IEEE Conference on Robotics and Automation, San Francisco, CA, April 2000, pp. 3290-3295.

A Hybrid Architecture for Hierarchical Reinforcement Learning

Manfred Huber
Dept. of Computer Science and Engineering
University of Texas, Arlington TX

Abstract

Autonomous robot systems operating in the real world have to be able to learn new tasks and environmental conditions without the need for an outside teacher. While reinforcement learning represents a good formalism to achieve this, its long learning times and need for extensive exploration often make it impracticable for on-line learning on complex systems. The hybrid architecture presented in this paper addresses this issue by applying reinforcement learning on top of an automatically derived abstract Discrete Event Dynamic System (DEDS) supervisor. This reduces the problem of policy acquisition within this approach to learning to coordinate a set of closed-loop control strategies in order to perform a given task. Besides dramatically reducing the complexity of the learning task this framework also permits the incorporation of a priori knowledge and facilitates the inclusion of learned policies as actions in order to transfer skills to new task domains. To demonstrate the applicability of this approach, the architecture is used to learn locomotion gaits on a four-legged robot platform.

1 Introduction

Autonomous robot systems operating in an unknown and uncertain environment have to be able to be reactive and adaptive in order to cope with changes in the environment and the task constraints. In extreme situations they have to be able to do so without the need for outside supervision. Reinforcement learning [1, 10] provides a good mechanism for this since it allows an agent to learn from its interaction with the environment and a simple reward signal. This reward can be derived from external stimuli present in the environment or from signals internal to the system. A major drawback of this framework, however, is its inherent need for large amounts of exploratory actions in order to find an optimal control policy. Most applications of reinforcement learning techniques have thus been either off-line in simulation or on relatively simple robot mechanisms [2, 4, 5, 6]. To address this problem, recent work has investigated the use of abstract actions within this learning framework [3, 8]. Such temporal abstractions promise to dramatically decrease the amount of on-line experience required by limiting the set of points at which the learning agent has to consider a new action to the times when specific control or sensor events occur. This, in turn, reduces the complexity of the control policies required to address given tasks. In addition, the use of closed-loop strategies as actions allows the learning system to ignore certain aspects of the state space. This provides the potential to bridge hidden state and facilitates state abstraction, permitting control policies to be learned in smaller, more abstract state spaces.

A second problem of most reinforcement learning approaches is that past experiences as well as control policies acquired for a given task do not transfer to other situations. This makes it necessary to re-learn from scratch when the task requirements or the environment changes. In many cases, however, previously learned tasks represent important subproblems of subsequent tasks and the transfer of learned policies could thus lead to a dramatic improvement in the learning performance for more complex tasks. To cope with this problem, a control architecture for such systems should be able to dynamically extend the abstract action space by including previously acquired control policies. This would incrementally increase the built in competence of the system and enable it to perform more and more complex tasks over time.

To address these issues, the hybrid control architecture presented here uses closed-loop control policies as abstract actions. Use of such actions permits to define the system as a Discrete Event Dynamic System (DEDS) on an abstract, discrete state space, within which a policy for the given task is learned. Furthermore learned policies can be treated in the same way as the original closed-loop actions and can be included over time to provide the basis for the acquisition of more complex control strategies. To illustrate this architecture and to show the potential of such a hierarchical extension of the action space, this framework is first applied to a simple rotation task on the four-legged walking robot shown in Figure 1. In a second experiment the resulting turning gait is added to the action set and a policy for a more complex locomotion task is learned. The task chosen here is a general goal reaching task in an open environment and demonstrates the advantages of permitting a hierarchical extension of the action space.

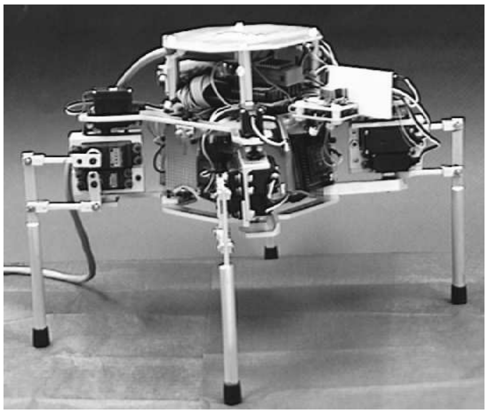

Figure 1: The four-legged walking robot.

2 The Hybrid Control Architecture

In a large variety of real world systems the overall dynamics has a continuous character while much of the decision making follows a discrete pattern which becomes more and more coarse with increasing behavioral competence. This permits such systems to be described as hybrid discrete/continuous systems. The approach used in this paper attempts to take advantage of this character by employing a hybrid control structure as an interface between the physical world and a learning component. This structure employs continuous, closed-loop control policies as abstract actions and uses these to build an abstract discrete event model of the system behavior which forms the basis for the reinforcement learning task. Use of this hybrid scheme efficiently reduces the size of the state space used by the learning component and provides a means to incorporate a priori knowledge to guide exploration and thus to influence the learning performance explicitly.

2.1 Closed-loop Control Policies as Action Primitives

The main advantage of using abstract, temporally extended actions in a learning framework is that they can be used to reduce the number of decision points for the learning component and thus permit faster learning. In order to do this effectively, however, the abstract actions have to provide some reactivity and be able to suppress small perturbations locally. This robustness alleviates the need to strategies which frequently switch between actions to actively counter outside influences. The approach presented here addresses this by using stable and convergent closed-loop control policies as abstract actions. Use of such actions not only provides some local robustness but also leads to a partitioning of the underlying physical space into a set of stable regions within which the control policies drive the robot system towards an attractor. Figure 2 shows the control surface of such a closed-loop policy.

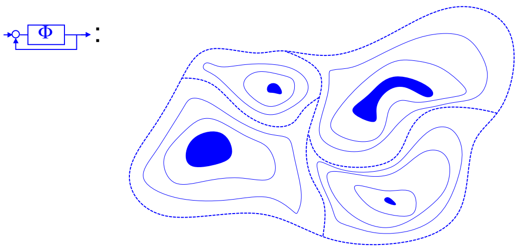

Figure 2: Control surface for a closed-loop control policy.

The lines in this figure represent the equipotential lines of the control surface, the dashed lines indicate the boundaries between different basins of attraction, and the shaded regions are the attractors for this control policy. This plot illustrates the "funneling" character of these closed-loop actions. Whenever the system is within a basin of attraction, it is driven towards the corresponding attractor even in the presence of small perturbations. This dramatically reduces the need to learn compensatory actions throughout system operation and permits control actions to be executed until convergence. The behavior of the overall system can then be described largely in terms of these attractors.

In addition to the local robustness, the choice of stable closed-loop policies as actions also facilitates the concurrent activation of more than one control policy. In the approach taken here an "extended nullspace" constraint is used to permit concurrent activations

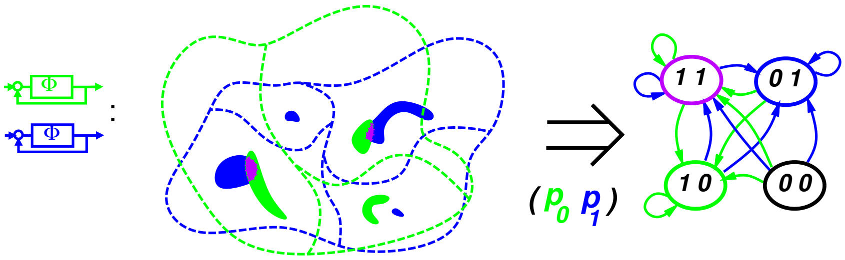

Figure 3: Mapping from controllers to predicate states.

while resolving resource conflicts and avoiding unstable behavior. This constraint prioritizes the control actions such that a lower priority action can not counteract the progress of a higher priority one and thus ensures that the stability and convergence properties of the original closed-loop controllers are inherited by the composite actions. Using this scheme thus permits the construction of a relatively large and locally robust action repertoire from a small number of initial policies.

2.2 The Discrete Event Supervisor

If each control action is executed until convergence, the behavior of the system under the control of this stable closed-loop control policy can be largely characterized by the attractors of the available control policies. This permits to transform the underlying physical space into a discrete space of system equilibria. Each of these discrete states is characterized by predicates, p_i, corresponding to the attractors of the control surfaces of the individual abstract actions. Using this and interpreting convergence of actions as control events, the system can subsequently be modeled as a DEDS with an abstract state space. Figure 3 shows this mapping from a set of two closed-loop control policies to an abstract finite state description of the system behavior.

This DEDS model takes the form of a finite state automaton where the numbers in the states represent the value of the predicates indicating if the system is within an attractor for the corresponding abstract action. The arrows, on the other hand, represent all possible transitions within this state space if a given control action is executed. While this state abstraction might produce hidden state, its action dependent choice should ensures that the discrete space encompasses all tasks directly addressable by the underlying actions.

The use of this description of the behavior of the system dramatically reduces the state space that has to be considered by the reinforcement learning component, thus offering the potential for a dramatic reduction in the learning time. Moreover, the use of the DEDS framework [9] opens the construction of task specific control strategies to a large body of formal techniques which permit the incorporation of potentially time varying constraints. These can be used to limit exploration to safe and relevant action alternatives and to simulate maturational processes throughout a subsequent learning phase.

2.3 Learning a Discrete Event Control Policy

To address new tasks, the constrained DEDS model is interpreted here as a Markov decision process and Q-learning[10] is used to acquire a control policy for a given reinforcement signal. This learning scheme effectively learns a sequence of control actions which performs the desired task. At the same time, the approach take here utilizes the exploratory actions performed throughout the learning phase to estimate the transition probabilities between predicate states. This improves the abstract model of the system behavior on-line. Such a model, in turn, can later be used for off-line learning [7], as well as for the transfer of learned control policies into the space of abstract actions. Figure 4 shows the complete hybrid control and learning architecture.

In this approach, all direct sensory input and actuator output is handled by the abstract actions in the lowest layer. Activation and convergence of the individual or composite actions are then interpreted as discrete events in the abstract DEDS model which forms the basis for the reinforcement learning system. The learning process, in turn, determines which control action to execute in each of the discrete states, leading to a sequence of control activations which successfully perform a given task.

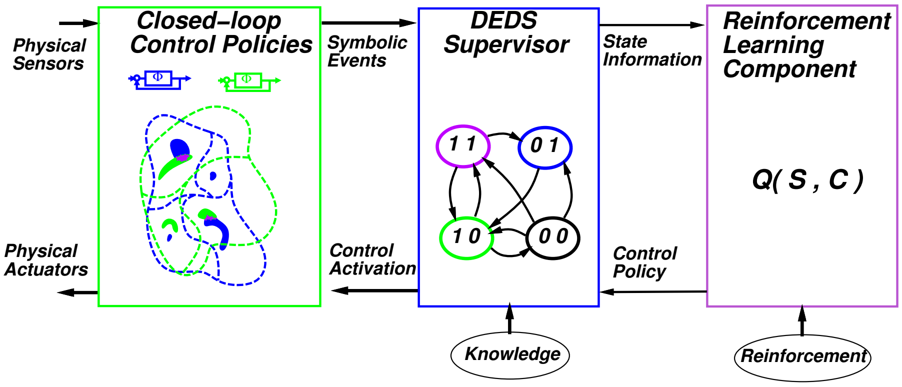

Figure 4: The hybrid control architecture.

2.4 Robot Locomotion Experiments

To illustrate this learning and control architecture, this section presents an example of the overall architecture applied to a the on-line acquisition of a goal-directed turning gait for a four-legged walking robot. The initial set of closed-loop control laws used here contains a set of contact configuration controllers which attempt to optimize the stability of a given three or four-legged stance by either moving individual legs or by shifting the center of mass of the robot. In addition a set of posture controllers which move the robot in order to optimize the posture of individual legs, and a set of path controllers are provided as control actions. The path controllers used here adjust the orientation and location of the center of mass of the robot to approach a given goal[1]. This choice of control actions then leads to a discrete system model where states are described by a vector of predicates corresponding to the stability of individual stances, to the quality of the kinematic configuration of the legs, and to the achievement of the given goal location.

For this experiment, the set of actions is limited in the DEDS model to actions which do not translate the center of mass of the robot. Furthermore a static stability constraint is imposed which prevents the robot from falling throughout the learning process. All these constraints reduce the size of the discrete state space to 16 states with a total of 157 possible actions. Using this control system, the robot is then put on a flat surface and started with an immediate reward signal proportional to its angular progress. Whenever the robot reaches the desired heading, a new reference heading is generated randomly and the learning process is continued. Figure 5 shows a representative learning curve for this turning gait.

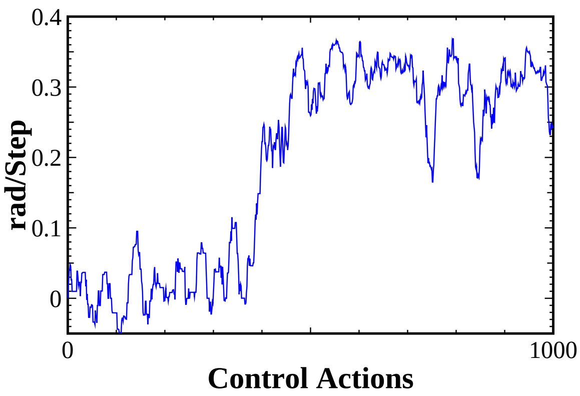

Figure 5: Learning curve for a turning gait in a 5-dimensional predicate space.

This graph shows that the system rapidly acquires a robust strategy for turning towards a given heading within about 600 learning steps. Each learning step here corresponds to the execution of a single control action and the achievement of a desired heading generally requires a larger sequence of such steps. Most of the variation occurring after this point is due to the continuing resetting of the reference heading and to the remaining 10% of purely random exploratory actions. On the real robot the acquisition of this turning gait takes approximately 12 minutes. This learning performance clearly illustrates the potential power of closed-loop control policies as abstract actions in this hybrid control architecture.

[1]For more details on the underlying controllers and the turning gait experiment see [3]

3 Hierarchical Action Spaces

In many situations, previously learned task are related to new tasks and the transfer of acquired control policies is thus important to reduce the learning complexity. In the approach presented here, the underlying action space can be easily extended to include such learned control policies. One important factor in this is that the use of the DEDS supervisor as the substrate for learning allows each abstract action to use its own, task-specific state abstractions, independent of the DEDS model currently used by the learning system. Furthermore, due to the exploration driven learning process, learned policies generally incorporate robust strategies which can deal with local deviations from the predicted behavior. This provides each of these policies with its own "funneling" effect towards stable regions in the discrete state space. Figure 6 shows the discrete "control surface" of a learned policy.

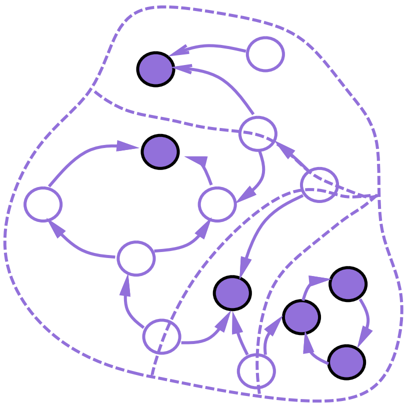

Figure 6: Policies as discrete closed-loop control elements.

This figure also shows that the discrete policy partitions the underlying space into different attractor regions. These attractors (indicated by shaded states) can be either static as in the case of policies with absorbing goal states, or dynamic in which case the attractor forms a stable cycle through the discrete predicate space. In both cases reaching of an attractor can be used as a control event, permitting the learned policy to be treated in the same way as the initial abstract actions. Doing this promises more complex tasks to be learned using previously acquired solutions to subtasks.

3.1 Walking Experiment

To demonstrate the reuse of learned control policies and to investigate the benefits of such a hierarchical extension of the action space, simulation experiments are performed here in order to learn a more complex goal-directed locomotion task. In two experiments, this task is learned with and without the inclusion of an additional action in the form of the goal-directed rotation gait acquired in the previous section. In this example, the initial action space is not limited and only the static stability constraint is imposed, resulting in a 12-dimensional predicate space and a total of 231 different actions. The immediate reward used here is proportional to the amount of progress towards a fixed goal location and the robot is started from random locations with the objective of reaching the given goal. Whenever this goal is reached, the robot is placed in a new, random location. Figure 7 shows the average learning curves over 10 runs with and without the learned turning policy as an additional action.

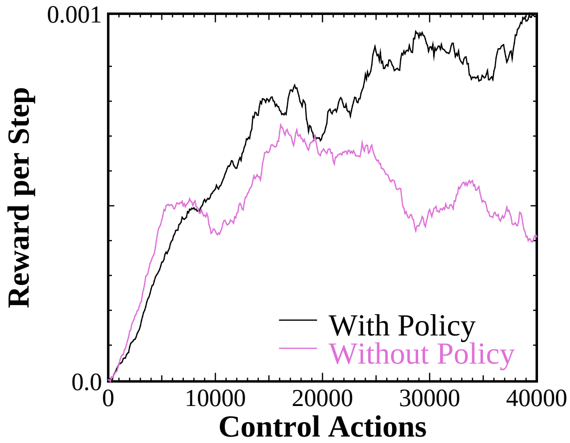

Figure 7: Learning curves with and without the rotation policy.

These learning curves show that the inclusion of the policy initially slows down learning due to the higher number of actions available and the resulting need for more exploration. Over time, however, the learned policy with the additional action outperforms the one without and leads to the policy illustrated by the example paths in Figure 8.

The long-term advantage provided by the inclusion of learned control policies as additional actions results here from two separate aspects. First, the use of such complex control actions can reduce the number of steps required to perform a given task. After an initial phase in which the system has to discover the utility of these

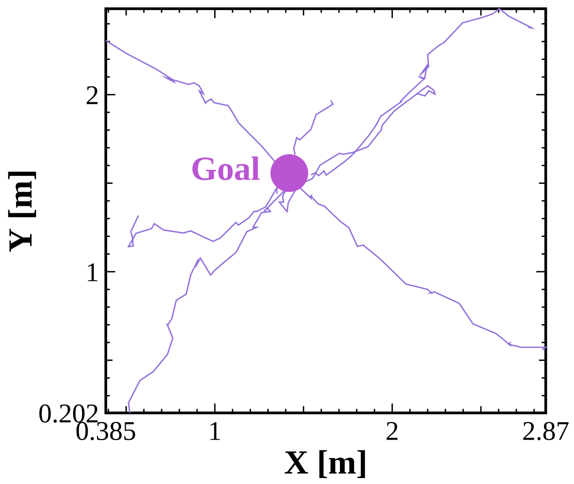

Figure 8: Example paths for the policy learned using the acquired turning gait as a possible action.

complex actions in the new context, this can lead to an acceleration of the overall learning process. A second effect of the use of more temporally extended actions is that they can bridge hidden state introduced by the state abstractions employed. This, in turn, permits the system to acquire control policies which could not be formed using only the initial set of closed-loop control elements.

4 Conclusions and Future Work

The hybrid control architecture presented in this paper is designed to address on-line learning and control in complex robot systems. To achieve this it employs a set of closed-loop control policies as abstract actions and an abstract DEDS model in order to reduce the complexity of the subsequent reinforcement learning problem. In addition, this framework permits to transfer previously acquired skills by including learned control policies in the set of abstract actions. This in turn can further reduce the learning times for subsequent tasks and permit more complex tasks to be learned. The experiments performed, however, also show that the increase in the size of the action space can lead to degrading initial performance, illustrating the importance of better action selection methods. Furthermore, the incorporation of learned policies can lead to additional predicates in the DEDS state space and thus to additional, more powerful state abstractions.

Acknowledgments

Parts of this work were conducted in the Laboratory for Perceptual Robotics at the University of Massachusetts at Amherst and supported by NSF IRI-9503687, IRI-9704530, and CDA-9703217.

References

[1] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. Technical Report 93-02, University of Massachusetts, Amherst, MA, 1993.

[2] V. Gullapalli. *Reinforcement Learning and its Application to Control*. PhD thesis, University of Massachusetts, Amherst, MA, 1992.

[3] M. Huber and R. A. Grupen. A feedback control structure for on-line learning tasks. *Robotics and Autonomous Systems*, 22(3-4):303–315, December 1997.

[4] L.-J. Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, January 1993.

[5] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992.

[6] M. J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[7] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13, 1993.

[8] D. Precup and R. S. Sutton. Multi-time models for temporally abstract planning. In *Advances in Neural Information Processing Systems 10*, pages 1050–1056, Denver, CO, December 1997. MIT Press.

[9] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–97, January 1989.

[10] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.