IJCSE International Journal of Computer Science and Engineering Open Access
Survey Paper Volume-2, Issue-5 E-ISSN: 2347-2693

Hierarchical Reinforcement Learning in Complex Learning Problems: A Survey

Samiksha Mahajan*
Dept. of Information Technology and Computer Science, VVES's Vikas College, Mumbai 400 083, India

www.ijcseonline.org

Received: 07/04/2014 Revised: 03 /05/2014 Accepted: 22/05/ 2014 Published: 31 /05/2014

Abstract — Reinforcement Learning (RL) is an active research area of machine learning research based on the mechanism of learning from rewards. RL has been applied successfully to variety of tasks and works well for relatively small problems, but as the complexity grows, standard RL methods become increasingly inefficient due to large state spaces. This paper surveys Hierarchical Reinforcement Learning (HRL) as one of the alternative approaches to cope with issues regarding complex problems and increasing the efficiency of reinforcement learning. HRL is the subfield of RL that deals with the discovery and/or exploitation of underlying structure of a complex problem and solving it using reinforcement learning by breaking it up into smaller sub-problems. This paper gives an introduction to HRL, discusses its basic concepts, different algorithms, approaches and related work regarding Hierarchical Reinforcement Learning. At last but not the least this paper briefly gives variation between flat RL and HRL following its pros and cons. It concludes with research scope of HRL in complex problems.

Keywords/Index Term—*Machine Learning; Reinforcement Learning; Hierarchical Reinforcement Learning*

I. INTRODUCTION

Reinforcement Learning is a type of Machine Learning, inspired by behaviorist psychology and one of the most active research areas in Artificial Intelligence. It is a computational approach to learning where an agent tries to maximize the total amount of reward it receives when interacting with a complex, uncertain environment. [1][2] RL is successfully studied in many other disciplines such as Game theory, Control theory, Operation research, Robotics, Economics, Information theory, Simulation based optimization, Statistics and Genetic algorithms.

Current line of research is to establish highly data efficient and robust RL algorithms that are able to solve complex problems. In such complex problems, Curse of Dimensionality [3] refers to the exponential increase in the state-space with each additional variable or dimension that describes the problem. Due to the table structure of state action pairs, with large state spaces, reinforcement learning becomes increasingly inefficient, which presents a major obstacle in successfully applying RL to complex domains. Learning in environments with extremely large state spaces is infeasible without some form of generalization.

A method for dealing with this is Hierarchical Reinforcement Learning. It is an approach to reinforcement learning which splits the global goals of a reinforcement learning agent up into smaller subgoals, and then attempts to tackle each subgoal separately. By doing this the state space is decreased and therefore the efficiency increased. Thus hierarchical reinforcement learning is mentioned as a good way of increasing efficiency in reinforcement learning.[2]

II. CONCEPTS RELATED TO HIERARCHICAL REINFORCEMENT LEARNING

Complex problems cannot be described by only a few variables, but fortunately the real world is highly structured with many constraints and with most parts independent of most other parts. Without structure it would be impossible to solve complex problems of any size [4]. Structure can significantly reduce the naive state space generated by sheer enumeration. Reinforcement learning is concerned with problems represented by actions as well as states, generalizing problem solving to systems that are dynamic in time. Hence we often refer to reinforcement learning problems as tasks, and sub-problems as sub-tasks [5]. Below are the concepts related to hierarchical reinforcement learning.

- **Hierarchical Decomposition**

Many large problems have hierarchical structure that allows them to be broken down into sub problems. The sub problems, being smaller are solved more easily. The solutions to the sub problems are recombined to provide the solution for original large problem. A heuristic from [6] for problem solving is decomposing and recombining or divide and conquer in today's parlance.

Decomposition can make finding the final solution significantly more efficient with improvements in the time and space complexity for both learning and execution. Many times similar tasks need to be executed in different contexts. If reinforcement learner is unaware of underlying structure, it would relearn the same task in multiple different contexts, when it is clearly better to learn the task once and reuse it. In [8] a hierarchical reinforcement learning approach was used to create a program which plays a board game called 'The Settlers of Catan', which is a popular modern board game. It is a very complex board game and therefore a flat reinforcement learning approach would have been inefficient. In their approach they use hierarchical reinforcement learning and model trees for value function approximation. Both Q-learning and SARSA are used as the conventional reinforcement learning algorithms at different stages of the learning process.

- **Semi Markov Decision Process (SMDP)**

In reinforcement learning environments, states are often assumed to fulfill the Markov property. MDP is the probabilistic model of a sequential decision problem, where states can be perceived exactly, and the current state and action selected determine a probability distribution on future states. The model is Markov if the state transitions are independent of any previous environment states or agent actions. [2]

SMDP is a generalization of the Markov Decision Process (MDP) where the times between transitions are allowed to be random variables whose distribution may depend upon the current state, the action taken, (possibly) the next state. An SMDP can be seen as representing the system at decision points, while MDP represents the system at all times. Semi-Markov decision processes (SMDPs) serve as theoretical basis for many hierarchical RL approaches developed during the last decay. Hierarchical reinforcement learning is a sample-based framework for solving SMDPs that finds the "best" policy consistent with a given hierarchy [8]. These algorithms enable agents to construct hierarchical policies that allow using multi-step actions as "subroutines." HRL algorithms can be described using the SMDP framework.

An SMDP is defined as a tuple M=(S, A, P, R). S is the set of states, and A is the set of actions the agent may take at a decision point. P is a transition probability function, where P(s, N|s, a) denotes the probability that action a taken in state s will cause a transition to state s in N time steps. Rewards can accumulate over the entire duration of an action. The reward function R(s, N|s, a) is the expected reward received from selecting action a in state s and transitioning to state s with a duration of N time steps.
The use of hierarchical actions transforms a Markov Decision Process (MDP) into a Semi-Markov Decision Process (SMDP) and that convergence results still hold for the learning algorithms known to converge in the absence of hierarchy. [9, 10, 11, 12]

- **State Abstraction**

The method for dividing up the global state space of a problem into smaller state spaces is given in [13]. It describes how a problem's state space can be divided up into a series of intervals if the states in the same interval block have the same properties in terms of transitions and rewards. Decomposing large MDPs lead to state abstraction, thus reduce the overall state space of the given problem and thus reduce learning time. An abstracted state space is smaller than the state space of an original MDP. There are broadly two kinds of conditions under which state-abstractions can be introduced [5, 14]. They are given as follow:

- Eliminate irrelevant variables: When reinforcement learning algorithms are given redundant information, they will learn the same value-function or policy for all the redundant states. Thus eliminating irrelevant variables leads to state abstraction. This type of state abstraction is used by [15] who exploit independence between variables in dynamic Bayesian nets. It is introduced by [14] for Max Node and Leaf Irrelevance in the MAXQ graph [16] extend this form of model minimization to state-action symmetry couched in the algebraic formations of morphisms. They state conditions under which the minimised model is a homomorphic image and hence transitions and rewards commute. Some states may not be reachable for some sub-tasks in the task-hierarchy. This Shielding condition [14] is another form of elimination of irrelevant variables, in this case resulting from the structure of the task-graph.
  Funnelling: Funneling is a type of state abstraction where abstract actions move the environment from a large number of initial states to a small number of resulting states.

The effect is exploited for example by [18] in plant control and by [19] to reduce the size of the MDP.

- **Abstract Actions**

Approaches to HRL employ actions that persist for multiple time-steps. These temporally extended or abstract actions hide the multi-step state-transition and reward details from the time they are invoked until termination. Abstract actions are employed in many fields including AI, robotics and control engineering. They are similar to macros in computer science that make available a sequence of instructions as a single program statement. Abstract actions may execute a policy for a smaller MDPs or SMDPs. This establishes a hierarchy where a higher-level parent task employs child subtasks as its abstract actions, which is called as Task Hierarchies [14].

In hierarchical problem solving, the decomposed subproblem can be solved by abstract actions. It can be realized by the set of primitive actions to reach the subgoal. Thus in hierarchical reinforcement learning the corresponding action policy can be learned based on the abstract states identified subgoals and abstract actions [17].

- **Optimality**

HRL cannot guarantee optimal solution to decomposed problems. Types of optimality in HRL can be given as follows:

- Hierarchical Optimality

The overall learned policy is the best policy consistent with the hierarchy. Policies that are hierarchically optimal are ones that maximize the overall value function consistent with the constraints imposed by the task-hierarchy.

- Recursive Optimality

In Recursive optimality, sub-task policies to reach goal terminal states are context free ignoring the needs of their parent tasks. Here the sub-tasks can be re-used in various contexts, but they may not be optimal in each situation.

- Hierarchical Greedy Optimality

In Hierarchical Greedy Optimality, a subtask policy proceeding to termination may be sub-optimal and by constantly interrupting the sub-task a better sub-task may be chosen. It is better than hierarchical and recursive optimality but does not guarantee of global optimality.

III. APPROACHES FOR HRL

Hierarchical reinforcement learning involves breaking the target Markov decision problem into a hierarchy of sub problems or subtasks. There are three general approaches to defining these subtasks i.e three prominent HRL approaches are: Options, a formalization of abstract actions; HAMQ, a partial program approach, and MAXQ value function decomposition including state abstraction. While choosing the appropriate hierarchy, we have to look at available domain knowledge: If some behaviors are completely specified – use Options; if some behaviors are partially specified – use HAM; if less domain knowledge available – use MAXQ. We can use all three to specify different behaviors in tandem.

- **Options**

Options approach is to define each subtask in terms of a fixed policy that is provided by the programmer (or that has been learned in some separate process).

An option is a triple $< I, \pi, \beta >$ in which, $I \subseteq S$ is an initiation set, β: $S^+ \rightarrow [0, 1]$ is a termination condition, and π: $S \times A \rightarrow [0, 1]$ is a policy [20]. An option $< I, \pi, \beta >$ is available in state s_t if and only if $s_t \in I$. If the option is taken, then actions are selected according to π until the option terminates stochastically according to β. In particular, a Markov option executes as follows. First, the next action a_t is selected according to probability distribution π $(s_t, \cdot)$. The environment then makes a transition to state s_{t+1}, where the option either terminates, with probability β (s_{t+1}), or else continues, determining a_{t+1} according to π $(s_{t+1}, \cdot)$, possibly terminating in s_{t+2} according to β (s_{t+2}), and so on. When the option terminates, the agent has the opportunity to select another option. An option is temporally extended action with well defined policy. When option policies and termination depend on only the current state s, options are called Markov options. The option policy and termination depends on the entire history sequence of states, actions and rewards since the option was initiated. Options of this sort are called semi-Markov options. Options can only input complete policy and requires complete specification of policy.

- **HAMQ (Hierarchical Abstract Machine)**

In the hierarchy of abstract machines (HAM) approach to HRL the designer specifies abstract actions by providing stochastic finite state automata called abstract machines that work jointly with the MDP [9].

A HAM is a program which, when executed by an agent in an environment, constrains the actions that the agent can take in each state. An abstract machine is a triple $<\mu; I; \delta)$ where μ is a finite set of machine states, I is a stochastic function from states of the MDP to machine states that determines the initial machine state, and δ is a stochastic next-state function, mapping machine states and MDP states to next machine states [5]. HAMs are a way to partially specify procedural knowledge to transform an MDP to a reduced SMDP. Extended HAM approach is given by introducing more expressible agent design languages for HRL (Programmable HAM -PHAM) [22] and ALisp, a Lisp-based high-level partial programming language [23].

- **MAXQ**

The third approach MAXQ is to define each subtask in terms of a termination predicate and a local reward function [21]. MAXQ algorithm expects the hierarchical structure to be supplied by the designer. It suggests breaking the main problem's value function up into an additive combination of smaller value functions, each associated with a smaller problem i.e. MAXQ represents the value of a state as a decomposed sum of sub-task completion values plus the expected reward for the immediate primitive action. A completion value is the expected cumulative reward to complete the sub-task after taking the next abstract action. Detailed overview of MAXQ is given [24]. The MAXQ algorithm is tested against flat Q-Learning and significantly outperforms it. Function approximation and optimistic exploration are combined to allow MAXQ to cope with large and even infinite state spaces [35].

IV. ALGORITHMS AND RELATED WORKS IN HRL

There are several important design decisions that must be made when constructing a hierarchical reinforcement learning system. [21]

- *Specify subtasks*

Hierarchical reinforcement learning involves breaking the target Markov decision problem into a hierarchy of sub problems or subtasks. The three general approaches to defining these subtasks are Options, Ham and MAXQ as mentioned in above section III.

- *Employ state abstractions within subtasks*

The second design issue is whether to employ state abstractions within subtasks. A subtask employs state abstraction if it ignores some aspects of the state of the environment.

- *Non-hierarchical execution*

The third design issue concerns the non-hierarchical execution of a learned hierarchical policy. Extra learning i.e. in all states (and at all levels of the hierarchy) is required, in order to support this non-hierarchical execution. In HRL, the only states where learning is

required at the higher levels of the hierarchy are states where one or more of the subroutines could terminate.

- *Learning algorithm*

The final issue is what form of learning algorithm to employ. Finding online algorithms that work for general hierarchical reinforcement learning has been difficult, particularly within the termination predicate family of methods. Some relied on each subtask having a unique terminal state[30]; Some employed a mix of online and batch algorithms to train her hierarchy[31]; and work within the options framework usually assumes that the policies for the sub problems are given and do not need to be learned at all.

Some important algorithms in HRL are as given below:

- **H-DYNA Algorithm**

DYNA is a reinforcement learner that uses both real and simulated experience after building a model of the reward and state transition function. A gating mechanism called Hierarchical-DYNA (H-DYNA) was developed [30] which first learns elementary tasks such as to navigate to specific goal locations. Each task is treated as an abstract action at a higher level of control. H-DYNA differs from hierarchical planners in two ways: first, the abstract models are learned using experience gained while learning to solve other tasks in the same environment, and second, the abstract models can be used to solve stochastic control tasks.

- **HEXQ**

The HEXQ (hierarchical exit Q function) approach is a series of algorithms motivated by MAXQ value-function decomposition and bottom-up structure learning [25]. It has ability to abstract tasks dynamically. HEXQ automatically builds task-hierarchies from interactions with the environment assuming an underlying finite state factored MDP. It employs both variable elimination and funnel type state abstractions to construct a compact representation. As one subtask is learnt for each exit, HEXQ solutions are hierarchically optimal. HEXQ can use hierarchical greedy execution to try to improve on the hierarchically optimal solution, and the HEXQ policy converges to the globally optimal result for task-hierarchies were only the root-subtask has stochastic transition or reward functions. The introduction of parallel decomposition of variables and sequential actions allows HEXQ to create abstract state variables not supplied by the designer. It conveys safe decomposition [25]. In more recent versions, state variables are tackled in parallel [26].

[36] propose a Bayesian network model causal graph based approach - Variable Influence Structure Analysis (VISA) that relates the way variables influence each other to construct the task-hierarchy [27]. Unlike HEXQ this algorithm combines variables that influence each other and ignores lower-level activity.

- **HDG (Hierarchical Distance to Goal) Algorithm**

HDG learning algorithm [31] uses a hierarchical decomposition of the state space to make learning to achieve goals more efficient with a small penalty in path quality and introduces the important idea of composing the value function from distance components along the path to a goal. It uses hierarchical approach to solving problems when goal of achievement are given to the agent dynamically. It is descendent of Watkins' Q-learning algorithm.

HDG algorithm was extended to automatically generate hierarchies in goal directed systems [33]. But again none of these systems focused on discovering subgoals in the state space to facilitate hierarchy.

- **HABS(Hierarchical Assignment of Behaviors by Self-organizing)**

The HABS algorithm [32] is derived mainly from HASSLE. It was developed to overcome the *action explosion* problem in HASSLE and to allow neural networks to be used as function approximator for the high level policy. If this feature could be dropped and replaced by something that always uses a fixed set of behaviors as high level actions, the number of high level actions would remain constant when the problem size grows. That way the useful aspects of HASSLE will retain, like using the abstracted state space and starting with a priori uncommitted subpolicies.

- **Feudal Reinforcement Learning**

The feudal reinforcement learning algorithm [34] emphasizes state abstraction. It is referred as information hiding and involves hierarchy of learning modules. It has the idea that decision models should be constructed at coarser granularities further up the control hierarchy.

- **HASSEL Algorithm**

HASSLE algorithm [28] outperformed variants of plain RL in deterministic and stochastic versions of a large MDP. It discovers subgoals and the corresponding specialized subtask solvers by learning how to transition between abstract states. In the process subgoal abstract actions are generalized to be reused or specialized to work in different parts of the state-space or to reach different goals. HASSEL is extended with function approximation [29].

Some recent work in HRL is discussed below:

A method for learning a hierarchy of actions in a continuous environment is given [37] by learning a qualitative representation of the continuous environment and then and actions to reach qualitative states.[38] presents methods that learn to parameterize and order a set of motion templates to form abstract actions (options) in continuous time. A skill discovery method for reinforcement learning in continuous domains is introduced [39] that constructs chains of skills leading to an end-of-task reward. The method is further developed to build skill trees faster from a set of sample solution trajectories.

Automatic basis function construction to HRL extended [40]. The approach is based on hierarchical spectral analysis. Hierarchical Approaches of graphs induced on an SMDP's state space from sample trajectories. Brief review of more recent progress in general representation discovery is provided in [41]. The review includes temporal and homomorphic state abstractions, with the latter generalized to representing value functions abstractly in terms of a basis functions. A hierarchical reinforcement learning method based on action sub rewards is proposed which can reduce state spaces greatly and choose actions with favorable purpose and efficiency so as to optimize reward function and enhance convergence speed [42]. The approach is given [43] to optimizing Natural Language Generation for situated interactions using HRL with Bayesian Networks. SHARSHA contributes to research in hierarchical reinforcement learning. It supports convergent policy learning within hierarchical reactive plans, while other convergent methods rely on more constrained representations and a non-interruptible execution model [48].

Some real world applications where HRL is implemented successfully are Toy Robot [44], Flight Simulator [45], AGV Scheduling [46], Keepaway soccer [47].

V. FLAT RL AND HRL

The main reason for introducing hierarchical architectures in RL is to bridge the gap between theoretical considerations to machine intelligence and practical application to real-world problems. In large-scale problems the performance of flat RL is too poor with regard to learning speed and computational effort. It also scales badly with the size of state and action-spaces and the length of the sequence of actions to reach the terminal state.

Advantages of HRL over flat RL:

a) Improved exploration as it can take big steps at high levels of abstraction.
b) Learning from fewer trials as fewer parameters must be learned and subtasks can ignore irrelevant features of full state.
c) Faster learning for new problems because subtasks learned on previous problems can be reused
d) Allows transfer at multiple levels of hierarchy, which can speed up learning.
e) Task decompositions are helpful in reducing the size of the problem, and therefore in exorcising the Curse of Dimensionality

Limitations of HRL:

a) HRL automatically handles exploration-exploitation shift but at the same time it lacks in sufficient exploration and sufficient subtlety.
b) Some of the existing algorithms only work well for the problem which they were designed to solve.

VI. CONCLUSIONS AND FUTURE SCOPE

Hierarchical reinforcement learning is a key to scaling reinforcement methods to large, complex, real world problems. Complex learning task can be broken down into several smaller subtasks in multiple levels of hierarchy, by using appropriately formulated background knowledge. So that it can be learned more quickly, easily and then recombined to solve complex large problems.

While choosing the appropriate hierarchy approach, we need to look up the domain knowledge. If some behaviors are completely specified –use Options, if some behaviors are partially specified – use HAM and if less domain knowledge available – use MAXQ approach. We can use all three to specify different behaviours in system.

HRL research must give more leverage to automated discovery of hierarchy and state abstraction. Exploration must improve before attempts to learn simultaneously at multiple levels of hierarchy.

Even applied successfully on complex problems, HRL research needs effectiveness (in terms of optimality, and choosing right options) on more large and complex continuous control tasks and real world problems. HRL research must give more leverage to automated discovery of hierarchy and state abstraction. Also exploration must improve before attempts to learn simultaneously at multiple levels of hierarchy.

VII. REFERENCES

[1] Richard Sutton and Andrew Barto (1998).*Reinforcement Learning*.MIT Press.ISBN 0-585-02445-6.
[2] Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement learning: A survey." arXiv preprint cs/9605103 (1996).
[3] Richard Ernest Bellman (1961). *Adaptive control processes: a guided tour*. Princeton University Press.
[4] Rusell, Stuart, and Peter Norvig. "Artificial intelligent: A modern approach." (2003).
[5] Hengst, Bernhard. "Hierarchical approaches." *Reinforcement Learning*. Springer Berlin Heidelberg, 2012. 293-323.
[6] Polya G (1945) How to Solve It: A New Aspect of Mathematical Model. Princeton University Press
[7] Pfeiffer, M (2004). Reinforcement Learning of Strategies for Settlers of Catan. Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education, Reading, UK. November 2004
[8] A. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Special Issue on Reinforcement Learning, Discrete Event Systems Jouranl*, 13:41–77, 2003.
[9] Parr, R. and Russell, S. (1997). Reinforcement learning with hierarchies of machines. In Proceedings of Advances in Neural Information Processing Systems 10. MIT Press.

[10] Parr, R. (1998). Hierarchical Control and Learning for Markov Decision Processes. PhD thesis, University of California at Berkeley.
[11] Precup, D. and Sutton, R. S. (1997). Multi-time models for temporally abstract planning. In Proceedings of Advances in Neural Information Processing Systems 10. MIT Press.
[12] Precup, D., Sutton, R. S., and Singh, S. (1998). Theoretical results on reinforcement learning with temporally abstract behaviors. In Proceedings of the Tenth European Conference on Machine Learning, ECML'98. Springer-Verlag.
[13] Asadi, M and Huber, M (2004). State Space Reduction for Hierarchical Reinforcement Learning. In Proceedings of the 17th International FLAIRS Conference, pp. 509 - 514, Miami Beach, FL. 2004 AAAI
[14] Dietterich TG (2000) Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research 13:227- 303
[15] Boutilier C, Dearden R, Goldszmidt M (1995) Exploiting structure in policy construction. In: Proceedings of the 14th international joint conference onArtificial intelligence -Volume 2, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 1104-1111
[16] Ravindran, Balaraman. "SMDP homomorphisms: An algebraic approach to abstraction in semi markov decision processes." (2003).
[17] Chiu, Chung-Cheng, and Von-Wun Soo. "Subgoal identification for reinforcement learning and planning in multiagent problem solving." *Multiagent System Technologies*. Springer Berlin Heidelberg, 2007. 37-48.
[18] Forestier, J-P., and Pravin Varaiya. "Multilayer control of large Markov chains."*Automatic Control, IEEE Transactions on* 23.2 (1978): 298-305.
[19] Dean, Thomas, and Shieu-Hong Lin. "Decomposition techniques for planning in stochastic domains." *IJCAI*. Vol. 2. 1995.
[20] Sutton, Richard S., Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning."*Artificial intelligence* 112.1 (1999): 181-211.
[21] Dietterich, Thomas G. "Hierarchical reinforcement learning with the MAXQ value function decomposition." *arXiv preprint cs/9905014* (1999).
[22] Andre D, Russell SJ (2000) Programmable reinforcement learning agents. In: Leen TK, Dietterich TG, Tresp V (eds) NIPS, MIT Press, pp 1019-1025
[23] Andre D, Russell SJ (2002) State abstraction for programmable reinforcement learning agents. In: Dechter R, Kearns M, Sutton RS (eds) Proceedings of the Eighteenth National Conference on Artificial Intelligence, AAAI Press, pp 119-125
[24] Dietterich, Thomas G. "An overview of MAXQ hierarchical reinforcement learning." *Abstraction, Reformulation, and Approximation*. Springer Berlin Heidelberg, 2000. 26-44.
[25] Hengst, Bernhard. *Discovering hierarchy in reinforcement learning*. University of New South Wales, 2003.
[26] Hengst B (2008) Partial order hierarchical reinforcement learning. In: Australasian Conference on Artificial Intelligence, pp 138-149
[27] Jonsson A, Barto AG (2006) Causal graph based decomposition of factored mdps. In Journal of Machine Learning, vol 7, pp 2259-2301
[28] Bakker B, Schmidhuber J (2004) Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In: Proceedings of the 8-th Conference on Intelligent Autonomous Systems, IAS-8, pp 438-445
[29] Moerman W (2009) Hierarchical reinforcement learning: Assignment of behaviors to sub-policies by self-organization. PhD thesis, Cognitive Artificial Intelligence, Utrecht University
[30] Singh, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. Machine Learning, 8, 323-339.
[31] Kaelbling, L. P. (1993). Hierarchical reinforcement learning: Preliminary results. In Proceedings of the Tenth International Conference on Machine Learning, pp. 167{173 SanFrancisco, CA. Morgan Kaufmann.
[32] Moerman, Wilco, and Cognitive Artificial Intelligence. *Hierarchical reinforcement learning: Assignment of behaviours to subpolicies by self-organization*. Diss. PhD thesis, Cognitive Artificial Intelligence, Utrecht University, 2009.
[33] Moore, A. W., Baird, L. C., and Kaelbling, L. (1999). Multi-value-functions: Efficient automatic action hierarchies for multiple goal MDPs. In Dean, T., editor, Proceedings of the International Joint Conference on Artificial Intelligence *(IJCAI99)*, volume 2, pages 1316–1321, San Francisco, CA. Morgan Kauffman Publishers, Inc.
[34] Dayan P, Hinton GE (1992) Feudal reinforcement learning. Advances in Neural Information Processing Systems 5 (NIPS)
[35] Jong NK, Stone P (2009) Compositional models for reinforcement learning. In: The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases
[36] Jonsson A, Barto AG (2006) Causal graph based decomposition of factored mdps. In Journal of Machine Learning, vol 7, pp 2259-2301
[37] Mugan J, Kuipers B (2009) Autonomously learning an action hierarchy using a learned qualitative state representation. In: Proceedings of the 21st international joint conference on Artificial intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 1175-1180
[38] Neumann G, Maass W, Peters J (2009) Learning complex motions by sequencing simpler motion templates. In: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, New York, NY, USA, ICML '09, pp 753-760
[39] Konidaris G, Barto AG (2009) Skill discovery in continuous reinforcement learning domains using skill chaining. In: Bengio Y, Schuurmans D, Lafferty J, Williams CKI, Culotta A(eds) Advances in Neural Information Processing Systems 22, pp 1015-1023
[40] Osentoski S, Mahadevan S (2010) Basis function construction for hierarchical reinforcement learning. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1, International Foundation for Autonomous

Agents and Multiagent Systems, Richland, SC, AAMAS '10, pp 747-754

[41] Mahadevan S (2010) Representation discovery in sequential descision making. In 24thConference on Artificial Intelligence (AAAI), Atlanta July 11-15 2010

[42] Fu, Yuchen, et al. "A Reward Optimization Method Based on Action Subrewards in Hierarchical Reinforcement Learning." *The Scientific World Journal* 2014 (2014).

[43] Dethlefs, Nina, and Heriberto Cuayáhuitl. "Combining hierarchical reinforcement learning and Bayesian networks for natural language generation in situated dialogue." *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, 2011.

[44] Kober, Jens, Erhan Oztop, and Jan Peters. "Reinforcement learning to adjust robot movements to new situations." *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*. AAAI Press, 2011.

[45] Ryan, Malcolm, and Mark Reid. "Learning to fly: An application of hierarchical reinforcement learning." *In Proceedings of the 17th International Conference on Machine Learning*. 2000.

[46] Ghavamzadeh, Mohammad, and Sridhar Mahadevan. "Hierarchical average reward reinforcement learning." *The Journal of Machine Learning Research* 8 (2007): 2629-2669.

[47] Stone, Peter, Richard S. Sutton, and Gregory Kuhlmann. "Reinforcement learning for robocup soccer keepaway." *Adaptive Behavior* 13.3 (2005): 165-188.

[48] Shapiro, Daniel, Pat Langley, and Ross Shachter. "Using background knowledge to speed reinforcement learning in physical agents." *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001.

AUTHOR PROFILE

Samiksha Mahajan received her M.Sc and M.Phil in Computer Science from Nagpur University. She has teaching experience at UG and PG level for computer science and IT. Research interests: Artificial Intelligence, Machine Learning.