Unsupervised Learning of HTNs in Complex Adversarial Domains

Michael Leece
Computational Cinematics Studio
UC Santa Cruz
mleece [at] ucsc.edu

Abstract

While Hierarchical Task Networks are frequently cited as flexible and powerful planning models, they are often ignored due to the intensive labor cost for experts/programmers, due to the need to create and refine the model by hand. While recent work has begun to address this issue by working towards learning aspects of an HTN model from demonstration, or even the whole framework, the focus so far has been on simple toy domains, which lack many of the challenges faced in the real world such as imperfect information and continuous environments. I plan to extend this work using the domain of real-time strategy (RTS) games, which have gained recent popularity as a challenging and complex domain for AI research.

Introduction

Using games to push the limits of Artificial Intelligence has a long history. From computer chess and Kasparov vs. Deep Blue in the 80's and 90's, to the uncertain and partner-dependent environment of bridge in the later 90's, to the still-challenging complexity of Go, these domains have served well as controlled environments where AI techniques and models can be developed and refined for practical use in the real world. And while computer Go programs are on the brink of defeating the top humans, another domain has emerged where artificial agents are woefully far from competing with humans–real-time strategy (RTS) games.

There are a wide variety of RTS games, but the general characteristics of the genre involve a contest between players that is based around both economic and militaristic management. Players generally start with a small economy, which they must grow in order to produce an army with which to fight their opponent. Interactions between players include harassment to slow down the opponent's economy, surprise attacks to circumvent static defenses, head-on assaults, and more. As the game proceeds, players must balance investing into their economy, which will help them later down the road, and investing in their military, which will help defend them or allow them to pressure the opponent.

RTS games are challenging for artificial agents for a number of reasons. First, they are imperfect information games. Knowledge of what your opponent is doing must be actively sought, in order to make the correct trade-off decisions that will allow a player to punish the opponent. Second, they are real-time. Rather than being divided up into alternating turns, each player is making decisions and performing actions asynchronously, which greatly limits the amount of search that can be done in time-sensitive situations. Third, they are adversarial (a property more inherent to games). Many AI tasks involve simply accomplishing some goal, but in these games there is an agent actively opposing your actions, and able to react to your plans. Additionally, planning is required at multiple levels of abstraction, from high-level resource division between economy and military, to low-level individual unit control.

Fortunately, RTS games are also popular in and of themselves with human players. This provides us with a large amount of human expertise that can be drawn on. In addition, some games have reached the level where professional leagues of players exist, who train extensively to compete with each other. This gives us confidence that the expertise of these players is truly meaningful, and that the signal:noise ratio in their games is high. In addition, replays of these expert players are posted online, for other players to study and learn from, which we also can take advantage of.

Due to these attractive characteristics, research in the field of RTS AI has been growing steadily in recent years. An annual tournament has even begun, where AI developers can enter their state-of-the-art agents to be tested against others over thousands of games, with a match against a high-level human player for the winning agent. While the AIs have made incremental improvements over the years, they remain unable to defeat human players reliably.

However, the overall trend of the work so far has been towards a compartmentalized view of AI, developing individual competencies and attempting to tie them together into a complete agent. As one small example, there has been a good deal of work in the field of "micro", the term for individual unit control within fights. And in reality, when it comes to fights with units without special abilities, the AIs developed surpass human ability. However, fights are not independent of the rest of the agent. An AI committing to a "rush" strategy has different fight priorities than one that is playing for a longer game, and this needs to be reflected in the micro decisions that are made. Currently, this informa-

tion flow between components is either lacking or nonexistent.

A more attractive option would be to use a single high-level planner, obviating the need for communication protocols between components that use completely different models to perform their tasks, and therefore may not be well-suited for conveying their goals to each other. We choose to use Hierarchical Task Networks for this goal. This choice is chiefly driven by the hierarchical nature of plans in RTS games when expressed by humans, and also by the flexibility and expressivity of HTNs, important for reasoning at the multiple levels required.

HTNs are a very popular planning framework, and have been used in many real-world domains, from manufacturing to medical care (Fernández, Aler, and Borrajo 2005) (Musen 1989). However, there is one great drawback traditionally cited to them whenever they are considered, and this is the cost of expert labor involved in creating a useful HTN. Some work has been done in the learning of HTN method pre/postconditions, and even of learning the full method decomposition structure from scratch. However, these have focused on extremely simplistic domains such as the classical Blocks World or logistics transportation proglem. At the same time, work from Google Brain and on Deep Neural Networks (Lee et al. 2009) is demonstrating that with sufficient data, meaningful patterns can be learned even for increasingly complex problems. The question then becomes, can we do the same for HTN models?

Beyond simply developing an agent that can play RTS games using an HTN planning approach, the greater focus of my research is to develop a more general method for learning an HTN model from unlabeled expert demonstrations in an unsupervised manner, which can then be applied to other domains sharing similar characteristics to RTS games.

Related Work

Many of the different individual aspects of this proposal have seen work already, which gives us a number of starting points to work from.

On the more theoretical side, algorithms for learning HTN models have been developed and analyzed rigorously. In both (Garland, Ryall, and Rich 2001) and (Garland and Lesh 2003), the authors discuss the challenges and possible extensions for learning an HTN model based on annotated replays, and provide an algorithm that returns a provably sound and complete model that is consistent with the data (if possible). However, a key drawback is the annotation limitation, which requires each sequence of primitive actions to be clustered according to the higher level task that it is attempting to perform. In a similar vein, (Ilghami et al. 2002) and (Ilghami et al. 2005) present work on learning HTN method preconditions, given the task structure decomposition, and succeed in some simple domains.

More recent work has moved toward the issue of attempting to learn the entire HTN model from traces, including the task structures. HTN-MAKER, proposed in (Hogg, Munoz-Avila, and Kuter 2008), presents an approach to deducing methods based on goal annotations that indicate when a goal has been achieved, and learning methods based on state/action matchings that move from a given state into the desired state. (Zhuo et al. 2009) presents HTN-learner, an extension of HTN-MAKER, which learns both an action model and preconditions based on partial state observability, by deducing the effects of actions from changes in the state. However, it still requires demonstrations to be provided as decomposition trees, divided by which higher-level goal is being pursued at any given point. (Hogg, Kuter, and Munoz-Avila 2010) integrates reinforcement learning into the HTN-MAKER framework, in order to ascertain values for the learned methods to decide which are more likely to be useful in any given situation. Using this setup, the authors manage to improve the rate at which the HTN model learns from demonstration, requiring less examples before achieving competency. Additionally, in the tech support domain, some work has been done on using HMMs to align traces for better learning (Lau et al. 2004).

However, all of these papers work in very simple domains. Blocks World is the most common, with logistics management being the second. RTS games are significantly more complex, and it is unproven that these approaches will work in an environment where expert demonstration only showcases a miniscule fraction of the available space. This is the main goal of my research, extending the HTN learning work into more meaningful domains; in particular, RTS games.

We have reason to believe this is possible, given that hand-crafted HTNs have succeeded in the past in these complex domains. Historically, Bridge Baron (Smith, Nau, and Throop 1998) used HTNs to become the world champion computer bridge program, dealing particularly well with the imperfect information scenario that arises during bidding. Additionally, before the rise of MCTS, HTNs were one of the leading approaches for dealing with the increased complexity of Go as compared to chess (Meijer and Koppelaar 2001) (Willmott et al. 1999).

Additionally, a number of systems with similar approaches have been developed for RTS games already. Weber et al. (Weber, Mateas, and Jhala 2012) (Weber 2012) have developed a goal-driven autonomy system for SC:BW itself that uses a hand-crafted HTN-like planning component to achieve its goals, although it is more reactive than an HTN planner. Ontanón et al. have created Darmok (Ontañón et al. 2010), a case-based planning system that has been trained both by human demonstration with annotations and also from pure observation of human play (Ontanón et al. 2009). Outside of the RTS genre, (Hoang, Lee-Urban, and Muñoz-Avila 2005) developed a system for playing a first-person shooter game that used an HTN for strategic and tactical decision making, with a reactive system integrated to provide actual in-fight control.

Completed Work

Sequence Mining for Common Action Patterns

Our initial work is in a similar vein as the trace alignment from (Lau et al. 2004), though we use a completely different technical approach. Namely, in order to learn from a large database of unlabeled demonstrations, one approach is to find the commonalities that exist in your data set, and

assume that these common occurrences are meaningful. In our case, we look for common sequences of actions and believe that these represent the first level of abstract tasks in an HTN model for the domain (we use StarCraft:Brood War (SC:BW), a popular RTS game).

We used the Generalized Sequential Pattern (GSP) algorithm developed by (Srikant and Agrawal 1996), which takes as input a database of sequences, and outputs a list of patterns that occur frequently within that database. In our case, each sequence consisted of a trace of actions performed by a player in a game of SC:BW, abstracted to a level where patterns could be identified (i.e. exact location information was removed). In addition, the GSP algorithm includes capabilities for defining a maximum gap allowed between elements of a pattern. That is, two elements can only be considered part of the same pattern if they occur within some fixed time of each other.

With this approach, we were able to extract low-level action patterns such as army movement and army production cycles that correspond with how human players describe their gameplay. Additionally, with a much greater gap, we were able to identify larger patterns that correspond to specific opening strategies and economic management traditions later in the game. Both of these results are encouraging for the approach, and we plan to continue on to see if these patterns can be translated into abstract tasks in an HTN model.

These results have been accepted to the AIIDE '14 workshop for AI in Adversarial Real-Time Games.

Opponent Modeling

In an adversarial imperfect information domain, it is necessary to have a model of one's opponent in order to make accurate decisions. In addition, having an accurate model is critical to learning algorithms, as the players that they learn from are working from the imperfect information, rather than the full state information. For example, if we wish to learn the precondition for an attacking task as occurring when the player believes he has more troops than the opponent, we need a model for this belief. While only tangentially related to HTN learning itself, we have also completed some work in this area, as the existing work did not align exactly with the modeling we would like.

While there is not room here for great detail, we used a factorizable Markov Random Field with a fairly straightforward construction to model the dependencies between unit counts in SC:BW. Given this construction, we learned the clique potential tables via training on expert replays downloaded from the fan website TeamLiquid [1]. When tested on unseen replays, we achieved relatively strong performance (better than prior work), but a more dedicated modeling approach with more expertise could almost certainly improve on this even more.

These results have been accepted to CIG '14, and are awaiting publication.

[1] www.teamliquid.net

Proposed Work

While this is still under advisement, there are two main approaches that I am hoping to explore for this project, which can be classified as top-down and bottom-up.

Top-down

This approach is in line with most of the prior work in HTN learning, as listed above. As such, it will mostly require extending the existing work to be able to deal with the significantly increased complexity, uncertain information, and adversarial nature of the environment. All of these have been addressed with HTNs in the past, but not from the learning standpoint.

When we say significantly increased complexity, consider this example. In the logistics environment, the final goal for a plan may be something of the form "(at _package1 _loc1)", from which it may backtrack to realize that it needed to achieve the abstract goals of getting the package in a truck, moving the truck to loc1, and unloading it, each of which may decompose into primitive actions. By contrast, the only hard-and-fast goal in a game of SC:BW is "Win Game". Backtracing the abstract goals that lead to this accomplishment will be significantly more complicated than the examples that systems like HTN-MAKER learn to solve.

The most straightforward way to solve this issue is to receive annotation from human players, providing us with goals and their associated states. While we plan to test this, the final goal is a fully-automated system, and so we are looking for ways to deduce possible goal states based on replays, possibly by looking for commonalities in states that consistently lead to victories.

Bottom-up

This approach is less well-formed, but is an extension of our current work looking for common action patterns. We plan to continue the work by attempting to build up from the first level of action sequences to a full tree of frequent action patterns, and attempt to use these as HTN methods for which we must learn the preconditions. While prior HTN learning approaches have focused on speed of learning, we instead would like to take advantage of the wealth of replays available to us, in the belief that, with enough data, meaningful patterns will emerge.

Evaluation

There are a few different evaluation metrics that I plan to use for this project. The first is simple strength of gameplay. If we are learning meaningful methods and goals, this should correspond to an increase in gameplay ability, which can be concretely measured against other artificial agents, as well as against human players via online ladder rankings. The second, more human-expensive method, is to have a domain expert analyze the generated methods to determine whether or not they correspond with true goals that human players work towards within a game. The final method would be to use annotated replays, where humans have specified what goals they were following with each action, in order to learn our methods and goals. We can then compare our goals to

the human annotations, and analyze the differences or similarities. Some of these databases already exist in use in other systems, such as Darmok as mentioned above.

Conclusion

In conclusion, I hope to extend research on learning HTN models from demonstration into new and more complex domains, which have qualitatively different characteristics and challenges than the existing work. These complications are similar in nature to the ones we would encounter in attempting to apply HTN learning to real-world environments, and as such this is an important problem. In addition, the strengths on which I hope to base my approaches (in particular, large amounts of expert demonstrations) align with the general direction of the field, with more and more data-driven approaches being found to be useful in practical environments. As a result, I feel that progress in this area is both possible and useful to the general public, and as such am excited to be working on it going forward.

References

Fernández, S.; Aler, R.; and Borrajo, D. 2005. Machine learning in hybrid hierarchical and partial-order planners for manufacturing domains. *Applied Artificial Intelligence* 19(8):783–809.

Garland, A., and Lesh, N. 2003. Learning hierarchical task models by demonstration. *Mitsubishi Electric Research Laboratory (MERL), USA–(January 2002)*.

Garland, A.; Ryall, K.; and Rich, C. 2001. Learning hierarchical task models by defining and refining examples. In *Proceedings of the 1st international conference on Knowledge capture*, 44–51. ACM.

Hoang, H.; Lee-Urban, S.; and Muñoz-Avila, H. 2005. Hierarchical plan representations for encoding strategic game ai. In *AIIDE*, 63–68.

Hogg, C.; Kuter, U.; and Munoz-Avila, H. 2010. Learning methods to generate good plans: Integrating htn learning and reinforcement learning. In *AAAI*.

Hogg, C.; Munoz-Avila, H.; and Kuter, U. 2008. Htn-maker: Learning htns with minimal additional knowledge engineering required. In *AAAI*, 950–956.

Ilghami, O.; Nau, D. S.; Munoz-Avila, H.; and Aha, D. W. 2002. Camel: Learning method preconditions for htn planning. In *AIPS*, 131–142.

Ilghami, O.; Munoz-Avila, H.; Nau, D. S.; and Aha, D. W. 2005. Learning approximate preconditions for methods in hierarchical plans. In *Proceedings of the 22nd international conference on Machine learning*, 337–344. ACM.

Lau, T.; Bergman, L.; Castelli, V.; and Oblinger, D. 2004. Sheepdog: learning procedures for technical support. In *Proceedings of the 9th international conference on Intelligent user interfaces*, 109–116. ACM.

Lee, H.; Grosse, R.; Ranganath, R.; and Ng, A. Y. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 609–616. ACM.

Meijer, A., and Koppelaar, H. 2001. Pursuing abstract goals in the game of go. *BNAIC01 Sponsors*.

Musen, M. A. 1989. Automated support for building and extending expert models. *Machine Learning* 4(3-4):347–375.

Ontanón, S.; Bonnette, K.; Mahindrakar, P.; Gómez-Martín, M. A.; Long, K.; Radhakrishnan, J.; Shah, R.; and Ram, A. 2009. Learning from human demonstrations for real-time case-based planning.

Ontañón, S.; Mishra, K.; Sugandh, N.; and Ram, A. 2010. On-line case-based planning. *Computational Intelligence* 26(1):84–119.

Smith, S. J.; Nau, D.; and Throop, T. 1998. Computer bridge: A big win for ai planning. *Ai magazine* 19(2):93.

Srikant, R., and Agrawal, R. 1996. *Mining sequential patterns: Generalizations and performance improvements*. Springer.

Weber, B. G.; Mateas, M.; and Jhala, A. 2012. Learning from demonstration for goal-driven autonomy. In *AAAI*.

Weber, B. 2012. *Integrating learning in a multi-scale agent*. Ph.D. Dissertation, UC Santa Cruz.

Willmott, S.; Richardson, J.; Bundy, A.; and Levine, J. 1999. An adversarial planning approach to go. In *Computers and Games*. Springer. 93–112.

Zhuo, H. H.; Hu, D. H.; Hogg, C.; Yang, Q.; and Munoz-Avila, H. 2009. Learning htn method preconditions and action models from partial observations. In *IJCAI*, 1804–1810.