Variable Resolution Hierarchical RL

Bernhard Hengst
National ICT Australia
Computer Science and Engineering
University of New South Wales
Sydney 2052 Australia
bernhardh@cse.unsw.edu.au

UNSW CSE TR 0309

Abstract

The contribution of this paper is to introduce heuristics, that go beyond safe state abstraction in hierarchical reinforcement learning, to approximate a decomposed value function. Additional improvements in time and space complexity for learning and execution may outweigh achieving less than hierarchically optimal performance and deliver anytime decision making during execution. Heuristics are discussed in relation to HEXQ, a MDP partitioning that generates a hierarchy of abstract models using safe state abstraction. The approximation methods are illustrated empirically.

1 Introduction

Not only do humans have the ability to form abstractions, but we control the amount of detail required to represent complex situations and to make decisions. To decide the best way to travel to a conference in Vancouver, for example, we may choose from available modes of intercity transport; car, bus, aircraft or train. The final decision may even take into consideration connections at either end for each mode of primary transport. However, the way we exit our home (front door, back door or garage door) is unlikely to be an influencing factor in our decision, although the final execution of the plan will use one of the doors to exit the home. How could a reinforcement learner model these pragmatics?

The Markov property significantly reduces the potential hypothesis space for MDPs but the state space still grows exponentially with the number of variables. For many complex environments we can exploit additional constraints. Examples of weak kinds of inductive bias are repetition, spacial coherence and weak coupling. By abstracting away redundant or irrelevant information and modelling the problem approximately we can reduce the complexity and solve it more efficiently.

The objective of this paper is to consider variable levels of model resolution to approximate hierarchically decomposed Markov decision problems that have already been safely state abstracted. Section 2 reviews model reductions for MDPs and hierarchically decomposed MDPs. Section 3 reviews the automatic HEXQ decomposition of a MDP and shows that it produces a hierarchy of abstract models. Section 4 introduces three variable resolution heuristics. Section 5 shows the results for Kaelbling's 10×10 maze [1]. We conclude with some discussion on future research directions.

2 Modelling, Homomorphism and Abstractions

Ashby [2] described a *model* as a state-action homomorphism between Markov machines. A homomorphism is a many-one mapping that preserves certain operations of interest. For example, the state transition function in a MDP is a good model of the environment if it accurately reflects the probabilistic behaviour of the environment. Additional homomorphic reductions may be possible to further simplify these models. Dean and Givan's model minimisation [3] is such a homomorphism (stochastic bisimulation homogeneity). This type of homomorphism is related to algorithms by Boutilier *et al.* [4] that use structure in factored MDPs represented as two stage temporal Bayes nets to effect the reduction.

Additional abstraction opportunities my be available by introducing multi-level hierarchical decompositions where each level acts as gating mechanism switching in sub-controllers or behaviours in a similar manner in which programs call subroutines. Hierarchical reinforcement learning frameworks such as HAMs [5], *options* [6], MAXQ [7], HEXQ [8] and ALisp [9] are not homomorphic reductions from original "flat" MDPs, but usually constrain the sub-task policies in various ways to simplify the learning. It appears it is not possible in general to significantly improve the computational complexity of solving a MDP and give optimality guarantees. For example in HAMs the designer specifies the underlying abstract machine that constrains the policies. The solution quality will depend on the heuristics implemented by the machine. Optimality is therefore often considered only in relation to a given hierarchical representation.

We thus concern ourselves with homomorphic reductions of *given* hierarchical MDP representations. The aforementioned hierarchical reinforcement learning frameworks all use a SMDP formalism at abstract levels. Ravindran and Barto [10] have defined a SMDP homomorphism h as a set of many-one mappings $\langle f, g_s \rangle$ from SMDP $M\langle S, A, T, R \rangle$ to SMDP $M'\langle S', A', T', R' \rangle$ with $h((s,a)) = (f(s), g_s(a))$, where $f : S \rightarrow S'$ and $g_s : A_s \rightarrow A'_{f(s)}$ for $s \in S$, such that $\forall s,\ s' \in S,\ a \in A_s$ and $N \in \mathcal{N}$: (1)

$T'(f(s), g_s(a), f(s'), N) = \sum_{s" \in [s']_f} T(s, a, s", N)$ and (2) $R'(f(s), g_s(a), N) = R(s, a, N)$. The surjection f induces equivalent classes of states s of M (i.e. $[s]_f$). $R(s, a, N)$ is the expected reward for performing action a in state s and completing it in N steps time.

A safe state abstraction is a homomorphism. By a ***safe*** abstraction we mean that the state value function is the same for the original and reduced MDP. Dietterich identified three kinds of safe state abstraction conditions for a MAXQ decomposed value function: (1) The elimination of irrelevant variables within a sub-task. This is closely related to model minimisation. (2) ***Funnel actions***, are temporally extended actions moving from a larger number of states to a smaller number of resultant states. (3) ***Shielding***, ensures states for which all parent tasks are terminated do not require values to be stored. Interestingly, funnel actions can be interpreted as a kind of hierarchical model minimisation in which a whole sub-task is abstracted to a single state. This condition is difficult to satisfy in a discounted setting because the discount rate applied to the resultant state depends on N, the number of steps to termination of the sub-task. With the introduction of an additional discount function, safe state abstraction conditions for funnel actions under discounting can be weakened to those for the undiscounted case [11]. In the rest of this paper we will restrict ourselves to episodic MDPs with undiscounted value functions (stochastic shortest path problems).

3 Hierarchies of Abstract Models using HEXQ Partitions

A HEXQ partition for a factored MDP [8] attempts to find variable wise repetitive sub-state regions that have equivalent internal transition and reward functions and from which exits can be controlled. Formally, a ***HEXQ Partition*** G of the states factored by two variables, $s = (x, y)$, of a MDP into regions g is defined by the following conditions for all $g, g' \in G$

1. Same Context : all states in a region must have the same y label:
for all $s = (x, y),\ s' = (x', y') \in g$
$y = y'$

2. Markov transitions : all states with the same x label in different regions must have similar reward and transition functions to states inside their regions:
for all $a \in A$, $s = (x, y),\ s' = (x', y) \in g$ and $t = (x, y'),\ t' = (x', y') \in g'$
$T^a_{ss'} = T^a_{tt'}$ and $R^a_{ss'} = R^a_{tt'}$

3. Reachable exits : all exit states must be reachable within the region from all entry states with probability one. All possible starting states for the MDP are also entry states.

The transition and immediate reward functions are $T^a_{ss'} = Pr\{s'|s, a\}$ and $R^a_{ss'} = E\{r|s, a, s'\}$ respectively. With the states $s \in S$ partitioned into regions $g \in G$ an ***exit*** from region g is a state-action pair (s^o, a) such that taking action a from state $s^o \in g$ may reach a state not in g (or the MDP may terminate) in a single transition. State s^o is referred to as the ***exit state***. The state reached after exiting is called an ***entry state*** to the next region or a ***termination state*** from the current region, depending on our perspective.

The HEXQ partition ensures regions with the same set of x labels are isomorphic in the sense that they have equivalent internal Markov properties with respect to state transitions and rewards. Thus a first (sub)homomorphism allows the irrelevant y variable to be eliminated from each sub-task. HEXQ parameterises a set of sub-MDPs over the x label region classes so that each sub-MDP is allowed to reach only one possible exit state. A second

homomorphism allows x labelled region states to be safely aggregated at the next level because their exits can be controlled. The Cartesian product of the region classes and the y variable labels produce the state labels in a new abstract state space $S' = [s]_G$, hence $f : S \rightarrow [s]_G$. Together with the sub-MDP exit policies as abstract actions, this defines a safely abstracted SMDP. Condition (1) for a SMDP homomorphism is satisfied as the transition function from an exit state to sub-MDP termination states is independent of the starting states inside the sub-MDP. For condition (2) we note that the component of reward accumulated inside a sub-MDP is not the subject of the second homomorphism and remains untouched. To show $R' = R$ it is sufficient to show that the expected reward on exit of abstract action a after N steps, having starting in state s satisfies $R'_{exit}(s, a, N) = R_{exit}(f(s), g_s(a), N)$. As the sub-MDP exit state is independent of the starting state inside the sub-MDP and the discount factor is 1 this condition is also satisfied. The decomposition of the value function in HEXQ was inspired by MAXQ. The HEXQ partitioning can be applied recursively resulting in a hierarchy of sub-MDPs not unlike a MAXQ graph for which all of Ditterich's safe abstraction conditions hold.

The HEXQ decomposition produces a hierarchy of abstract models. The decomposed hierarchically optimal value function for state s in SMDP m at level e, where (abstract) action a invokes sub-MDP $m-1$ defined over region g, is given by

$$V^*_m(s) \quad = \quad \max_a [V^*_{m-1}(s) + E^*_m(g, a)] \tag{1}$$

At the lowest level $V^*_{m-1} = 0$ as all primitive actions are exit actions and do not invoke lower level routines. The ***HEXQ action-value function*** E^1 or exit value function for all states s in region g is the expected value of future rewards ***after completing*** the execution of the (abstract) action a exiting g and following the policy π thereafter. E includes the expected primitive reward on exit, but does not include any rewards accumulated in g.

$$E^*_m(g, a) = \sum_{s'} P^\pi(s'|g, a)[R_{exit} + V^*_m(s')] \tag{2}$$

$P^\pi(s'|g, a)$ is the probability of transition to state s' after (abstract) action a terminates from any state $s \in g$ and R_{exit} is the expected final primitive reward on transition to state s'.

The following examples illustrate the HEXQ decomposition of MDPs into a hierarchy of increasingly abstract models. Figure 1 (a) is the plan view showing two of 10 similar multi-room floors in a multi-storey building connected by two (east and west) banks of elevators. The state of this MDP is described by three variables; floors (labels 0-9), rooms (0-24) and locations in room (0-8). An agent uses one step actions to move north, south, east, west, up or down and starts on floor 0. The goal is to move to position "G" on floor 1 and execute the up action (to say turn the coffee maker on). The up/down actions only work at the elevator where they move the agent one floor at a time. The reward is -1 for taking each action. The HEXQ hierarchical decomposition for the building MDP generates two abstract SMDP. The first SMDP aggregates room locations into a single state and therefore has 250 states, one for each room, and 8 abstract actions, 4 to leave the room via the N,S,E or W and 4 to execute up or down at each elevator or "G" position. The top level SMDP has only 10 abstract states, one for each floor, and 9 abstract actions, moving to each elevator and pressing up or down and moving to the coffee machine and pressing up. The storage required to compute the value function for the original MDP is 13,500 values. The first safely abstracted model of the original problem requires 2216 values. The final safely abstracted model requires 1306, an order of magnitude saving.

[1]The notation E is used here in preference to Q used in [8] to avoid confusion with the normal Q function

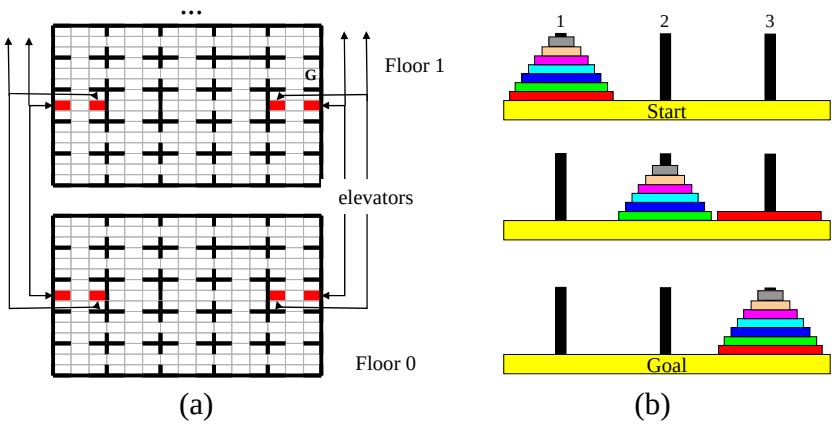

Figure 1: (a) The plan view of two out of ten floors connected via two banks of elevators, (b) The Towers of Hanoi Puzzle with 7 disks.

The second example, Figure 1 (b), shows a seven disk Tower of Hanoi (ToH) puzzle where the objective is to move the disc stack from peg 1 to peg 3, one disk at a time, but never with a lager disk on top of a smaller. This can be modelled as a MDP with 7 state variables encoding the peg position of each disk, 6 primitive actions indicating a disk move from one peg to another and a reward of -1 per disk move. The HEXQ decomposition of the ToH MDP has 7 levels, one for each disk in ascending order of size. Each level has three (abstract) states, three exit states and six exits. The ToH MDP is an example where the HEXQ hierarchical RL space complexity scales linearly in the number of variables, n. The action value function requires only $54n - 36$ values compared to the original MDP that requires 6×3^n values.

4 Variable Resolution Approximations

When space or time is constrained we still wish to find good solutions within the resources available. The hierarchy of abstract models generated by HEXQ suggests that further approximation, by varying the resolution of modelling, may improve the computational complexity over and above that achieved by safe state abstraction.

4.1 Variable Resolution Value Function

HEXQ uses a best first search[2] as a result of the recursion present in equation 1 to decide actions at execution time. Limiting the depth of the search approximates the value function, assuming that either (1) there is a diminishing return in finer detail and these costs can be ignored or (2) sub-tasks accumulate near constant internal reward during execution. In the latter case the constant accumulated reward within a sub-tasks can be added to the reward on exit, but other than linearly scaling the value function it will not effect the optimal policy. For example, in the extreme case with a depth limit of zero, equation 1 becomes $V^*_m(g) = \max_a E^*_m(g, a)$ reducing nicely to the usual "flat" Q learning representation for the most abstract approximation of the problem. It is important to note that limiting the search to a particular depth does not effect the ability to operate at more detailed levels. For example, at level 4 a depth of 2 searches to level 2 and at level 3 the search extends to level

[2] as does MAXQ [7]

1.

In the multi-storey building example, limiting the value function to top level values would result in choosing an arbitrary elevator to reach the first floor. In this case the agent may lengthen the journey by travelling to a non-optimal elevator bank. Increasing the depth of search to one level would result in an optimal path as the distance to each elevator bank is included in the decision. For the ToH puzzle, constraints ensure that the cost of an abstract action is constant at each level (given the disks are stacked on one peg to this level). This means that a depth zero search is sufficient to ensure an optimal policy, reducing the number of values that need to be searched in the 7 disk version from $6^7 = 279,936$ to only 6, more than 4 orders of magnitude saving! This is a huge reduction in decision time complexity.

4.2 Variable Resolution Exit States

The HEXQ decomposition generates one region sub-MDP for each hierarchical exit state. A *hierarchical* exit state means that the exit state is full resolution and distinguished at the base level of the hierarchy. This is necessary to ensure safe state abstraction. Limiting the resolution of the exit states reduces the number of sub-MDPs required. The effect is to reduce the number of separate policies for each region at the expense of possibly reaching each exit sub-optimally. The benefit is to reduce space complexity and learning time complexity. The loss of resolving exit states will lead in general to an overestimation of the value function as some rewards within the abstract exit state may be short-circuited.

For the multi-storey building, consider the floor region defined by the 25 abstract room states. Separate sub-MDPs are required for each of the four elevators, despite each of the two elevators sharing the same room. If we generate only one sub-MDP per elevator bank, that is, per abstract room state, the storage requirements to represent the value function are reduced from 1306 to 906. In this example there is no loss in accuracy of the value function, but in general the loss is limited to the intra-room distance.

4.3 Variable Resolution Exits

To allow safe state abstraction for funnel actions, HEXQ generates one abstract action for each sub-MDP exit, that needs to be represented and explored at the next highest level. Exits may also generate additional sub-MDPs if they have different hierarchical exit states.

A natural heuristic is to combine exits from a region when they always transition to the same next abstract state. Again the resolution can be adjusted to only combine exits between abstract state at a certain level of resolution. For the multi-storey problem, the two floor exits going up for each elevator bank could be combined as they always lead to the same next state (the room on the floor above). Similarly the down exits could be combined.

Combining exits makes it easier for sub-MDPs to exit and this will in general increase the internal value function of a sub-MDP. On the other hand there is an increased loss of control as exits cannot be discriminated at the next level. The net effect on the value function and resultant policy will depend on each specific problem instance and will need to be tested. It should be noted that the sub-MDP with combined exits may no longer be totally independent of the entry state. Nevertheless, combining exits can still provide a good approximation to the safe state abstraction produced by HEXQ.

The effect of combining elevator exits in the above building problem is to reduce the storage space requirements from 1306 to 866.

5 Kaelbling's 10×10 Maze Example

Kaelbling's introduced the maze, reproduced in figure 2, showing that it is possible to efficiently learn an approximate navigation policy to move to any goal at a small cost from optimality by using landmarks. In this maze the agent is required to navigate from a starting position to a goal position, both selected at random for each trial. The agent is assisted by 12 fixed landmarks (indicated by circles). The locations are partitioned into Voronoi regions indicating their closest landmark. The actions are to move north, south east or west. This task can be represented with the 3 state variables: agent-location (100 labels), goal-location (100 labels) and closest-landmark (12 labels). A dummy action is introduced to allow the agent to signal arrival at the goal. Primitive rewards are -1 for each action.

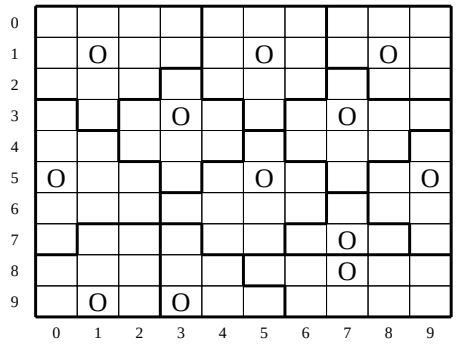

Figure 2: Kaelbling's 10×10 navigation maze. The regions represent the Voronoi partition given the circled landmarks.

Table 1 summarises the results in comparison to the flat learner's performance with deterministic actions for the maze. Storage is the number of Q or E values required by each algorithm to represent the value function. DFS is the average number of values that are searched to decide the next action. Stor% indicates the percentage of storage required in comparison with the flat learner. Steps are the average number of primitive actions to the goal. Regret is the deterioration in performance (steps to goal) compared to the optimum.

Table 1: HEXQ with variable resolution approximations solving Kaelbling's 10×10 maze.

Algorithm	**Storage**	**Stor%**	**DFS**	**Steps**	**Regret%**
Flat Q learning	40,000	100%	4	6.6	0%
HEXQ (safe state abstraction)	15,844	40%	52	6.6	0%
HEXQ VF depth 0	15,844	40%	15	7.5	14%
HEXQ state depth 0	5,108	13%	41	7.0	6%
HEXQ combining exits	9,644	24%	20	7.4	12%

For 'HEXQ VF depth 0' the value function was only evaluated at the highest level of abstraction and the value inside subtasks ignored. The decision time (DFS) is reduced by

a factor of more than 3 at the expense of 14% regret. This approximation ignores the exact position of the agent and goal and only considers their Voronoi region locations. There is no further saving in storage space.

For "HEXQ state depth 0" sub-MDPs are combined for all exits with the same depth 0 abstract exit state. This reduces the number of sub-MDPs required at level 2 from 100 to 12 and reduces storage requirements significantly. If the agent is in a goal region it will take the optimal path to the goal by following the local policy of the goal exit sub-MDP. If the agent is in a region next to the goal region then the exit values for each exit will be averaged over all the possible goal locations in the adjoining region because this approximation has taken away the power to discriminate between them. The optimum average value will be associated with that exit that minimised the average distance to the goal after exit. This exit will be in the middle of the boundary between the two Voronoi regions. Interestingly, the effect is similar to Kaelbling's original idea, that the agent should aim for the landmark of its next closest region on the way to the goal. This approximation gives the smallest regret (6%) and achieves the best reduction in storage (to 13%) for this problem.

Combining exits in the maze means that all possible transitions between adjacent Voronoi regions are combined. The number of inter-region exits is reduced from 122 to 46. At level 2 in the hierarchy the number of abstract actions is reduced accordingly. Overall the storage requirements reduced from 40% to 24% at the cost of 0.8 increase in average steps to the goal. The reason for the deterioration can be seen, for example, if the agent is right next to a goal that is in a neighbouring Voronoi region. As it cannot discriminate between exits leading to the goal region, it may choose one that lands the agent two moves away from the goal. The agent then needs to backtrack to the goal.

6 Discussion and Future Work

For practical applications of hierarchical reinforcement learning it is important to make decisions in a realistic time frame. We have presented approximations using variable resolution on a hierarchy of abstract models that can deliver any-time solutions. For example, the resolution depth of the search in compiling the value function can be performed by iterative deepening, interrupting the deliberation process when the extra cost of search is estimated to exceed the additional returns.

Variable resolution heuristics cannot give any optimality guarantees, but then we cannot make any guarantees in general for practical MDP decompositions. Some estimate of solution quality in relation to a hierarchical optimal solution may be possible for the heuristics by measuring the limits on abstract transition probabilities and sub-task rewards and employing bounded parameter MDPs [12].

HEXQ finds globally optimal solutions for deterministic factored MDPs. This would allow some stochastic problems to be decomposed and approximated by deterministic optimal policies. A clearer definition of conditions and limits on the accuracy of these various approximation techniques is left for future research.

To scale up reinforcement learning to real world applications, efficient practical representations will be required. This article has presented and demonstrated a number of variable resolution approximations to reduce the space and time complexity for HEXQ decomposed MDPs.

References

[1] Leslie Pack Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Machine Learning Proceedings of the Tenth International Conference,*

pages 167–173, San Mateo, CA, 1993. Morgan Kaufmann.

[2] Ross Ashby. *Introduction to Cybernetics*. Chapman & Hall, London, 1956.

[3] Thomas Dean and Robert Givan. Model minimization in markov decision processes. In *AAAI/IAAI*, pages 106–111, 1997.

[4] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[5] Ronald E. Parr. *Hierarchical Control and learning for Markov decision processes*. PhD thesis, University of California at Berkeley, 1998.

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

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

[8] Bernhard Hengst. Discovering hierarchy in reinforcement learning with HEXQ. In Claude Sammut and Achim Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 243–250. Morgan-Kaufman, 2002.

[9] David Andre and Stuart J. Russell. State abstraction for programmable reinforcement learning agents. In Rina Dechter, Michael Kearns, and Rich Sutton, editors, *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 119–125. AAAI Press, 2002.

[10] Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi markov decision processes. In *To appear in the Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 03)*, 2003.

[11] Bernhard Hengst. Safe state abstraction and discounting in hierarchical reinforcement learning. Technical Report UNSW CSE TR 0308, National ICT Australia, School of Computer Science and Engineering, University of New South Wales, Sydney NSW Australia, May 2003.

[12] Robert Givan, Sonia M. Leach, and Thomas Dean. Bounded-parameter markov decision processes. *Artificial Intelligence*, 122(1-2):71–109, 2000.