HTN Learning via Transfer Learning of Domain Landmarks

Greg Pennisi,[1] Morgan Fine-Morris,[2] Michael W. Floyd,[1]
Bryan Auslander,[1] Héctor Muñoz-Avila,[2] Jeff Heflin[2] and Kalyan Moy Gupta[1]
[1] Knexus Research Corporation, National Harbor, MD, USA
[2] Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, USA

Abstract

Hierarchical Task Network (HTN) planning uses task-subtask relationships to break complex problems into more manageable subtasks, similar to how human problem-solvers plan. However, one limitation of HTN planning is that it requires domain knowledge in the form of *planning methods* to perform this task decomposition. Recent work has partially alleviated this knowledge engineering requirement by learning HTN methods from traces of observed behavior. Although this greatly reduces the amount of knowledge that must be encoded by a domain expert, it requires a large collection of traces in order to infer important landmark states that are used during trace segmentation and method learning. In this paper we present a novel method for landmark inference that transfers knowledge of landmarks from previously encountered environments to new environments without requiring any traces from the new environment. We evaluate our work in a logistics planning domain and show that our approach performs comparably to the existing landmark inference method but requires far fewer traces.

1 Introduction

Human problem solving rarely treats complex tasks as monolithic assignments that must be solved at once, but instead subdivides them into a hierarchy of more manageable and simpler subtasks (Choi and Langley 2005). Such a hierarchical decomposition of problems is beneficial because it allows reasoning over multiple alternatives for achieving an individual subtask, reuse of similar skills across a variety of problems, and efficient generation of high-level plans without worrying about all low-level details. Automated planning algorithms have attempted to replicate the problem solving abilities of humans by using similar problem decomposition approaches. Hierarchical Task Network (HTN) planning uses higher-level planning *methods* that encode task-subtask relationships that allow for a hierarchical breakdown of complex tasks into simpler subtasks and finally primitive tasks (accomplished by action). Similarly, *domain landmarks* are based on the idea that while there may be multiple paths that can be taken for multiple problems in a domain, there are certain landmark states that must be reached along all of those paths. Although both of these are valuable approaches to simplify the planning process, they require a domain expert to encode additional information in the form of planning methods and landmark locations. Our recent research has led to the development of a state-of-the-art algorithm for *learning planning methods from existing behavior traces* (Fine-Morris et al. 2020). This algorithm infers domain landmark locations from a set of behavior traces, uses those landmarks to partition the traces, and then learns planning methods based on those subtraces. One limitation of this algorithm is that its landmark identification requires a large set of behavior traces to identify which states are part of many (or all) of the traces. If only a limited set of behavior traces exist, landmark identification will either not be possible or produce lower quality domain landmarks.

In this paper we propose a novel domain landmark identification method that does not rely on behavior traces from the current environment to identify landmarks but instead learns to classify states as landmarks or non-landmarks based on their high-level properties. Our approach leverages knowledge of landmarks from other environments and transfers that knowledge to the current environment, thereby allowing rapid landmark identification in previously unseen environments. These inferred landmarks can then be used by other algorithms, such as an HTN learning algorithm or landmark-based automated planner. Our work makes the following key contribution: *a novel method for identifying high-quality landmark states that does not rely on an existing collection of behavior traces but instead uses information from previously identified landmarks.*

In the remainder of this paper we describe related work, followed by our domain landmark identification method and how it can be used in combination with a state-of-the-art HTN method learning algorithm. Section 3 presents our novel landmark identification process, which is later evaluated in a logistics domain in Section 4. We finish with concluding remarks in Section 5.

2 Related Work

In our work we focus on domain landmarks: landmarks for collections of problems in a domain. For example, if a city

is split into two parts by a river and there is only one bridge connecting the two parts, any path going from one of the city's parts to the other one must go through the bridge. Hence, being at the bridge is a landmark. There has been prior work done that identifies planning landmarks for a given problem. In contrast, in this work we are interested in landmarks across multiple problems.

Casting the learning problem of learning domain landmarks as word embeddings has been explored in previous work (Fine-Morris et al. 2020; Gopalakrishnan, Muñoz-Avila, and Kuter 2018). They receive as input a collection of annotated plan traces. Each annotated trace is a sequence of $s\ a\ s'$, state - action - state, such that a is applicable in s and s' is the state resulting from applying a into s. The basic premise is to view a collection of annotated plan traces as text, each annotated plan trace as a sentence and each individual atom in the state and each action as a word. This input is given to Word2Vec (Mikolov et al. 2013) to generate vector representations of these atoms. Crucially, atoms that frequently co-occur will have a similar vector orientation. Using clustering algorithms on the distance between the vectors' orientations, clusters of atoms can be identified. The atoms that are the closest between two neighboring clusters usually correspond to planning landmarks. The procedures differ in scope. For example, Fine-Morris et al. (2020) allows learning methods with an arbitrary number of subtasks whereas the other approach (Gopalakrishnan, Muñoz-Avila, and Kuter 2018) can only learn two subtasks per method. In contrast to those works, in this paper we transfer landmarks between domains so we can construct HTN methods on domains for which no input training traces are given.

A variety of procedures have been explored to learn HTNs. Some are incremental such as ICARUS (Choi and Langley 2005) and HTN-Maker (Hogg, Munoz-Avila, and Kuter 2008), which augment their HTN knowledge with new training instances. In ICARUS, learning occurs as a result of the system detecting gaps in its HTN knowledge. In HTN-Maker, methods are learned whenever new annotated traces are given. Other systems are not incremental and instead learn methods by analyzing a collection of annotated traces. The systems described in the previous paragraph behave that way. Another such system is HTNLearn (Zhuo, Muñoz-Avila, and Yang 2014), which cast the HTN learning problem as a constraint satisfaction problem such that when it is solved (e.g., by a constraint solver such as MAXSAT (Borchers and Furman 1998)), it will learn HTN methods. Our work is also non-incremental. It uses landmark properties from the source domain to identify landmarks in the target domain. The most important difference versus all previous work is that our approach does not require training examples in the target domain to identify landmarks.

3 Landmark Transfer

The primary motivation for our work is to provide a landmark inference approach that can be used by existing algorithms that learn hierarchical task networks (HTNs). More specifically, we focus on providing landmarks for our existing algorithm for learning HTNs (Fine-Morris et al. 2020), although our approach is not coupled to this algorithm and can be used in any application that requires landmark inference.

3.1 HTN Method Learning

The algorithm takes in three inputs: a set of observed traces T, a set of action operators A, and a set of tasks G. Each trace $t_i \in T$ is an alternating sequence of states (s) and actions (a) that were collected while observing an agent in the environment, where the agent is assumed to perform an action in response to each encountered environment state. If trace t_i contains m_i observed actions, then $t_i = s_0\ a_1\ s_1 \ldots s_{m_i-1}\ a_{m_i}\ s_{m_i}$. The set of actions A provide the actions that may be performed in the environment and each action is a tuple $\langle$*name, preconditions, effects*$\rangle$, where *name* is the name of the action, *preconditions* are the conditions of the state that must hold for the action to be performed, and *effects* are the changes to the state that occur when the action is performed. The tasks in G represent a set of specific tasks than an agent may attempt to achieve in the environment. The output of the algorithm is a set of HTN methods H, where each method in H is a tuple $\langle$*name, preconditions, subtasks*$\rangle$. Since HTN methods are hierarchical, the list of *subtasks* contains the lower-level tasks that should be achieved and may include both primitive subtasks (i.e., those achievable using an action) or non-primitive subtasks (i.e., those requiring an HTN method to decompose the subtasks into simpler subtasks).

The algorithm has the following five steps: (1) identify landmark states from raw traces, (2) determine the goal achieved by each trace, (3) split the traces into subtraces based on the identified landmarks, (4) learn right-recursive and single-subtask methods from the subtraces, and (5) learn landmark methods from each trace. After the learning process is complete, the algorithm returns the set H of learned methods containing landmark methods, right-recursive methods, and single-subtask methods. Our description of this algorithm was only meant to provide a high-level summary of how it operates. For full details, please refer to the original publication (Fine-Morris et al. 2020).

This HTN learning algorithm is able to learn HTN methods with as few as a single training trace. However, the algorithm's approach to identifying landmark states (Step 1) requires analysing numerous traces to identify states that are commonly reached. Thus, while the algorithm can learn with limited training traces, in practice it is unable to due to the landmark identification process it uses. To overcome this limitation, and to allow the algorithm to be practically used with limited training traces, we propose a novel landmark identification approach that transfers landmark identification knowledge from other environments.

3.2 Labelling Source Landmarks

We define the *source environment* as the environment for which some landmark knowledge already exist and the *target environment* as the environment that requires identification of landmarks. We assume the existence of a labelling function lab_s that labels any state $s_s \in \mathcal{S}_s$, where $\mathcal{S}_s \subset \mathcal{S}$, from the source environment as either a *landmark* or *non-landmark*: $lab_s : \mathcal{S}_s \rightarrow \{landmark, non{-}landmark\}$.

This labelling function could be provided, learned (e.g., like how the HTN algorithm identifies landmarks), or the result of expert labelling. Thus, each state in the source environment can be represented as a state-label pair: $\langle s_s, lab_s(s_s)\rangle$.

These state-label pairs could be used to train a classifier to discriminate between landmark states and non-landmark states. However, the complexity of the state representation may be sufficiently high such that the classification problem becomes difficult or computationally complex. Instead, we represent the state using higher-level *state metrics* that reduce the complexity of the state encoding. Our use of state metrics is motivated by previous work that found such metrics can reduce the representation complexity in realistic environments and simplify the reasoning that needs to occur over those representations (Floyd et al. 2017). We assume the availability of r such metric functions $f_1, \ldots, f_r$, each of which computes a single high-level metric of any state $s \in \mathcal{S}$: $f_1, \ldots, f_r : \mathcal{S} \rightarrow [0, 1]$. No restrictions are placed on the types of computations performed by these metrics and they can range from domain-independent to highly domain-dependent. Using the available state metrics, a state s can be represented as the state encoding $e_s = \langle f_1(s), \ldots, f_r(s)\rangle$. Similarly, a state-label pair $\langle s_s, lab_s(s_s)\rangle$ from the source environment can be converted to the encoding-label pair $\langle e_{s_s}, lab_s(s_s)\rangle$, since we assign the state encoding the same label as the original state.

3.3 Target Landmark Classification

The encoding-label pairs, which represent the landmark label knowledge from the source environment, are then used to train a binary classifier that predicts whether a state is a *landmark* or *non-landmark*. We make no assumptions about the specific classifier that is used, only that it: (a) takes as input a feature vector of length r where each feature is a value in the range $[0, 1]$, (b) produces a binary classification to discriminate between landmarks and non-landmarks, and (c) can use $|\mathcal{S}_s|$ encoding-label pairs during training. Outside of those requirements, any classification algorithm may be used. The resulting classifier is used as a labelling function lab_t that labels any state encoding $e \in \mathcal{E}$ as either a *landmark* or *non-landmark*: $lab_t : \mathcal{E} \rightarrow \{landmark, non{-}landmark\}$.

Thus, given a new target environment, landmark states in that environment are extracted as follows:

1. **Encode states**: For all states in the target environment, generate their state encoding: $\forall s_t \in \mathcal{S}_t$, where $\mathcal{S}_t \subset \mathcal{S}$, $e_{s_t} = \langle f_1(s_t), \ldots, f_r(s_t)\rangle$. $\forall s_t \in \mathcal{S}_t \exists e_{s_t} \in \mathcal{E}_t$.
2. **Label states**: Use the classifier to generate labels for each of the state encodings: $\forall e_{s_t} \in \mathcal{E}_t$, $\langle e_{s_t}, lab_t(s_t)\rangle$.
3. **Extract landmarks:** For any state s_t where $lab_t(s_t) =$ *landmark*, add s_t to the list of landmarks L.

Returning to the algorithm described in Section 3.1, this process would be used in place of Step 1 and produce a list of landmarks that will be used by the remaining steps of the algorithm.

4 Evaluation

Our evaluation looks to examine how landmark quality impacts planning performance when using the HTN planning methods that were learned using an HTN learning algorithm. More specifically, we examine the following experimental hypotheses: **H1:** When using our landmark inference approach, the performance will be lower but comparable to when high-quality landmarks are used; **H2:** When using our landmark inference approach, the performance will be higher than if low-quality landmarks are used; and **H3:** Using landmarks, even if they are lower quality, will result in increased performance than if no landmarks are provided.

4.1 Domain

The domain we use for our experiments is a simulated logistics domain where a truck navigates between locations on a map by performing movement actions. *Locations* on the map are represented by nodes and *connections* as edges, with each edge being equidistant. Goals represent locations a truck should move to.

In this domain we use the following state metrics: *connectivity* (i.e., the number of connections the location has), *local clustering coefficient* (i.e., how tightly clustered the location is with its neighbors), *average neighbor connectivity* (i.e., the average connectivity of the location's neighboring locations), *closeness centrality* (i.e., how close the location is to all other locations on the map), and *distance to center* (i.e., the distance of the location from the center of the map). These metrics were selected because they are well-established graph metrics, and thereby usable in any graph-based representation, rather than being coupled to the particular domain we used for evaluation.

4.2 Experimental Conditions

Our experiments compare four landmark selection methods: (1) **Frequently Occurring Landmarks (FREQ)** - the landmark selection method that was originally used by the HTN learning algorithm where it examines a large set of behavior traces to find states that appear frequently in those traces and labels them as landmarks; (2) **Inferred Landmarks (INF)** - our novel landmark inference algorithm described in this paper; (3) **Random Landmarks (RAND)** - A fixed number of states are labelled as landmarks using a uniform random distribution; and (4) **No Landmarks (NONE)** - an empty set of landmarks is provided.

The *INF* method relies on learning a landmark classifier that transfers knowledge from previously encountered environments. To train the classier, 10 maps were randomly generated, each with a mean of *N* locations (where N varies depending on the experiment). These represent previously encountered environments. For each of the 10 maps, traces of a truck performing shortest-path planning to 250 randomly selected goal locations were generated. Those traces were used by the *FREQ* approach to identify landmark states, and then all states are encoded and labelled as either landmarks or non-landmarks. The state encodings from all 10 maps were used to train a decision tree classifier, which is used during the evaluation process.

Our evaluation is repeated 100 times with each evaluation run using a different randomly generated map, each with a mean of *N* locations. Whereas the maps used to train the classifier represent past environments, these maps represent

Table 1: Average planning time (*Time*), average plan length (*Length*), and average methods learned (*Methods*) on map sizes 15, 30, and 60 for all four landmark selection methods.

	Time			**Length**			**Methods**		
	15	*30*	*60*	*15*	*30*	*60*	*15*	*30*	*60*
FREQ	0.2	0.6	2.4	3.0	3.8	4.3	12.5	14.2	14.8
INF	0.2	0.6	2.5	3.0	3.6	4.5	12.5	14.2	15.5
RAND	0.2	0.7	4.2	2.7	4.0	5.5	9.0	15.0	19.2
NONE	0.3	0.9	5.0	3.4	5.1	6.8	13.2	19.8	24.9

the current environment. For each experimental run (i.e., each map), a trace of a truck completing 250 additional goals is generated and provided to the *FREQ* method (to provide it with sufficient information to operate, giving it an advantage over the other methods). Candidate landmarks are computed for a map by each method, and those landmarks are used to learn HTN methods using the previously described HTN learning algorithm. 20 additional goals are generated and plans are generating using the four sets of HTN methods (i.e., one set from each of the landmark inference approaches). The planning performance is measured for the HTN methods generated by each approach using the following metrics: *average planning time (seconds)*, *average plan length (actions)*, and *average methods learned*.

4.3 Results

Our results for maps of an average size of N=15, N=30, and N=60 are shown in Table 1. Across all metrics and map sizes, the approaches that provide non-empty landmark sets (*FREQ*, *INF*, and *RAND*) are statistically significant improvements over *NONE* (using a paired t-test). This shows that any landmarks, even randomly selected ones, are better than none, providing strong support for **H3**.

Across nearly all map sizes and metrics, there are small but not statistically significant differences between *FREQ* and *INF*, with *FREQ* tending to perform slightly better. The one exception is the *average plan length* metric when *N=30*, where *INF* shows significantly better results than *FREQ*. How close the results were between *FREQ* and *INF* was unexpected, given that *FREQ* was always provided data from the current map (i.e., the traces of 250 goals being achieved), whereas *INF* only had data from the 10 previously observed training maps. These results provide partial support for **H1** since *INF* actually outperformed our expectations, which provides better overall support for the benefit of our landmark inference method.

Comparing the various methods of landmark inference, the informed methods (*FREQ* and *INF*) generally perform better than selecting landmarks at random. The one exception to this trend is on maps of size *N=15*. For these smaller maps, *RAND* has the lowest average plan length and the lowest average number of methods learned. However, as the map size increases to *N=30* and *N=60*, *RAND* performs significantly worse. We believe this occurs because landmarks become more important as the size of the map increases and true landmarks begin to emerge. More specifically, the maps begin having sub-neighborhoods form that can only be traversed between by passing through a landmark location. For smaller maps, landmarks are less important since there are generally multiple short paths between any pair of locations. This provides partial support for **H2** since our landmark inference approach does not improve performance for small maps.

5 Conclusions

In this paper, we presented a novel method for inferring landmark states that transfers landmark identification knowledge from previously encountered (source) environments to the current (target) environment. Our evaluation used the landmarks produced by our approach as part of an existing state-of-the-art HTN learning algorithm in a logistics domain. The results demonstrated that even though our approach required far fewer input traces in the target environment, it performed comparably to the existing data-heavy landmark inference approach. We plan to evaluate our landmark inference approach in more complex and varied domains as part of future work.

Acknowledgments

This research was supported in part by ONR N68335-19-C-0570, N00014-18-1-2009 and N68335-18-C-4027, and NSF grant 1909879. The opinions in this paper are from the authors and not necessarily from the funding agencies.

References

Borchers, B., and Furman, J. 1998. A two-phase exact algorithm for max-sat and weighted max-sat problems. *Journal of Combinatorial Optimization* 2(4):299–306.

Choi, D., and Langley, P. 2005. Learning teleoreactive logic programs from problem solving. In *International Conference on Inductive Logic Programming*, 51–68. Springer.

Fine-Morris, M.; Auslander, B.; Floyd, M. W.; Pennisi, G.; Munoz-Avila, H.; and Gupta, K. M. 2020. Learning hierarchical task networks with landmarks and numeric fluents by combining symbolic and numeric regression. In *Proceedings of the 8th Annual Conference on Advances in Cognitive Systems*.

Floyd, M. W.; Karneeb, J.; Moore, P.; and Aha, D. W. 2017. A goal reasoning agent for controlling UAVs in beyond-visual-range air combat. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 4714–4721. AAAI Press.

Gopalakrishnan, S.; Muñoz-Avila, H.; and Kuter, U. 2018. Learning task hierarchies using statistical semantics and goal reasoning. *AI Communications* 31(2):167–180.

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

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Zhuo, H. H.; Muñoz-Avila, H.; and Yang, Q. 2014. Learning hierarchical task network domains from partially observed plan traces. *Artificial intelligence* 212:134–157.