Rapid Acquisition of Hierarchical Plan Knowledge

Pat Langley

Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306
http://www.isle.org/~langley/

Advances in machine learning often result from innovations in knowledge representation. Here we review recent work on *hierarchical problem networks* or *HPNs*, which encode knowledge about how to decompose planning tasks into subproblems and use this content to solve new tasks. We also report HPNL, an implemented system that acquires such expertise in an incremental, piecemeal, and rapid manner from sample solutions, along with experimental studies of its learning behavior.

In this framework, procedural knowledge comprises a set of conditional *methods* that decompose problems – sets of goals – into simpler subproblems. In particular, each HPN method includes:

- A *head* that describes a goal the method aims to achieve;
- An *operator* O that accomplishes this goal when applied;
- A *subproblem* containing goals based on O's conditions;
- *State conditions* for when to use the decomposition; and
- *Unless conditions* of goals that must *not* be unsatisfied.

For a method to be relevant, all of its state conditions must be satisfied in the current situation but none of its unless conditions may be satisfied. The former are used mainly to select among possible decompositions; the latter are used primarily to order them in ways that avoid goal interactions.

Problem solving with HPNs involves search through a space of hierarchical plans that achieve top-level goals from an initial state. The key difference from HTN planning is that HPNs rely on a stack of *problems* rather than tasks or individual goals, which supports checking of unless conditions that avoid goal interactions. Backtracking can arise when the HPN's methods have overly general state conditions or when they lack necessary unless conditions. However, given appropriate knowledge about a domain, an HPN planner mimics a deterministic procedure in selecting a useful decomposition at each step. Langley and Shrobe (2021) have reported HPD, a hierarchical problem solver that implements this approach to knowledge-guided planning.

We have also developed HPNL (Langley, 2023), a system that acquires new HPN methods from a sequence of sample solutions, each having the form of a hierarchical plan that solves a given problem. Learning involves four processes:

- *Identifying HPN structure*. For each subplan that achieves a goal G by applying an operator O, HPNL produces a new method with G as its head, O as its operator, and O's conditions as its subproblem. Unlike schemes for learning HTNs, there is no need to decide when two methods should have the same head, as this is determined by the goal that each one achieves.
- *Inferring dynamic state conditions*. The main purpose of state conditions is to ensure all arguments in a method's operator are bound. To find state conditions for a method *M* with operator *O*, for each argument of *O* not bound in *M*'s head, HPNL finds relations that held when the sample decomposition occurred but that are inconsistent with *O*'s conditions. Inconsistency is based on domain constraints extracted from operator definitions.
- *Finding static state conditions*. If some operator arguments still remain unbound, HPNL adds conditions based on static relations by chaining outward from literals with the unbound arguments that appear in the decomposition state. The system notes relations that contain such an argument, records any new arguments that they introduce, finds relations that contain them in turn, and continues until no new arguments arise.
- *Introducing unless conditions*. To identify unless conditions that constrain the ordering in which decompositions occur, HPNL examines the sequence in which a subproblem's goals were achieved in the sample solution. The system checks whether the composed conditions for a goal $G1$ achieved earlier would be 'clobbered' by the composed effects for a goal $G2$ achieved later. If so, then it adds an unless condition to $G2$'s method that requires the goal $G1$ is not unsatisfied.

Once HPNL has found state and unless conditions for a new method, it is generalizes the structure by substituting constant arguments with variables in a consistent manner. When a new method C is isomorphic to an existing method M, then it does not add C to memory but instead increments a counter for M. The problem solver uses this number to select among methods when more than one is applicable.

Experiments with on-line learning in three planning domains have provided evidence that HPNL acquires planning knowledge which substantially reduces both search and problem-solving time on novel problems from the same do-

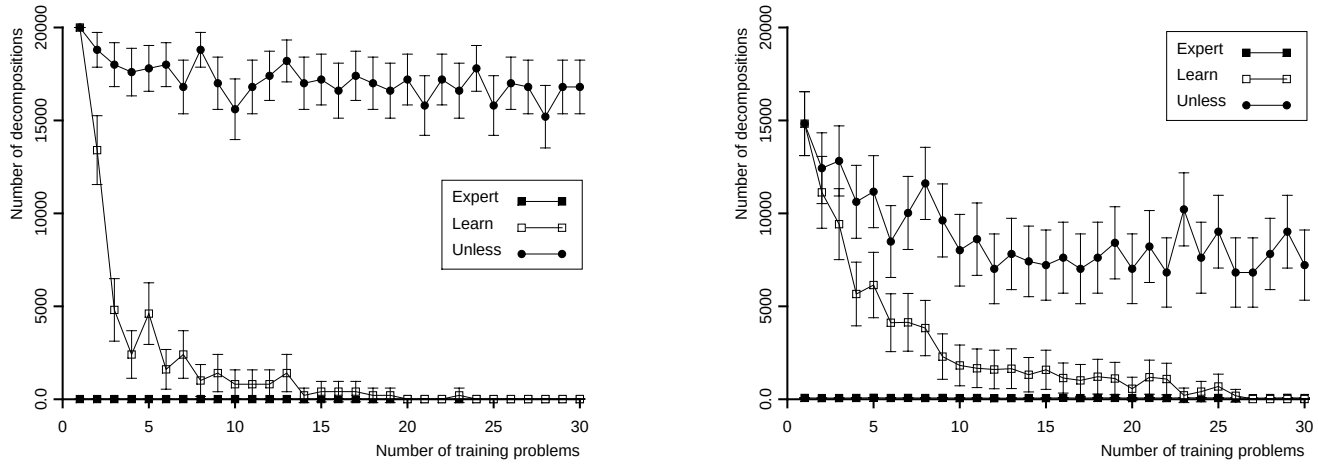

Figure 1: Number of decompositions needed to solve problems in Blocks World (left) and Logistics (right) by an expert HPN knowledge base, HPN methods learned from sample plans, and learned methods that lack *unless-goals* conditions. Each curve reports means and 95 percent confidence intervals over 100 random problem orders.

main. As shown in Figure 1, this improvement was rapid, with search on Blocks World and Logistics tasks being eliminated after tens of training problems. Moreover, CPU time was highly correlated with the number of decompositions considered, even in a recursive domain like the Blocks World, which suggests that the system does not suffer from the well-known utility problem. Search was not entirely eliminated in a third domain, Depots, but effort was still reduced substantially and learning was rapid.

HPNL's approach to learning differs from earlier methods for acquiring search-control knowledge (Langley, 1996), including ones that rely on explanation-based learning and inductive logic programming. The former relies on operator composition to identify conditions on control rules, which tends to produce overly specific knowledge. The latter uses relational induction to find more general conditions, but it requires multiple examples for comparison purposes. In contrast, HPNL invokes domain constraints to identify conditions on hierarchical methods from single subplans, which avoids both drawbacks. Other systems like ICARUS (Choi & Langley, 2018) acquire hierarchical methods incrementally from problem solutions, but they do not incorporate *unless* conditions that ensure goals are tackled in the proper order. In summary, HPNL learns knowledge for hierarchical planning in an effective and human-like manner.

Acknowledgements

This research was supported by Grant N00014-20-1-2643 from the US Office of Naval Research, which is not responsible for its contents. Some material from this abstract has appeared earlier in the papers by Langley (2023) and by Langley and Shrobe (2021) referenced below.

References

Choi, D., & Langley, P. (2018). Evolution of the ICARUS cognitive architecture. *Cognitive Systems Research*, *48*, 25–38.

Langley, P. (1996). *Elements of Machine Learning*. San Francisco, CA: Morgan Kaufmann.

Langley, P. (2023). Learning hierarchical problem networks for knowledge-based planning. *Proceedings of the Thirty-First International Conference on Inductive Logic Programming*. Windsor Great Park, UK.

Langley, P., & Shrobe, H. E. (2021). Hierarchical problem networks for knowledge-based planning. *Proceedings of the Ninth Annual Conference on Advances in Cognitive Systems*. Cognitive Systems Foundation.

Nau, D., Au, T-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, **20**, 379–404.