Hierarchical Strategy Learning with Hybrid Representations

Sungwook Yoon
Computer Science & Engineering
Arizona State University
Tempe, AZ 85281
Sungwook.Yoon@asu.edu

Subbarao Kambhampati
Computer Science & Engineering
Arizona State University
Tempe, AZ 85281
rao@asu.edu

Abstract

Good problem solving knowledge for real life domains is hard to define in a single representation. In some situations, a direct policy is a better choice while in others, value function is better. Typically, direct policy representation is better suited to strategic level plans, while value function representation is better suited to tactical level plans. We propose a hybrid hierarchical representation machine (HHRM) where direct policy representation and value function based representation can co-exist in a level-wise fashion. We provide simple learning and planning algorithms with our new representation and discuss their application to Airspace Deconfliction domain. In our experiments, we provided our system `LSP` with two level HHRM for the domain. `LSP` could successfully learn from limited number of experts' solution traces and show superior performance compared to average of human novice learners.

Introduction

Many real life problems can be most compactly described with hierarchical representation (Erol, Hendler, & Nau 1996; Dietterich 2000; Parr & Russell 1997). Often, compact strategies for each level in the hierarchy can be represented in a different form. Consider riding a bike to some local grocery. At the top level, the solution strategy can be symbolically represented as "bike to the crossroad" and "turn left". At the bottom level, the solution must describe all the physical movement that can be applied to the environment. Bottom level actions look like, "tilt the wheel 5 degrees to the right, set the pedal RPM at 60 for 2 seconds, tilt the wheel 2.5 degrees to the left etc.,". Such actions can be more compactly represented as "maximize the stability of the bike, en-route to the crossroad". The latter solution strategy takes the form of value function, and the former top level solution strategy has direct policy representation.

A compact solution strategy for *Airspace Deconfliction* problem can also be represented as "hybrid" representations in the hierarchy. Airspace deconfliction problem is placing airspaces in 4-D space, or temporal and spatial space. An airspace is defined by its occupancy in 4-D space, where it happens in latitude, longitude and altitude space and when it happens in the time space. Before an operation of any task that needs some airspaces, the person in charge of the operation should ask *Airspace Authority* and get the permission. Airspace requests can overload some portion of the 4-D space and Airspace Authority needs to do deconfliction operation. Airspace Authority deconflicts the airspace by changing the requested airspaces in 4-D space. The changes should minimally affect the original objective of the airspace requests and accommodate as many airspace requests as possible. Airspace deconfliction domain will be our ongoing example as well as target experiments domain.

Experts on Airspace Deconfliction problem use "hybrid" strategy during deconfliction. First, experts choose the top level solution strategy by looking at the usage of the airspaces. For example, if the use of the airspace is "fighter", they choose to raise the altitude of the airspace. If the use of the airspace is "missile", experts choose to delay the airspace. Second, the detailed altitude or time is decided by considering the minimal movement that guarantees the deconfliction. The latter strategy can be most compactly represented as a form of cost function, that the experts minimize. Thus, the best solution strategy can be represented and learned with hybrid hierarchy representation.

We propose to study hybrid hierarchical representation machine (HHRM). In our study, we assume that the hierarchy structure of the target domain is given to our algorithm. We also assume that sample solution trajectories are given to our system. We then attempt to extract hybrid hierarchical strategy (HHS) from the training trajectories. The learned HHS is then tested against unseen problems. Ideally, we would expect that the sample solutions are derived from representative problems of the target domain. In what follows, we describe our representation scheme and learning algorithms.

Representation and Learning Algorithm

We represent actions in a hierarchy. Actions other than the most bottom level actions are abstract actions. Only the bottom level actions are physically realizable actions. We consider simple hybrid hierarchical action representation scheme. In our representation, direct policy or value function based representations are mixed. Each level is represented in a single representation. That is, if an action in the top level is represented in a direct policy, all the actions in the top level are represented in direct policy representation.

At any level of the hierarchy, an action is represented in one of the following form,

$$\mathbb{A} = \begin{cases} \mathbb{R}(x_1, \ldots, x_n) \\ \min_{A'} \mathbb{F}_{\mathbb{A}'}(x_1, \ldots, x_n) \end{cases} \quad (1)$$

First, there is a relational abstract action or symbolic action. This is ideally suited for direct policy representation. For example, `raise-altitude(Fighter)` and `set-minimum-altitude(Fighter 34000ft)` are symbolic actions for airspace deconfliction domain. The latter action corresponds to bottom level action and can be physically realized. Second, there is a minimization (or maximization) action. This action describes which functions are to be minimized with a predefined set of actions in the lower level. For example, $\min \mathbb{F}$ = `cost of higher altitude flying` is such an action. Let the $\min \mathbb{F}$ be at level l and $\min \mathbb{F} \in A_l$. The actions A_{l+1} in the right below the level of the current level l that minimize the value of $\mathbb{F}$ are sought for $\min \mathbb{F}$.

Next, we define HHRM by giving the description of the relation between two consecutive levels of actions and the top level actions. This is similar to typical inductive definition.

$$A_l : \{a \in A_{l+1}\} \quad (2)$$

Equation 2 shows how we define two consecutive levels of actions. The definition provides the set of lower level actions A_{l+1} that belong to the current level action A_l.

$$\begin{array}{l} \texttt{raise-altitude}(x) : \\ \{\min_{x,a}(\texttt{altitude-change-cost}(x,a) + \\ \texttt{altitude-conflict-cost}(x,a))\} \end{array} \quad (3)$$

For example, Equation 3 shows a hierarchy definition for Airspace Deconfliction Domain. This hierarchy defines the relation between a top level action `raise-altitude(x)` with the lower level action, which minimizes the cost of changing altitude and conflict.

The top level actions are defined by the set of action templates for the top level.

$$\begin{cases} \texttt{raise-altitude(x)} \\ \texttt{lower-altitude(x)} \\ \texttt{move(x)} \\ \texttt{delay(x)} \\ \texttt{advance(x)} \\ \texttt{add-point(x)} \\ \texttt{delete-point(x)} \\ \texttt{shrink-radius(x)} \end{cases} \quad (4)$$

Equation 4 shows the set of top level action definition for Airspace Deconfliction Domain.

One interesting aspect of our action hierarchy is that the representations can be nested each other. Symbolic action can occur inside the minimization action, or vice versa. For example, in the Airspace Deconfliction problem, the top level action can be minimizing global cost function $\min \mathbb{G}$. $\min \mathbb{G}$: { `raise-altitude(x), lower-altitude(x), move(x), delay(x), advance(x), add-point(x), delete-point(x)` } defines the relation between the top level action and the level below. In this representation example, the top level action is in value function representation and the level below is in direct policy representation.

Learn-Abstract-Actions $(\mathbb{J}, H)$
// experts' trajectories $\mathbb{J}$, Action Hierarchy Definition H
$B \leftarrow \{\}$ // initialize variable binding
FOR each level l in H
  IF l is SYMBOLICABSTRACTACTION
    Learn-Action-Selection-Rules ($\lrcorner$, H, l,B)
  IF l is COSTMINIMIZATION
    Learn-Cost-Minimization-Weight($\lrcorner$, H, l, B)
  update-binding(B)

Figure 1: Pseudo-code for learning hybrid hierarchy actions

Available actions in the lower hierarchy is limited by the actions in the hierarchy. The variable binding from the upper level hierarchy limits some of the variable binding of the lower level actions. In the above example of Equation 3, when action `raise-altitude(Fighter)` is selected, the variable x in the lower level actions are bound to Fighter. The idea of limiting the scope of the actions in the lower hierarchy is similar to previous studies in (Andre & Russell 2000; Erol, Hendler, & Nau 1996)

Unlike HTN studies, our hierarchy representation does not have the "precondition" for each task in the hierarchy. Due to this fact, it is hard for planners to directly use our representation as solution strategy. Next, we describe our learning algorithm for HHRM.

Learning Algorithm

In this work, we do not aim to solve problems directly with the definition of the hierarchy, rather we rely on expert's guidance to extract useful strategy. In the experts' guidance, we have the tags for hierarchy information. For example, experts' action sequence consists of ordered bottom level actions, e.g., `set-minimum-altitude(Fighter 34000ft)`.

Our learner has the access to the hierarchy information for the sequence, "raise-altitude(Fighter):" "minimize the cost of altitude chage:" "set-minimum-altitude(Fighter 34000ft)", "set-maximum-altitude(Fighter 38000ft)". Currently, the human tags the abstract action information. To remove the human tagging help, we are planning to use Viterbi algorithm or more advanced plan recognition algorithms for finding the unseen hierarchy information.

Figure 1 shows our algorithm for learning. The algorithm iterates through action hierarchy to learn abstract actions. Learn-Action-Selection-Rules is used for learning symbolic abstract actions. In our initial implementation, we have used relational rule learning program based on PRISM algorithm (Cendrowska 1987). Learn-Cost-Minimization-Weight is used for learning cost-minimization abstract actions. We implicitly assume that the cost function is represented as a weighted combination of input features. More specifically, we assume that the combination of the fea-

tures is linear. Note that only at the top level actions are learned with global scope, while the lower level actions are learned with local scope, limited by variable bindings of the higher level actions. For example, to learn cost-minimization action for $\mathbb{F}$ = ``cost of higher altitude flying'' action, the cost is limited to cost functions that involves ``Fighter'', if the top level action is tagged with raise-altitude(Fighter).

Airspace Deconfliction Experiments

We have tested our ideas on Airspace Deconfliction domain and have obtained positive results. In our experiments, our learning system called LSP observes the experts' actions for the training deconfliction problem. Then LSP performs against unseen problem for testing purpose.

For this domain, we defined a three level hierarchy. The top level consists of symbolic actions that move the airspaces in horizontal, vertical or temporal directions, as shown in previous section. The middle level actions are cost minimization functions, that minimize the predefined cost functions for each of the top level actions. The bottom level actions are physically realizable actions that set constant values to attributes of the airspaces. In our testing, we focused on learning the top level actions. For the middle level cost minimization functions, we used "minimal displacement" function, which will be described in detail in the following subsections. We expect that the weight for the cost functions can be easily learned with perceptron variants of algorithms.

In the following, we describe detailed experimental environment. We start with how we generate relational features. Then we give the skewed data problem and the approach we took to resolve the problem. We conclude with the learned abstract rules and performance analysis on the target domain.

Feature Generation

We describe the target problem with relational representation, a set of true ground facts of first order logic. For example, a relation set { (use ACM-B-1 smss), (conflict ACM-B-1 ACM-A-1), ... } describes some part of the Airspace Deconfliction problem. The description expresses that the use of airspace "ACM-B-1" is "smss" and it is in conflict with "ACM-A-1".

We generate binary features for symbolic actions in the target domain. We describe the properties of arguments of the actions. Each description of the argument of an action type is a binary feature. In what follows, we formally define our feature set. It will be obvious and straightforward that the feature set can be automatically enumerated.

We used Taxonomic syntax (McAllester & Givan 1993) for the description of properties of arguments of symbolic actions. Variable free taxonomic syntax is a convenient tool for relational representation as has been studied (Yoon, Fern, & Givan 2002). The syntax describes objects with some property. Taxonomic Expression is defined as follows.

$$C = (R, i, C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_{n(R)}) \qquad (5)$$

R is the relational symbol. For example, "use" in (use ACM-B-1 smss) is an R. $n(R)$ is the arity function for the relational symbol R. i is the set of objects that the expression C in Equation 5 describes. We interpret the equation as follows, the set of objects that satisfy the predicate R at the position of i, when all the other argument positions k are in C_k. In other words, the equation describes the set of objects, that make the relation R true at position i, once other positions are in corresponding class expressions. For example, a taxonomic expression $C =$ (use 0 smss), is ACM-B-1 in our on-going example, since ACM-B-1 makes (use $*$ smss) true when it is in the 0th position. For a more detailed description of taxonomic syntax, please refer (Yoon, Fern, & Givan 2002). Note that a limited set of taxonomic expressions can easily be enumerated from the domain definition.

We can now describe the arguments of the actions with taxonomic representation.

$$F = (A \ \ i \ \ C) \qquad (6)$$

We define binary features as Equation 6. The feature F describes that the action type is A and the argument i is in C. Thus, the feature is a binary feature. The feature is true only for the actions with the action type A and the ith argument is in class C.

For example, (delay 0 (use 0 smss)) is a feature in Airspace Deconfliction domain. The feature is true for actions with type "delay" and the 0th argument is in class expression (use 0 smss), which designates airspaces of usage "smss". Thus the feature means that "delay when the use of the airspace is smss". As taxonomic expressions, the binary features can be automatically enumerated for the given set of taxonomic expressions. The whole feature set can thus be automatically enumerated from domain definition.

Skewed Data

Typically as with fraud detection (Phua, Alahakoon, & Lee 2004), learning from trajectories suffers from skewed data. For any situation, learning system can only observe one action in the trajectory. The rest of the potentially selectable actions are treated as negative examples. Thus, there are many negative examples per positive example, and the training data is unbalanced or skewed. One can use a sampling based technique to cope with the unbalanced data (Chawla *et al.* 2002) or one can use cost based approach (Maloof 2003). We took the latter approach. We gave higher weight to the features that classify positive example correctly. This has been successful for several research studies (Maloof 2003; Yoon, Fern, & Givan 2005), and was also effective for our experiment on Airspace Deconfliction domain. Self simulation can also boost the learning performance as studied in (Fern, Yoon, & Givan 2003)

The other problem in the mimicry study is the "good actions" in the considered "negative examples". Though the expert selected an action A in situation S, and we treat other potential actions as "bad" actions, there can be some good actions among the bad actions. They are just not taken by the expert. This incurs the overfiting problem. (Yoon, Fern, & Givan 2005) addressed the issue with a novel representation scheme. In our current study, we allow minimal number of

negative example coverage. Our approach relieves the overfiting problem and our experiments show that our approach works well.

Learned Rules

Following lists some of the top level symbolic actions learned with our algorithm. We hand translated the features to English for readers' convenience.

- "Raise the altitude" if the use of airspace is ADOA
- "Advance the time" if the airspace conflicts with the use of "ACSS"
- "Delay the time" if the airspace conflicts with the use of "ACSS"
- "Delay the time" if the use of the airspace is SMSS
- "Move the airspace horizontally" if the use of the airspace is CAP

From our observation of the domain expert, the learned rules seem to make sense for the target domain. The subjective scoring in the following section corroborates this.

Performance Analysis

We have tested our learning algorithm in the Airspace Deconfliction scenarios. Our implemented module is a part of a Meta learning system called GILA (general integrated learning architecture). In the architecture, there are three learning components and one meta executive, that selects solutions among the proposed ones by each component. Each of the learning component is called ILR (integrated learner and reasoner). Our module, called `LSP`, was one of the ILRs. In this testing, we did not learn value function. In place of the value function, we fixed one value function for each abstract (symbolic) action. We used "minimum displacement" cost function. This function minimizes the displacement of the original value for each abstract action while removing the conflict. For example, when "raise altitude" abstract action is applied, the corresponding "minimum displacement" function selects altitude that removes the current conflict and the change is minimal. There is an obvious pitfall to this approach, since in some cases maximal change could be required. In our future study, we will explore the learning of the cost minimization function and will overcome this obvious pitfall.

For each learning and testing scenario or episode, single trajectory of an expert is given to the system. Then each ILR learns from the trajectory with its own learning algorithms. After learning is finished, a testing case is given to the whole system. Each ILR finds solutions and proposes them. MRE then selects the best solution from its own selection criteria, mostly favoring the ones that reduce the conflicts most. Figure 2 shows testing results. Note that the the deconfliction problems are very complex and hard even for humans.

In Figure 2, each column of C, D, and LSP shows performance of each ILR. A scenario consists of a pair of learning trace and testing problem. The values in C, D, and LSP column shows how much portion of the solutions have come from the corresponding ILR and the number is the percentage of the solution proposed by the ILR and used in the final solution. As can be seen in the figure, LSP's solution has been selected most frequently. This shows that our approach of HHRM is well suited for the learning problem. What is more interesting is the fact that our learning system generated features automatically without human help. QoS (quality of solution) columns show the subjective score of each solution. The score has been made available by a human referee. The score of GILA is not perfect. However the fact that average of human novices got less score than our system underlines very promising initial experimental results. One interesting phenomenon here is the correlation of the LSP's solution participation and the QoS. The quality of the solution depends on which kind of action has been used in which situation. That is what our abstract level action learning system is targeted for. Thus, not surprisingly but very encouragingly, as `LSP` solution participates more the quality of the solution proportionally increases.

For now, Gila scores above 50% of experts' performance. This performance has been achieved with learning from single trajectory. We expect that the performance would be raised as we apply learning algorithms to the cost minimization function learning. Other potential improvements include self-simulation and additional background knowledge. The latter approach would help our system `LSP` to generate complex features which is not easy for blind enumeration.

Related Works

There have been some studies that attempted to incorporate "hybrid" representation for the planning problems. For example, adding continuous variables on top of typical symbolic variables to planning problems (Younes, Musliner, & Simmons 2003; Kveton, Hauskrecht, & Guestrin 2006) have been studied. These efforts limited the continuous variables to a small number of numeric resources or the size of the domains was also kept very small. It is hard to apply these techniques directly to any real life problems including deconfliction. One weakness of these approaches is the ignorance of the apparent hierarchical structure in many of the application domains.

Many hierarchical representation studies (Nau *et al.* 1999; Dietterich 2000; Precup & Sutton 1998) stayed within single representation scheme, direct policy representation or value function representation. These studies did not investigate the potential of the hybrid representations in their hierarchy of strategies.

Our study is somewhat similar to the learning of precondition for HTN (Ilghami, Nau, & oz Avila 2002), though we don't specify explicit preconditions for our hierarchy representation. Once strategies are learned, our system can perform against any unseen problems. This is similar to many mimicry studies in AI (Morales & Sammut 2004; Abbeel & Ng 2004), but our learning algorithm deals with much bigger scale domains with novel HHRM.

Conclusion and Future Works

We have proposed a novel hybrid hierarchical representation machine (HHRM). We tested the idea on Airspace Deconfliction domain and our experimental results show that our

Scenario	C	D	LSP	QoS (Average)	QoS (Median)
1	8	39	**47**	61.4	70
2	19	**49**	27	40	10
3	17	26	**52**	55.6	70
4	0	23	**77**	76.7	80
5	6	43	**51**	65	70
6	0	32	**63**	83.0	90
7	15	**44**	35	58.6	80

Figure 2: HHRM performance chart: Tested on 7 scenarios. Each Scenario has one learning trace and one performance trace. The numbers in other cells represent performance measure against each performance trace in each row, learned with learning trace in the corresponding row. C,D, LSP are independent learners and our HHRM is LSP. C,D,LSP are independent but an executive module called MRE selects solutions during solving the Airspace Deconfliction problems. The numbers show what percentage of solutions each component contributed to the final solution. As can be seen, LSP is selected most frequently. QoS (Quality of Solution) is scored by a human referee. Here again, as LSP contributes more the Quality of Solution increased in both Average and Median. This shows that our proposed learning and representation works well on a real life scale domain.

representation scheme can work well on real life sized problems.

One critical assumption in our study is regarding the knowledge of the hierarchy and tagging of the higher level actions in the example trajectory. Our immediate extension of the current work is developing algorithms that can get by with the tagged information or hierarchical structural information. We expect that we can lift the need of tagging information, with Viterbi-style algorithms that find the unseen tags with maximum likelihood.

Acknowledgement

This research has been supported in part by Integrated Learning Program of DARPA. We thank Phil Dibona for providing us with experimental results.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*.

Andre, D., and Russell, S. 2000. Programmable reinforcement learning agents.

Cendrowska, J. 1987. Prism: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies* 27(4):349–370.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. Smote: Synthetic minority oversampling technique.

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

Erol, K.; Hendler, J. A.; and Nau, D. S. 1996. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence* 18(1):69–93.

Fern, A.; Yoon, S.; and Givan, R. 2003. Approximate policy iteration with a policy language bias. In *Proceedings of the 16th Conference on Advances in Neural Information Processing*.

Ilghami, O.; Nau, D. S.; and oz Avila, H. M. 2002. Camel: Learning method preconditions for htn planning. In *AIPS-02*.

Kveton, B.; Hauskrecht, M.; and Guestrin, C. 2006. Solving factored MDPs with hybrid state and action variables. *Journal of Artificial Intelligence Research* 27:153–201.

Maloof, M. 2003. Learning when data sets are imbalanced and when costs are unequal and unknown.

McAllester, D., and Givan, R. 1993. Taxonomic syntax for first-order inference. *Journal of the ACM* 40:246–283.

Morales, E., and Sammut, C. 2004. Learning to fly by combining reinforcement learning with behavioural cloning. In *ICML*.

Nau, D.; Cao, Y.; Lotem, A.; and Munoz-Avila, H. 1999. Shop: Simple hierarchical ordered planner. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 968–973.

Parr, R., and Russell, S. 1997. Reinforcement learning with hierarchies of machines. In Jordan, M. I.; Kearns, M. J.; and Solla, S. A., eds., *Advances in Neural Information Processing Systems*, volume 10. The MIT Press.

Phua, C.; Alahakoon, D.; and Lee, V. 2004. Minority report in fraud detection: classification of skewed data. *SIGKDD Explor. Newsl.* 6(1):50–59.

Precup, D., and Sutton, R. S. 1998. Multi-time models for temporally abstract planning. In Jordan, M. I.; Kearns, M. J.; and Solla, S. A., eds., *Advances in Neural Information Processing Systems*, volume 10. The MIT Press.

Yoon, S.; Fern, A.; and Givan, R. 2002. Inductive policy selection for first-order MDPs. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*.

Yoon, S.; Fern, A.; and Givan, R. 2005. Learning measures of progress for planning domains. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*.

Younes, H.; Musliner, D.; and Simmons, R. 2003. A framework for planning in continuous-time stochastic domains.