10

Subgoal Identifications in Reinforcement Learning: A Survey

Chung-Cheng Chiu and Von-Wun Soo
National Tsing Hua University
Taiwan

1. Introduction

Designing an algorithm to build a program to solve all forms of problems is an attractive idea. The programmers don't need to spend efforts on figuring out an optimal way to solve the problem, the algorithm itself will explore the problem and automatically finds the solution to the problem. This amazing feature of automatic programming is what makes reinforcement learning so appealing, and have the potential to apply on virtually every single task of our world. Reinforcement learning is a general framework to find an optimal solution for the given task. Its generalization reduces the effort of programmers on mapping a specific task into a reinforcement learning problem, but this feature is also the main performance bottleneck of reinforcement learning. Since there are not many constraints within the framework, the search space of the policy is exponentially proportional to the dimension of the state space. When mapping a high dimensional problem into a reinforcement learning problem, the conventional reinforcement learning algorithm becomes infeasible. Most of the real world problems are high-dimensional, and it is the major limitation for reinforcement learning. Therefore, how to find a way to reduce the search space and improve the search efficiency is the most important challenge.

On dealing with a high dimensionality problem, there are two common approaches to improve the performance. One is to reduce the dimensionality; the other is to find a better optimization algorithm. The first approach has drawn much attention in recent years, and many interesting works have been proposed this way to boost the performance of reinforcement learning. This article is going to review some recent advances in the dimensionality reduction approach.

Approaches for dimensionality reduction can be classified into two categories. One is value function approximation, and the other is abstraction. The first category approximate the utility function as a specific form of function, usually linear function, and the optimization of value function becomes much more efficient when dealing with linear functions. This approximation not only reduces the cost of optimization, but also provides the abilities to dealing with continuous-variable problems and generalizing the original policy to similar problems. The second category identifies the structure of the problem, represent the problem with higher-level concepts, and remove irrelevant parameters. Approaches within this category identify the sub-spaces of the state space that do not need to include other sub-spaces when solving the problem. For example, on solving a "make-coffee" problem, we can divide the problem into two sub-problems "reach the coffee maker" and "cook coffee with

coffee maker". This article is going to talk a little bit on the value function approximation, and the main focus is to review some recent advances in abstractions.

Abstraction approaches require determining sub-problems, and the sub-problems are also known as *subtasks*. On representing the original problem with subtasks, the program then can make decision among these subtasks instead of low-level actions. The new representation scales down the complexity of the problem. The problems remain to solve are how to identify these subtasks and how to learn policies of them. The second task depends on the first task: we need to construct subtasks so that to learn their policies. One way to implement this framework is to construct subtasks and define sub-policies manually. When a problem is divided into several sub-problems, it is usually easy for a programmer to implement the sub-policies, so the approach can be practical for simple problems. But when dealing with large-scale decision making problems, programming these sub-policies becomes effort demanding, and identifying a good set of subtasks is difficult if not impossible. This leads to the issue of how to identify these subtasks. Many algorithms have been proposed to attack this problem, but the complexity of these algorithms is proportional to the complexity of the problem, so they become impractical for challenging problems. How to automate these processes remains an open problem.

Even subtasks are constructed, the sub-policy learning problem is still critical. Adding sub-tasks to the original problem actually increases the complexity, so simply applying the conventional reinforcement learning algorithms provides no benefit. Except for implementing the sub-policies manually, the benefit of constructing subtasks comes from the potential for dimension reductions. (1) A subtask can be shared among solving different sub-problems and can save the time for learning redundant subtasks. (2) It is much easier to find irrelevant parameters within a subtask. An object can be relevant in one task and irrelevant in others; separates the problem into different subtasks allow the specific task to remove the parameters for that object. Removing irrelevant parameters reduces the dimensionality of the problems. Therefore, despite of subgoal identification issue, we will also talk about dimension reductions among subtasks.

The second section introduces existing frameworks for representing subtasks. The third section reviews existing works for subtask discovery. The fourth section discusses the issue of dimension reduction among subtasks. And the conclusions are given in the last section.

2. Subtask representation

We start the introduction for subtask representation from a widely used model, the option framework Sutton et al. (1999). The option framework defines a subtask as an option. Each option consists of three components, initiation set, terminal conditions, and the option policy function. The initiation set determines what states the option can be applied, the terminal conditions specify when the option will be terminated, and the option policy corresponds to the sub-policy of that subtask. An initiation set and terminal conditions define the scope of a subtask. The defined options are added to the action list of policies, and the decision maker can choose to apply these temporally-extended actions instead of trying a sequence of one-step actions. This approach define a general framework for defining a subtask, and the component of a subtask is simple. Due to such modulation, the option framework is widely used representation discovery algorithms.

The framework of hierarchies of abstract machines (HAM) Andre & Russell (2000); Parr & Russell (1997) is another classic approach for hierarchically structuring a task. The framework composes policies as hierarchies of stochastic finite-state machines. The root layer decision maker decides what subtask to be performed, and each subtask is one or a collection of predefined control programs. The control programs can be a simple task like moving an arm to a specific position, and the policy learning addresses how to use these abstract actions to perform the task and ignores low-level controls. The control programs are easy to be implemented, and the combination reduces the complexity of policy learning. One major difference between HAMs and option frameworks is that HAMs restrict the available choices of actions, while option framework augments the action set. Therefore, HAMs construct a more compact representation for policies. On the other hand, HAMs require more domain knowledge to define the precise architecture, and applying this framework is more challenging for subtask identification algorithms.

There are two kinds of design to learn the policy with subtasks. One is recursive optimality, the other is hierarchical optimality. In recursive optimality, each subtask performs its task without considering the context of its parent. On the other hand, the hierarchical optimality recognizes the fact that many tasks have to consider the hierarchy relation. For example, for driving to a specific point, whether you want to stop at that point or will keep driving to the other destination influences the driving behavior; you will decelerate before reaching the point in the former case, but will keep driving in an efficient speed in the latter case. Thus, the recursive optimality is a local optimal solution compared to hierarchical optimality.

The reason to seek recursive optimality is that this kind of design removes the dependency of a subtask to its parent. Without considering the context in which a subtask is executed, it is much easier to share and re-use subtasks. The design provides a more compact representation for a task, and the idea is proposed by Dietterich (2000). The algorithm, MAXQ, separates the utility values of performing the subtask and the utility within current task after the subtask is terminated (completion value): $Q_i(s, a) = V_a(s)+C_i(s, a)$ where $V_i(s)$ is the expected cumulative rewards for executing action a on state s, and $C_i(s, a)$ is the expected cumulative rewards before subtask i ends. With this form of value function decomposition and ignoring the long term expected utility after the current task, the hierarchical utility function can be computed by a compact recursive function call. Each subtask is an independent process–its execution does not need to consider the exterior variables, or the global states. Therefore, a subtask can be implemented as a function call. The concise and compact features of MAXQ lead to subsequent works Andre & Russell (2002); Marthi et al. (2006) that adopt the design of value function decomposition and extend it to achieve hierarchical optimality.

In sequential decision making, a common approach is to model the entire task with Markov decision processes (MDPs). An MPD $< S, A, P^a_{ss'}, R^a_{ss'} >$ is composed of a set of states S, a set of actions A, a transition function specifying the transition probability from s to s' with action a, and a reward function specifying the reward from s to s' with action a. In MDPs, the action execution is represented by only the sequence of these actions, and the performance time of each action is ignored. In hierarchical reinforcement learning, each subtask may take various amount of time, and ignoring the time factor becomes sub-optimal when making decisions. The semi-Markov decision process (SMDP) is a framework that extend MDPs to consider temporal-effect. Each action has an additional time variable, and the utility of an action is its expected utility over time. The Bellman equation for calculating utility values becomes

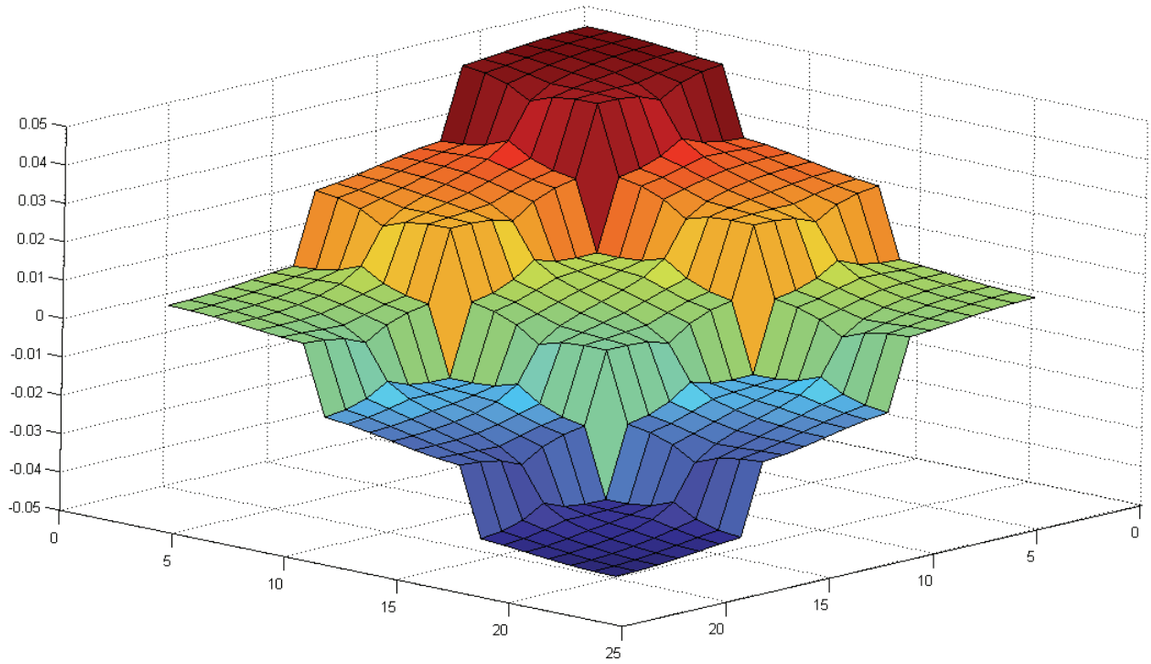


Fig. 1. The second eigenvector of the graph Laplacian on a 9-room example. Spectral graph theory has been widely applied in subgoal identification algorithms and value function approximation like Proto-value functions Mahadevan & Maggioni (2007).

$$V(s) = \max_{a \in A_s} \left[R(s,a) + \sum_{s',\tau} \gamma^{\tau} P(s', \tau | s, a) V(s') \right]$$

The policy function evaluates the utility of an action with the consideration of its various executing time, and fits the execution criterion of the subtask. Thus, this is the main model applied in hierarchical reinforcement learning.

3. Subtask identification

The fundamental works about hierarchical representation relies on manual definition for sub-tasks. This kind of design requires sufficient prior knowledge about a task, and its optimality depends on the proper construction of the hierarchical structure. Although hierarchical reinforcement learning framework provides an efficient formulation on a complex problem, the manual design requirement limits its flexibility. In many cases, we could only acquire partial solutions in problem solving that do not provide complete information for deciding a policy for states that the problem solver had not visited. These action sequences contain information for dimension reduction for the hierarchical reinforcement learning framework. The design of a HRL algorithm includes two processes, hierarchical control construction and abstractions. To formulate HRL automatically is to provide methods for performing these two processes with associated learning algorithms. This leads to researches on subtask identification and dimension reduction.

Subtask identification processes are interpreted as subgoal identification processes. Once subgoals are identified, subtasks are formulated to pursue these subgoals, and these subgoal

states are the terminal conditions, and subtask policies aim for reaching the subgoals. The existing algorithms for subgoal identification can be classified into three types: (1) Identifying subgoals as states that are most relevant to a task. (2) Identifying subgoals as states that provide an easy access to the neighbor regions. (3) Constructing subtasks based on factored state space. The first case identifies subgoals as states with a high visit frequency and reward gradient Digney (1998) or as highly-visited states based on only successful trajectories McGovern & Barto (2001). Şimşek & Barto (2004) uses relative novelty as a metric to classify subgoals and non-subgoal states.

The second case defined decomposition states as access states or bottlenecks that is similar to the graph cut in graph theory. Thus, to identify bottlenecks, Menache et al. (2002) applied graph cut based on conventional network flow analysis to the whole state transition graph. Chiu & Soo (2007) use a flooding algorithm to transmit network flow between starting position and terminal position of problem solver, and take states with local maximum flood value as subgoals. Şimşek & Barto (2008) took similar idea that their approach calculates the density of shortest paths through state nodes, which is called *betweenness* in their work, and choose states with local maximum density as subgoals.

Şimşek et al. (2005) applied normalized cut based on spectral clustering approach Shi & Malik (2000) to the state transition graph updated through newly observations. The graph cut itself only provides binary separation, so they took part of state space for analysis to reduce computation complexity, and may find different results on different trials. Chiu & Soo (2010) also adopt spectral graph theory, but instead of using graph cut result, they take the smoothness property of the spectral theory. With the spectral analysis, the edges with local maximum differences are considered bottleneck edges, and their connecting nodes are bottleneck states. Mannor et al. (2004) proposed a clustering method to identify blocks which are densely connected inside but weakly connected in between. The algorithm also finds multiple separations.

HEXQ Hengst (2002) evaluates the updating frequencies of variables to rank their hierarchy. The idea is that variables which change often tend to be at lower level of hierarchy, just as variables in the inner loop will change more often. The framework constructs options based on the projected graph. The heuristic considers one variable at a time, and can not model causal relations with more than one variable. Thus, Jonsson & Barto (2006) and Mehta et al. (2008) took dynamic Bayesian network to model the causal relation. Their work requires a pre-constructed dynamic Bayesian network.

Most of works for subgoal identifications are based on discrete domains. Konidaris & Barto (2009) proposed skill chaining to find options in continuous domain. The idea is similar to the LQR-tree mechanism which builds a tree gradually via sampling points in the state space, and finds a trajectory to link to the tree. Skill chaining takes other options' boundary as terminal states and creates a new option to link to the existing option sets.

4. Redundancy reduction

Identifying subgoals from a problem only completes the half job. The benefit of constructing hierarchical reinforcement learning is its potential for redundancy reduction, and it is easier for a programmer to implement the subtask policies. Redundancy reduction scaled down the complexity of a problem in order to help the learning process, which can be done by eliminating irrelevant parameters for decision making. Givan et al. (2003) proposed a notion of equivalence in which states can be merged without losing optimality. The states are

aggregated together if they have the same rewards and state transitions. The approach is applied on factored representation, and is guaranteed to be optimal. In the work of constructing basis functions for hierarchical reinforcement learning Osentoski & Mahadevan (2010), they used a graph reduction method to merge nodes connected to the same set of vertices, having the same edge labels, and a subset of their variables are the same.

Jong & Stone (2005) proposed a statistical hypothesis-testing approach to evaluate the relevance of state parameters. The method took the p-value of a single state to represent the overall projection result on a projected state. The algorithm defines the irrelevance of a state parameter if there is an action that is optimal among states that differ within only the parameter. Then all states projecting onto that state could share a unique optimal decision making without the state parameter, and the state parameters of this projection is apparently irrelevant. The irrelevant parameter analysis requires an optimal value function to determine the relevance of state parameters for decision making. Thus, the analysis can not help current problem, but the derived knowledge can be applied on similar problems to remove irrelevant parameters. The approach is applied on general MDPs, but it does not guarantee the optimality of abstractions.

Chiu & Soo (2010) proposed to use analysis of variance to derive irrelevant parameters from incomplete data, namely, a partial near-optimal solution. They defined irrelevant state parameters as parameters that do not affect the policy function. In other words, the value distribution of a policy function should be at least similar if not exactly identical among states that differ only on that state parameter. The similarity of distributions is compared via the variance analysis. The analysis estimates the policy function value distribution of a state and its projected states, and takes the mean and variance from the value distribution of the projected states onto that state. It estimates the approximate distribution before the value function is exactly calculated.

5. Conclusions

We survey some recent works about subgoal identification and redundancy reduction. Conventional reinforcement learning algorithms runs in polynomial time, but for most of the problems, this cost is not practical. Hierarchical reinforcement learning is one of the approaches to make this framework infeasible. The abstraction mechanism in hierarchical reinforcement learning not only plays the role of dimension reduction. Via assigning low level works to some simple programs, the new framework maps the problem into a architecture that closer to the style that human used to adopt for problem solving. Hierarchical reinforcement learning brings another thought of design for the programmer to define the problem, which makes problem solving much easier. Hierarchical reinforcement learning depends on temporal-extended actions, and there are many existing works proposed for constructing these actions. These works improve the performance of reinforcement learning to some extent, but the gains are not significant enough to scale down most of the complex problems. A good way to construct a compact reinforcement learning is an open problem, and is an important issue required further focus in the field of reinforcement learning.

6. References

Andre, D. & Russell, S. J. (2000). Programmable reinforcement learning agents, *in* T. K. Leen, T. G. Dietterich & V. Tresp (eds), *NIPS*, MIT Press, pp. 1019–1025.

Andre, D. & Russell, S. J. (2002). State abstraction for programmable reinforcement learning agents, *AAAI/IAAI*, pp. 119–125.

Chiu, C.-C. & Soo, V.-W. (2007). Subgoal identification for reinforcement learning and planning in multiagent problem solving, *MATES '07: Proceedings of the 5th German conference on Multiagent System Technologies*, Springer-Verlag, Berlin, Heidelberg, pp. 37–48.

Chiu, C.-C. & Soo, V.-W. (2010). Automatic complexity reduction in reinforcement learning, *Computational Intelligence* 26(1): 1–25.

Chiu, C.-C. and Soo, V.-W. (2010), AUTOMATIC COMPLEXITY REDUCTION IN REINFORCEMENT LEARNING. Computational Intelligence, 26: 1–25.

Şimşek, O. & Barto, A. G. (2004). Using relative novelty to identify useful temporal abstractions in reinforcement learning, *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, ACM, New York, NY, USA, p. 95.

Şimşek, O. & Barto, A. G. (2008). Skill characterization based on betweenness, *in* D. Koller, D. Schuurmans, Y. Bengio & L. Bottou (eds), *NIPS*, MIT Press, pp. 1497–1504.

Şimşek, O., Wolfe, A. P. & Barto, A. G. (2005). Identifying useful subgoals in reinforcement learning by local graph partitioning, *ICML '05: Proceedings of the 22nd international conference on Machine learning*, ACM, New York, NY, USA, pp. 816–823.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the maxq value function decomposition, *J. Artif. Intell. Res. (JAIR)* 13: 227–303.

Digney, B. (1998). Learning hierarchical control structure for multiple tasks and changing environments, *Proceedings of the Fifth Conference on the Simulation of Adaptive Behavior: SAB 98*.

Givan, R., Dean, T. & Greig, M. (2003). Equivalence notions and model minimization in markov decision processes, *Artif. Intell.* 147(1-2): 163–223.

Hengst, B. (2002). Discovering hierarchy in reinforcement learning with hexq, *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 243–250.

Jong, N. K. & Stone, P. (2005). State abstraction discovery from irrelevant state variables, *IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 752–757.

Jonsson, A. & Barto, A. (2006). Causal graph based decomposition of factored mdps, *J. Mach. Learn. Res.* 7: 2259–2301.

Konidaris, G. & Barto, A. (2009). Skill discovery in continuous reinforcement learning domains using skill chaining, *in* Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams & A. Culotta (eds), *Advances in Neural Information Processing Systems 22*, pp. 1015– 1023.

Mahadevan, S. & Maggioni, M. (2007). Proto-value functions: A laplacian framework for learning representation and control in markov decision processes, *Journal of Machine Learning Research* 8: 2169–2231.

Mannor, S., Menache, I., Hoze, A. & Klein, U. (2004). Dynamic abstraction in reinforcement learning via clustering, *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, ACM, New York, NY, USA, p. 71.

Marthi, B., Russell, S. J. & Andre, D. (2006). A compact, hierarchical q-function decomposition, *UAI*, AUAI Press.

McGovern, A. & Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density, *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 361–368.

Mehta, N., Ray, S., Tadepalli, P. & Dietterich, T. (2008). Automatic discovery and transfer of maxq hierarchies, *ICML '08: Proceedings of the 25th international conference on Machine learning*, ACM, New York, NY, USA, pp. 648–655.

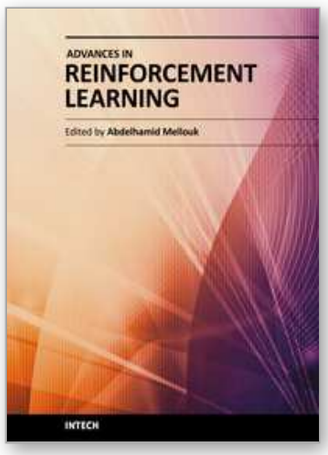
Menache, I., Mannor, S. & Shimkin, N. (2002). Q-cut - dynamic discovery of sub-goals in reinforcement learning, *ECML '02: Proceedings of the 13th European Conference on Machine Learning*, Springer-Verlag, London, UK, pp. 295–306.

Osentoski, S. & Mahadevan, S. (2010). Basis function construction for hierarchical reinforcement learning, *AAMAS '10: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 747–754.

Parr, R. & Russell, S. J. (1997). Reinforcement learning with hierarchies of machines, *in* M. I. Jordan, M. J. Kearns & S. A. Solla (eds), *NIPS*, The MIT Press.

Shi, J. & Malik, J. (2000). Normalized cuts and image segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.* 22(8): 888–905.

Sutton, R. S., Precup, D. & Singh, S. P. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning, *Artif. Intell.* 112(1-2): 181–211.



Advances in Reinforcement Learning
Edited by Prof. Abdelhamid Mellouk

ISBN 978-953-307-369-9
Hard cover, 470 pages
Publisher InTech
Published online 14, January, 2011
Published in print edition January, 2011

Reinforcement Learning (RL) is a very dynamic area in terms of theory and application. This book brings together many different aspects of the current research on several fields associated to RL which has been growing rapidly, producing a wide variety of learning algorithms for different applications. Based on 24 Chapters, it covers a very broad variety of topics in RL and their application in autonomous systems. A set of chapters in this book provide a general overview of RL while other chapters focus mostly on the applications of RL paradigms: Game Theory, Multi-Agent Theory, Robotic, Networking Technologies, Vehicular Navigation, Medicine and Industrial Logistic.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen