired state quickly while keeping the size of the driving control small in magnitude. This formulation is standard in optimal control theory ([Stengel 1994; Narendra & Annaswamy 1989]), in which the objective is to minimize costs instead of maximize rewards. However, both formulations are mathematically equivalent.

[3]The source code used to perform all the experiments is in a compressed tar file available through anonymous ftp at the following URL:

ftp://ftp.cc.gatech.edu/pub/ai/students/carlos/RLI/active-learning.tar.gz

The source code is in C++ and follows the standard software interface for reinforcement learning problems developed by Richard S. Sutton and Juan C. Santamaría. The documentation for the standard software interface can be found in URL:

http://envy.cs.umass.edu/People/sutton/RLinterface/RLinterface.html

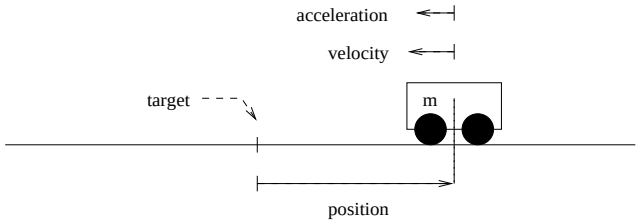

Figure 3: Double integrator. A car moving in a flat terrain subject to a the application of a single force.

The dynamics of the double integrator is very simple and it is described by the following equation,

$$M\frac{d^2p}{dt^2} = a$$

where M is the mass of the car, which is unknown to the agent. The acceleration is bounded to be in the range between the minimum and maximum acceleration values (i.e., $a \in [a_{\min}\ a_{\max}]$, where $a_{\min} = -1$ and $a_{\max} = 1$).

The simulation used a time step of $\Delta t = 0.05$ seconds and new control actions were selected every four time steps. A trial consists of starting the system at position $p = 1$ with velocity $v = 0$ and running the system until either 200 decision steps had elapsed (i.e. 40 simulated seconds) or the state gets out of bounds (i.e., when $|p| > 1$ or $|v| > 1$), whichever comes first. In the latter case, the agent receives a negative reward (i.e., a punishment) of 50 units to discourage it from going out of bounds. An experiment consists of 36 replications of 50 trials each, measuring the number of time steps and cumulative cost (i.e., negative sum of rewards) for each replication after each trial. The average number of time steps and cumulative cost across replications are used as measures of performance. At each trial the agent is given a light car ($M_1 = 1$) or a heavy car ($M_2 = 5$) with 50% probabilities respectively. The agent can perceive the position and velocity of the car but it cannot perceive the mass.

Multiple Hypotheses Scenario

In this experiment the mass of the car may be either light or heavy. The agent is able to directly observe the position and velocity of the car but not its mass; however, it is able to indirectly determine the value of the mass by applying specific actions at specific states. The ultimate goal of the agent is to maximize the performance measure. Thus, it must select a sequence of actions that best reveal the value of the mass while maximizing the sum of rewards. The belief state consists of the state of the environment, which the agent can observe perfectly; and the knowledge level, which is a probability variable representing the belief or *hypothesis* that the car is the light one. This experiment can be expanded easily to the case of N hypotheses by including $N-1$ probability variables representing each one of the $N-1$ possible hypotheses.

The belief state consists of the state of the car and the probability the agent is driving the light car, $s_t = \{p_t, v_t, q_t^{\text{light}}\}$. A new belief state results at every stage after the agent executes and action and receives a new sensation. The incoming sensation and Bayes' rule are used

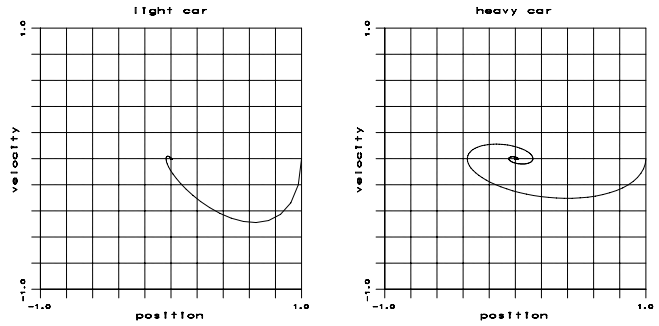

Figure 4: Trajectories in state space using the optimal control policy for the light car on both cars. Left: light car. Right: heavy car.

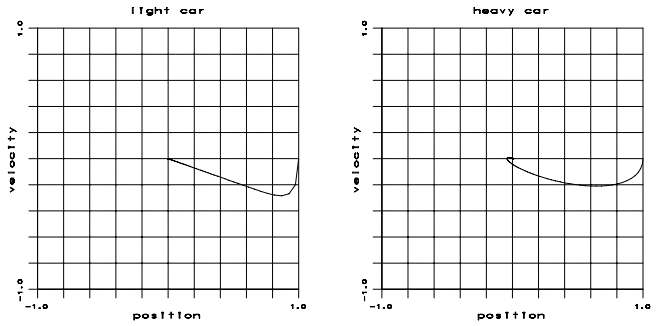

Figure 5: Trajectories in state space using the optimal control policy for the heavy car on both cars. Left: light car. Right: heavy car.

to construct the new belief state.

$$s_{t+1} = \Psi(s_t, a_t, z_{t+1}) = \{\, p_{t+1}, v_{t+1}, q_{t+1}^{\text{light}} \,\}$$

where

$$q_{t+1}^{\text{light}} = q_t^{\text{light}} \frac{P(\, z_{t+1} = F_{\text{light}}(p_t, v_t, a_t)\,)}{\sum_{k=\text{light / heavy}} P(\, z_{t+1} = F_k(x_t, u_t)\,) q_t^k}$$

The quantity $P(\, z_{t+1} = F_{\text{light / heavy}}(p_t, v_t, a_t)\,)$ represents the conditional probability of the light / heavy dynamics function in generating the perceived sensation z_{t+1} when the state of the car is $\{p_t, v_t\}$ and the action is a_t. Bayes' rule updates the probability by generating the a posteriori probabilities (q_{t+1}^{light}) based on the a priori probabilities (q_t^{light}) and the conditional probabilities ($P(z_{t+1} \mid k) = P(\, z_{t+1} = F_k(p_t, v_t, a_t)\,)$ (see, for example, [Ross 1993]).

The optimal control for the light car is different from the optimal control of the heavy car. The light car responds quickly to moderate forces; therefore the agent will not incur into large costs by applying top acceleration at the beginning of the task. On the other hand, the heavy car responds slowly to moderate forces, therefore the agent will not incur into large costs for not moving the car fast enough at the beginning of the task. Figures 4 and 5 show the trajectory in state space of both cars under the control of the optimal policies for the light and the heavy car respectively. The optimal policy for the light car overshoots when applied to the heavy car. Conversely, the optimal policy for the heavy car undershoots when applied to the light car. The optimal control for the case in which the agent does not know the mass of the car requires a compromise between exploration and exploitation.

In this experiment the agent knows that car (M) may be light (M_1) or heavy (M_2) but it does not know which one until it starts applying forces and observe the outcomes (i.e., a two hypotheses scenario). The agent may decide to apply a strong force all the way until the car is near the goal. However at some point it must decide to deaccelerate so that the car comes to a complete stop near the goal. When the agent is controlling with the heavy car, then it should start deaccelerating earlier than when it is controlling with the light car. For that purpose, the agent must try to identify which car it is controlling, but to be successful it must do so as it approaches the goal. The results shown below corresponds to the agent learning the dual control of the car during 200 trials of 200 stages each.

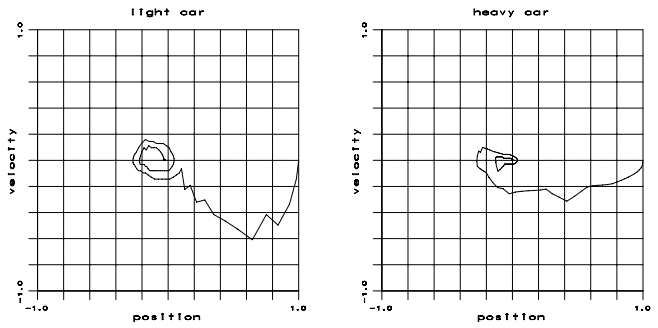

Figure 6: Trajectories in state space using the proposed dual control algorithm on both cars after 200 trials. Left: light car. Right: heavy car.

Figure 6 shows the trajectory in state space of both cars under the control of the proposed dual control algorithm. In both cases the agent learns to apply actions that collect new information while making progress on the task.

Figure 7 shows the plots of the agent's belief of the light car as a function of the stage after 200 trials for both cars.

Conclusions

This paper describes a heuristic approach for obtaining a control algorithm that exhibits the dual characteristic of appropriately distributing the control energy for learning and control purposes. The approach is an approximation based on the principle of optimality that uses finite storage resources to represent the information state and the action-value function, and finite computational resources to compute the value function. The control algorithm posses the distinguishing characteristic of regulating its learning as required by the performance measure and it perform this by considering the three consequences each action produce: the immediate reward, the next state in the environment, and the next knowledge level in the agent. The best action is the one that maximizes the sum of the immediate reward and the discounted value of the next state and knowledge

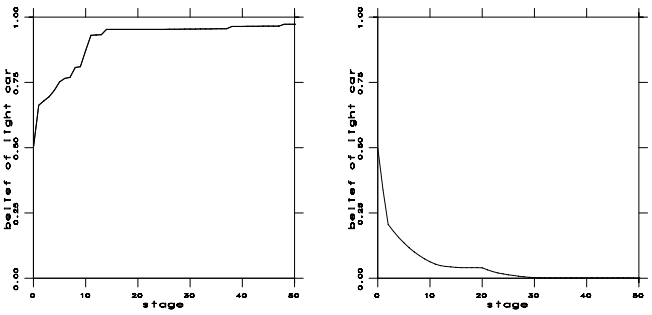

Figure 7: Hypothesis of light car after 200 trials on both cars. Left: light car. Right: heavy car.

level taken together. Some actions may change the state and leave the same knowledge level. The resulting state may presumably be a good one, but if the resulting knowledge level cannot confirm this hypothesis then their combined value may not be good. Similarly, other actions may change the knowledge level and leave the same state of the environment. The agent may now have better knowledge of the situation but the lack of progress may cause poor task performance. The optimal decision is the one that best compromises between the progress on the task and the gain of knowledge. However, the agent's only objective is to maximize the discounted sum of rewards. Thus, knowledge gathering actions are considered only with respect to this objective. This means that if the agent decides not to make much progress by collecting information at some point during the task it is only because such new knowledge will help the agent to make better, improved progress later in the task. A simple example is used to demonstrate the computational feasibility of the algorithm and its performance level when applied to a specific problem, and to provide some insight into the dual control theory.

Acknowledgements

The authors wish to thank Chris Atkeson for his comments and insightful discussions on the dual control problem.

References

Atkeson, C. G. 1993. Local trajectory optimizers. *Advances in Neural Information Processing Systems* 6.

Bar-Shalom, Y. 1990. Stochastic dynamic programming: Caution and probing. *IEEE Transactions on Automatic Control* AC-26(5):216–224.

Bellman, R. 1957. *Dynamic Programming.* Princeton, NJ: Princeton University Press.

Bertsekas, D. P. 1995. *Dynamic Programming and Optimal Control*, volume 1. Belmont, MA: Athena Scientific.

Cassandra, A. R.; Kaelbling, L. P.; and Littman, M. L. 1994. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 1023–1028. Seattle, WA: AAAI Press.

DeGroot, M. 1970. *Optimal Statistical Decisions.* New York, NY: McGraw Hill.

Fel'dbaum, A. A. 1965. *Optimal Control Systems.* New York, NY: Academic Press.

Jaakkola, T.; Singh, S. P.; and Jordan, M. I. 1995. Reinforcement learning algorithm for partially observable markov decision problems. In Tesauro, G.; Touretzky, D.; and Leen, T., eds., *Advances in Neural Information Processing Systems 7.* Cambridge, MA: MIT Press.

Lin, L. J. 1992. Self-improving reactive agents based on reinforcement learning. *Machine Learning* 8(3-4):293–321.

Meier, L. 1965. Combined optimal control and estimation. In *Proceedings of the 3rd Annual Allerton Conference on Circuit and System Theory.*

Narendra, K. S., and Annaswamy, A. M. 1989. *Stable Adaptive Systems.* Englewood Cliffs, NJ: Prentice Hall.

Parr, R., and Russell, S. 1995. Approximationg optimal policies for partially observable stochastic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1088–1094. Montreal, Quebec, Canada: Morgan Kaufmann.

Ram, A., and Leake, D. B. 1995. Learning, goals, and learning goals. In Ram, A., and Leake, D. B., eds., *Goal-Driven Learning.* Cambridge, MA: MIT Press. chapter 1.

Ross, S. M. 1993. *Introduction to Probability Models.* Boston, MA: Academic Press.

Rummery, G. A., and Niranjan, M. 1994. On-line q-learning using connectionist systems. Technical Report CUED/F-INFEG/TR66, Cambridge University Department.

Santamaría, J. C.; Sutton, R. S.; and Ram, A. 1996. Experiments with reinforcement learning in problems with continuous state and actions spaces. Technical Report UM-CS-1996-088, Department of Computer Science, University of Massachusetts, Amherst, MA.

Singh, S. P., and Sutton, R. S. 1996. Reinforcement learning with replacing eligibility traces. *Machine Learning* 22:123–158.

Stengel, R. F. 1994. *Optimal Control and Estimation.* Mineola, NY: Dover Publications.

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

Sutton, R. S. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning: Proceedings of the Seventh International Conference*, 216–224. New Brunswick, NJ: Morgan Kaufmann.

Sutton, R. S. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems* 8.

Tsitsiklis, J. N., and Van Roy, B. 1996. Analysis of temporal-difference learning with function approximation. *Advances in Neural Information Processing Systems* 9.

Watkins, C. J. C. H. 1989. *Learning from Delayed Rewards.* Ph.D. Dissertation, Univeristy of Cambridge, England.