Planning with macro-actions: Effect of initial value function estimate on convergence rate of value iteration.

Milos Hauskrecht
Department of Computer Science, Box 1910
Brown University, Providence, RI 02912
milos@cs.brown.edu

Working paper

Abstract

We investigate the use of temporally abstract actions, or macro-actions, in speeding-up the solution of Markov decision processes. We focus on an augmented MDP model that combines both primitive actions and macro-actions and was shown empirically to speed-up the convergence of value iteration routines under the proper choice of macro-actions. We show that the convergence rate of value iteration methods for this model is sensitive not only to the choice of macro-actions but also to the value function estimate used to initialize the procedure. The theoretical result is demonstrated also experimentally on a simple robot maze navigation problem.

1 Introduction

Complex sequences of actions or macro-actions have been used in planning in deterministic domains for some time [6, 12, 11] and they have been proven usefull in many applications. The ability to extend the idea to Markov Decision Processes (MDPs) is crucial for domains that cannot be modeled deterministically and that require a stochastic model of the dynamics. Robot navigation, with precompiled procedures to handle obstacles in the environment, or medical treatment planning with higher level treatment protocols are typical examples. In general any procedure or program involving multiple actions and spanning multiple time steps corresponds to a macro and requires an appropriate model.

Integrating programs or macro-actions into the decision process is a difficult task given that the execution of different macro-actions may extend over different periods of time. To deal with this problem, Precup, Sutton and Singh [17, 15, 16] have developed *multi-time models* and applied them to planning with MDPs. The new framework allows us to represent uniformly actions on different time scales, as well as, to apply standard problem-solving procedures developed for MDPs, including value and policy iteration [1, 10].

There are only a few reasonably direct ways in which macro-actions can be applied to solve MDPs. Two of them are most common. First, macro-actions can be combined with primitive actions to form an *augmented MDP model.* This model has been studied extensively in [17, 15, 16, 8, 9]. The second model works with macro-actions only and allow one to reduce the number of states of the underlying MDP to states connected by macro-actions. This model is called an abstract MDP model (after [9]) and was studied in [7, 9, 8, 14]. The abstract model is suitable for hierarchical methods (see e.g. [9]) and provides an alternative to various approximation methods for solving large MDPs [5, 2, 4, 3]. In our work we focus our attention solely on the augmented model that works with both primitive actions and macro-actions.

Precup, Sutton and Singh [15, 16] demonstrated empirically the advantage of adding macro-actions to the original model by speeding-up the convergence rate of value iteration when a good set of macros is used. [1] However this improvement is not always guaranteed and this also in the case when only the optimal macro-action is added to the model. In this paper we show that the improvement in the convergence rate is sensitive to the value function estimate used to initialize value iteration. More specifically, we prove that for some initial value functions the convergence rate of the value iteration for the model with additional macro-actions is always dominated by the original model without macro-actions. The result provides first theoretically backed insight on various tradeoffs involved in planning with a model augmented with macro-actions. The effect of the initial value function estimate on the convergence rate of value iteration is demonstrated and supported by experiments on a robot maze navigation problem, presented at the end of the paper.

2 Model of Temporally Abstract Actions for MDPs

2.1 Markov Decision Processes

A *Markov decision process* is a 4-tuple $\langle S, A, T, R\rangle$ where: S is a finite set of states; A is a finite set of actions; T is a transition model $T : S \times A \times S \rightarrow [0, 1]$, such that $T(s, a, \cdot)$ is a probability distribution over S for any $s \in S$ and $a \in A$; and $R : S \times A \rightarrow I\!R$ is a bounded reward function. Intuitively, $T(s, a, s')$ denotes the probability of moving to state s' when action a is performed at state s, while $R(s, a)$ is the immediate reward associated with action a in s.

Given an MDP, the objective is to construct a *policy* that maximizes expected accumulated reward over some horizon of interest. We focus on *infinite discounted horizon* problems, where we adopt a policy maximizing $E(\sum_{t=0}^{\infty} \beta^t \cdot r^t)$, where r^t is a reward obtained at time t and $0 < \beta < 1$ is a discount factor. In such a setting, we restrict our attention to stationary policies of the form $\pi : S \rightarrow A$, with $\pi(s)$ denoting the action to be executed in state s. The value of the optimal policy satisfies [1]:

$$V^*(s) = \max_a \left[R(s, a) + \beta \sum_{s' \in S} T(s, a, s') \cdot V^*(s') \right].$$

The equation could be rewritten also as $V^* = HV^*$, where $H : B \rightarrow B$ denotes a value function mapping, and $B : S \rightarrow R$ is a set of bounded real valued functions.

A number of techniques for constructing optimal policies exist. An especially simple algorithm is *value iteration* [1]. In value iteration we produce a sequence of value functions V^n by starting from an arbitrary V^0, and defining

$$V^{i+1}(s) = \max_{a \in A} \left[R(s, a) + \beta \sum_{s' \in S} T(s, a, s') \cdot V^i(s') \right] = (HV^i)(s). \tag{1}$$

The sequence of functions V^i converges to V^* in the limit. Each iteration is known as a *Bellman backup*. After some finite number n of iterations, the choice of maximizing action for each s forms an optimal policy π and V^n approximates its value.

[1] Obviously the benefit from adding macro-actions is diminished when bad macro-actions are used.

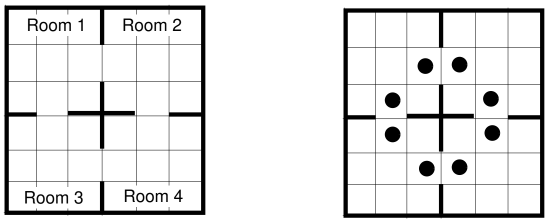

Figure 1: (a) A four-room example. (b) Peripheral states for a room partitioning.

2.2 Macro-actions

Macro-actions allow us to model complex sequences of action, for example, typical or standardized policies or programs. To illustrate this assume a simple robot navigation problem in Figure 1(a), in which the robot moves in four compass directions. In addition to the basic move actions a programmer may have provided a macro (or program) that enables the robot to exit one of the rooms through a particular door. Such a macro consists of multiple steps involving multiple primitive move actions.

Macro-actions can be modeled by different means. For example, macro-actions can be represented as programs with arbitrary termination conditions as suggested by Precup, Sutton and Singh [17, 15, 16] or using finite state machine representation and its hierarchical extensions as proposed by Parr [13]. In our work we assume a simpler macro-action model based on Hauskrecht et al. [9], in which every macro-action is represented as a local policy restricted to some region of a state space. Formally, our model relies on a *region-based decomposition* of an MDP as defined in [4, 9].

Definition 1 *A* region-based decomposition Π *of an MDP* $M = \langle S, A, T, R \rangle$ *is a partitioning* $\Pi = \{S_1, \cdots, S_n\}$ *of the state space* S*. We call the elements* S_i *of* Π *the* regions *of* M*. For any region* S_i*, the* exit periphery *of* S_i *is*

$$XPer(S_i) = \{s' \in S - S_i : T(s, a, s') > 0 \text{ for some } a \in A, s \in S_i\}.$$

The collection of all peripheral states is denoted

$$Per_\Pi(S) = \cup_i \{XPer(S_i) : i \leq n\}.$$

Figure 1(b) shows the set of peripheral states obtained if we partition the problem of Figure 1(a) into the four regions corresponding to different rooms.

Definition 2 *A* macro-action *for region* S_i *is a local policy* $\pi_i : S_i \rightarrow A$*.*

A *macro-action* is simply a local policy defined for a particular region S_i. Intuitively, this policy can be executed whenever an agent enters or is in the region and terminates when the agent leaves the region (if ever). Definitions in [15] or [13] are more general and can use arbitrary starting and termination conditions, and allow also non-Markovian policies.

The main problem with integrating macro-actions into the MDP is that macro-actions when executed can extend over different periods of time. The key insight of Precup, Sutton and Singh [17, 15, 16] is that one can treat a macro-action as a *primitive action* in the original MDP if one has an appropriate reward and transition model for the macro. They propose the following method of modeling macros.

Definition 3 *A* discounted transition model $T_i(\cdot, \pi_i, \cdot)$ *for macro* π_i *(defined on region* S_i*) is a mapping* $T_i : S_i \times XPer(S_i) \rightarrow [0, 1]$ *such that*

$$T_i(s, \pi_i, s') \quad = \quad E_\tau(\beta^{\tau-1} \cdot \Pr(s^\tau = s' \mid s^0 = s, \pi_i)),$$

where the expectation is taken with respect to time τ of termination of π_i. A discounted reward model $R_i(\cdot, \pi_i)$ *for π_i is a mapping $R_i : S_i \rightarrow I\!R$ s.t.*

$$R_i(s, \pi_i) = E_\tau(\sum_{t=0}^{\tau} \beta^t R(s^t, \pi_i(s^t)) \mid s^0 = s, \pi_i),$$

where the expectation is taken with respect to completion time τ of π_i.

The discounted transition model specifies the probability of leaving S_i via a specific exit state, similarly to a standard stochastic transition matrix, with one exception: the probability is *discounted* according to the expected time at which that exit occurs. As demonstrated in [15, 16], this clever addition allows the transition model to be used as a normal transition matrix in any standard MDP solution technique, such as policy or value iteration. [2] The reward model is similar, simply measuring the expected accrued reward during execution of π_i starting from a particular state in S_i.

3 Augmented MDP

Suppose we are given an MDP M and a set of macros defined for each region S_i induced by some state space partitioning Π. There are only a few reasonably direct ways in which macro-actions can be applied to solve M. In our work we focus on *augmented MDP* model [17, 15, 16, 8, 9], denoted M', constructed by extending the action space A using a set of macro-actions, assuming that macro models are used to determine transitions and rewards associated with these new actions. An alternative model with macro-actions is *abstract MDP* [7, 9, 8, 14] that works with macro-actions and peripheral states only. This allows one to reduce significantly the size of the state space compared to M.

Although the augmented model uses discounted transition matrices for macro-actions and thus it is not a classical MDP, it can be still solved using standard methods, such as value iteration. Because all *base level* actions (those in A) are present, the policy found is guaranteed to be optimal. Furthermore, the presence of macros may enhance the convergence of value iteration, as demonstrated in [15, 16]. [3] This is because the single "application" of a macro can propagate values through a large number of states and over a large period of time in a single step. However, the potential for enhanced convergence is not always guaranteed and it is sensitive to the initial value function. We will elaborate this issue next.

At first sight the planning with as augmented MDP is the same as for the original MDP, that is one can simply treat macroactions as any other actions and apply value iteration starting from an arbitrary initial value function. However the main difference is that primitive actions can be subsumed by macro-actions and that action choices are not necessarily alternatives. This can lead to situations in which optimal macro is never preferred and thus never used for speeding up the computation, contrary to our intention. To show that that value iteration for augmented MDPs can become inferior we provide the following theorem.

Theorem 1 *Let M and M' be MDPs such that M' differs from M only in additional actions, i.e. $A_M \subset A_{M'}$. Then $H_M V \leq H_{M'} V$ holds for an arbitrary value function V, that is, the value function mapping for the augmented MDP model always upper-bounds the value function mapping for the original MDP.*

Proof The value function for state s obtained after the update is computed as

$$V^+(s) = \max_{a \in A} Q^+(s, a),$$

[2] Our definition of the discounted transition model is consistent with Equation 1, while Precup, Sutton and Singh fold the discount factor into the transition model.

[3] We note that these savings do not account for the possible overhead associated with generating macros and constructing appropriate model for each macro.

where A is a set of actions and $Q^+(s,a)$ is an action-value function

$$Q^+(s,a) = R(s,a) + \sum_{s' \in S} T(s,a,s')V(s').$$

Then if M and M' differ only in additional actions, i.e. $A_M \subset A_{M'}$, regardless of transitions and rewards associated with additional actions, it holds

$$\max_{a \in A_M} Q^+(s,a) = H_M V(s) \leq \max_{a \in A_{M'}} Q^+(s,a) = H_{M'} V(s),$$

or in other words value function mapping $H_{M'}$ upper-bounds H_M. □

Note that the above theorem is universal and holds for any MDP including those with or without macroactions. The result can be used to show that value iteration (VI) for an MDP with both primitive and macro actions is inferior (slower) when initialized from the upper bound.

Theorem 2 *Let M' be an augmented MDP with both primitive and macro actions and M be an MDP with primitive actions only. If value iteration for both M and M' starts from the initial upper bound $V_0 \geq V^*$, then the value iteration for M' converges at equal or slower rate than the value iteration for the base model M and thus it is inferior.*

Proof A key feature of an augmented MDP is that it converges to the same fixed point solution as the original MDP, i.e. $V^* = H_M V^* = H_{M'} V^*$. Also, by theorem 1, the value function mapping $H_{M'}$ upper-bounds H_M. Then for any initial value function $V_0 \geq V^*$ holds $V^* \leq H_M V_0 \leq H_{M'} V_0$. As both $H_{M'}$ and H_M are also isotone we can write $V^* \leq H_M^2 V_0 \leq H_{M'}^2 V_0$ for two steps. Extending the result into i steps we get $V^* \leq H_M^i V_0 \leq H_{M'}^i V_0$. Therefore a value function obtained using VI for the MDP with primitive actions only is better (closer) to the optimal value function after i steps. As the model augmented with macroactions requires that at every step the Q-value for every macroaction is evaluated, the method is clearly inferior and leads to slower running times. □

The theorem shows that one should not blindly apply value iteration to MDPs with macroactions as the potential benefit and intended speed-up is not guaranteed. To avoid the suboptimal behavior one should always start value iteration from the initial lower bound. This choice guarantees that macroactions can be useful and improve the convergence.

Theorem 3 *Let M' be an MDP with both primitive and macro actions and M be an MDP with primitive actions only. If value iteration for both M and M' starts from the initial lower bound $V_0 \leq V^*$, then the convergence rate of value iteration for M' is no slower than the convergence rate for M.*

Proof The proof is similar to the previous case. Starting from the lower bound $V_0 \leq V^*$ and knowing that $H_{M'}$ and H_M are isotone contractions and that $H_{M'}$ upper bounds H_M, it must hold $H_M^i V_0 \leq H_{M'}^i V_0 \leq V^*$. Therefore a value function obtained using VI for M' converges to the optimal value function no slower than VI for M. □

Intuitively, longer and better macroactions influence the convergence of value iteration most. This is because they tend to propagate value function changes faster and improve the lower bound value function in larger steps. Thus they are more likely to provide the optimizing Q-value from among candidate actions and to determine the resulting value function value. However note, that starting from the appropriate bound does not neccesarily imply that the actual speed-up (in terms of running time) will be always achieved. This because when one considers a larger number of macro-actions in every state, the slowdown resulting from the computation of updates for every macro could outweight the benefit of the improved convergence.

4 Experimental results

To demonstrate the computational savings made possible by using macro-actions in planning tasks, we have performed experiments on the simple navigation problem in Figure 2. The

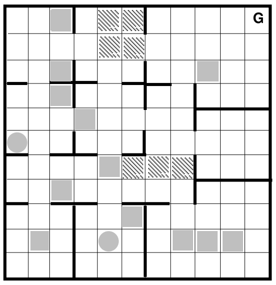

Figure 2: Test problem. Shaded squares denote locations with higher cost, patterned squares represent areas in which moves are more uncertain, shaded circles denote absorbing states with a finite positive cost, and G stands for a zero cost goal state.

agent can move in any compass direction to an adjacent cell or stay in place. The move actions are stochastic, so the agent can move in an unintended direction with some small probability. The objective is to maximize the expected discounted rewards (costs are modeled as negative rewards) incurred by navigating the maze, with each state, except the zero-cost absorbing goal state, incurring some cost. The costs and transition probabilities are not uniform across the maze.

We compared the results of value iteration for both the original MDP and the augmented MDP, the latter using a set of additional macros for every room in the problem. The macros were generated automatically using a simple heuristic strategy described in [9] that results in $|XPer(S_i)|+1$ macros for every region S_i: one macro per exit state pressing the agent towards that exit, plus a 'stay-in' macro pressing the agent not to move out of the region.

Figure **??** shows how the estimated value (maximal expected reward) of a particular state improves with the time (in seconds) taken by value iteration on both models. When the initial value function estimate is a lower bound the augmented MDP leads to faster convergence of the value function. This is because of the ability of macros to propagate value through a large number of states produces large changes in the value function in a single iteration step, overcoming the increased number of actions. Note, however, that when the initial estimate of the value function is an upper bound, the augmented MDP actually performs worse than the original MDP, as proved earlier. The average time (in seconds) taken per value iteration step in this example is 0.045 for the original MDP, 0.12 for the augmented MDP, which reflects the increased action space of the augmented MDP model, as expected.

5 Conclusion

The convergence rate of value iteration for MDPs can be boosted by adding a small set of good macro-actions, representing sequences of primitive actions or programs. This behavior was demonstrated empirically in [15, 16]. However, in our work we showed that the speed-up in convergence rate of value iteration in this model is not guaranteed and that the rate does not depend only on the quality of macro-actions used but it is also sensitive to the value function used to initialize the procedure. We proved that when the iteration starts from the upper bound of the optimal value function the augmented MDP model is always inferior to the original MDP with regard to convergence rate. This provides a first theoretically backed insight on various tradeoffs involved in planning with an augmented MDP model. The effect of sensitivity of the augmented MDP model to the initial value function estimate was demonstrated also empirically on a simple robot maze naviagtion problem.

An interesting and open question related to our result is the effect of initial value function estimate on reinforcement learning (RL) with both primitive actions and macro-actions. An important issue here is the fact that the upper bound value function estimate, which causes

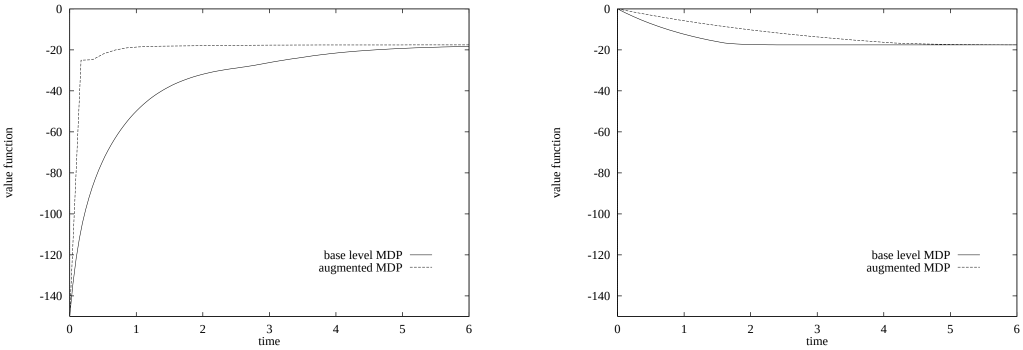

Figure 3: Solution quality versus time for the augmented and original MDP models. Results with a good initial value function estimate (w.r.t. the augmented MDP) are shown on the left. Results for a poor initial estimate are on the right. In the latter case, the augmented MDP converges more slowly than the original MDP.

the slowdown for MDPs, promotes exploration when applied in RL settings. Thus there are two opposite forces influencing the convergence to the correct value function, both depending on the initial value function estimate. Investigation of these remains an open question though our preliminary results indicate that the macro-action effect dominates.

References

[1] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, 1957.

[2] D. P. Bertsekas and J.. N. Tsitsiklis. *Neuro-dynamic Programming.* Athena, 1996.

[3] T. Dean and R. Givan. Model minimization in Markov decision processes. *AAAI-97*, pp.106–111, Providence, 1997.

[4] T. Dean and S.-H. Lin. Decomposition techniques for planning in stochastic domains. *IJCAI-95*, pp.1121–1127, Montreal, 1995.

[5] R. Dearden and C. Boutilier. Abstraction and approximate decision theoretic planning. *Artif. Intell.*, 89:219–283, 1997.

[6] R. Fikes, P. Hart, and N. Nilsson. Learning and executing generalized robot plans. *Artif. Intell.*, 3:251–288, 1972.

[7] J. P. Forestier, P. Varaiya. Multilayer control of large Markov chains. *IEEE Trans. on Aut. Control*, 23:298-304, 1978.

[8] M. Hauskrecht. Planning with temporally abstract actions. Technical report, CS-98-01, Brown University, 1998.

[9] M. Hauskrecht, N. Meuleau, C. Boutilier, L. Kaelbling, and T. Dean. Hierarchical Solution of Markov Decision Processes using Macro-actions. In Proceedings of the 14-th Conference on Uncertainty in Artificial Intelligence, pp. 220-229, 1998.

[10] R. A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, 1960.

[11] R. Korf. Macro-operators: A weak method for learning. *Artif. Intell.*, 26:35–77, 1985.

[12] S. Minton. Selectively generalizing plans for problem solving. *IJCAI-85*, pp.596–599, Boston, 1985.

[13] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. In M. Mozer, M. Jordan, T. Petsche, eds., *NIPS-11.* MIT Press, 1998.

[14] R. Parr. Hierarchical control and learning with hierarchies of machines. Chapters 1-3, under preparation, 1998.

[15] D. Precup and R. S. Sutton. Multi-time models for temporally abstract planning. In M. Mozer, M. Jordan, and T. Petsche, eds., *NIPS-11.* MIT Press, 1998.

[16] D. Precup, R. S. Sutton, and S. Singh. Theoretical results on reinforcement learning with temporally abstract behaviors. *10th Eur. Conf. Mach. Learn.*, Chemnitz, 1998.

[17] R. S. Sutton. TD models: Modeling the world at a mixture of time scales. In *ICML-95*, pp.531–539, Lake Tahoe, 1995.