Logic-Based Inductive Synthesis of Efficient Programs

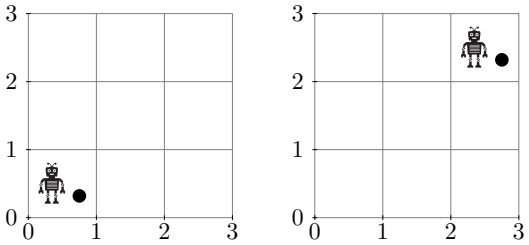
Andrew Cropper
Imperial College London, United Kingdom
a.cropper13@imperial.ac.uk

Abstract

Inductive programming approaches typically rely on an Occamist bias to select hypotheses with minimal textual complexity. This approach, however, fails to distinguish between the efficiencies of hypothesised programs, such as merge sort ($O(n\ log\ n)$) and bubble sort ($O(n^2)$). We address this issue by introducing techniques to learn logic programs with minimal resource complexity. We describe an algorithm proven to learn minimal resource complexity robot strategies, and we propose future work to generalise the approach to a broader class of programs.

1 Introduction

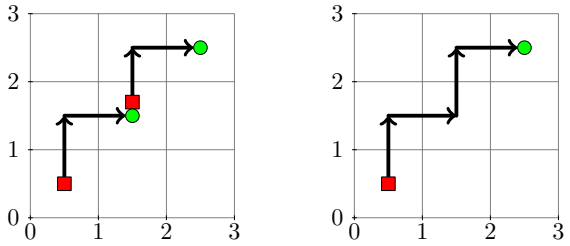
Suppose we are machine learning robot plans from initial/final state examples. Figure 1 shows a scenario where a robot is learning to move a ball from square 1/1 to square 3/3. Assume the robot can move *north*, *south*, *east*, and *west*, and can *grab* and *drop* the ball, then Figure 2 shows two plans with corresponding Prolog programs for this problem. Both programs transform the initial state to the final state and both have the same textual complexity. However, the programs differ in their efficiencies. Program (a) is inefficient because it involves two *grab* and two *drop* operations, whereas program (b) is efficient because it requires only one *grab* and one *drop* operation.



(a) Initial state (b) Final state

Figure 1: Robot planning example

However, most inductive programming approaches rely on an Occamist bias to select hypotheses with minimal textual complexity and cannot distinguish between the efficiencies of hypothesised programs. Clearly, learning efficient programs



(a) Inefficient solution (b) Efficient solution

Figure 2: Prolog programs for the planning example in Figure 1. A red square denotes a *grab* action and a green circle denotes a *drop* action.

is valuable in many AI domains. For example, if learning to sort lists, we want to learn an efficient strategy, such as merge sort ($O(n\ log\ n)$), rather than an inefficient strategy, such as bubble sort ($O(n^2)$), regardless of their respective textual complexities.

We address this issue by introducing techniques to learn minimal resource complexity logic programs. Resource complexity is a generalisation of the notion of time-complexity of algorithms, in which time is a particular resource. Our main contribution, thus far, is the introduction of a framework for minimising the resource complexity of logical robot strategies, and the demonstration of a learning algorithm proven to learn minimal resource complexity robot strategies.

2 Related work

Logic-based learning literature has addressed efficiently learning logic programs [Ahlgren and Yuen, 2013]. Likewise, AI planning literature has addressed developing efficient planners [Xing *et al.*, 2006]. By contrast, we want to learn logic programs which are optimal with respect to an objective function by which the quality of a hypothesis is measured. A common objective function, based on Occam's razor, is the length of the hypothesis, which in planning is often the number of actions required to execute a plan. However, if certain actions are costly, we may prefer a hypothesis which minimises the over-

all cost of the actions. Action costs have been used in answer set programming to learn optimal plans [Eiter *et al.*, 2003]. By contrast, we want to learn recursive programs, including recursive robot *strategies* [Cropper and Muggleton, 2015]. In contrast to traditional AI planning, which involves the generation of a plan as a sequence of actions transforming a particular initial state to a particular final state, a strategy is a potentially infinite set of plans, applicable to a class of initial/final state pairs. Although existing approaches support the construction of strategies [Laird, 2008], we are unaware of any approach which provides a provable convergent means for finding optimal strategies, nor of any approach which generalises to the broad class of programs which we propose.

3 Completed work

In [Cropper and Muggleton, 2015], we describe an approach to learn optimal resource complexity robot strategies based on the meta-interpretive learning (MIL) framework [Muggleton *et al.*, 2014; 2015; Cropper and Muggleton, 2016], a form of inductive logic programming which supports predicate invention and learning recursive theories. A strategy is a logic program composed of actions and fluents which transforms an initial state S_i to a final state S_j. The resource complexity of a strategy is defined as follows.

Definition 1 *Let $e = \langle S_1, S_2 \rangle$ be an example where S_1 and S_2 are initial/final states respectively and H a strategy. Then the resource complexity $r(H(e))$ is the sum of the action costs in applying the strategy H to e to transform S_1 to S_2.*

We assume a user-provided function $r : P_a \rightarrow \mathbf{N}$ which defines the resource cost of calling an action $p \in P_a$. In robot strategies, energy consumption and consumption of materials may be considered as resources. We define the resource complexity of a strategy over a set of examples.

Definition 2 *Let E^+ be a set of positive examples. Then the resource complexity of a strategy H is defined as follows:*

$$r(H, E^+) = arg\,max_{e \in E^+} r(H(e))$$

To find the minimal resource complexity strategy given a set of examples, we introduced Metagol$_O$, an implementation of the MIL framework which uses *iterative descent* to find resource optimal strategies. The main idea of iterative descent is to first find the minimal textual complexity strategy H_1, which is the most tractable to learn because the hypothesis space is exponential in the solution length [Lin *et al.*, 2014]. The resource complexity $r(H_1, E^+)$ of strategy H_1 provides an upper bound from which to descend, i.e. to search for a more efficient strategy with a resource complexity less than H_1. In [Cropper and Muggleton, 2015], we prove convergence of this search procedure to resource optimal strategies. Our experimental results agree with the theoretical optimal predictions and show, for instance, that when learning to sort lists, Metagol$_O$ learns an efficient quick sort strategy, rather than an inefficient bubble sort strategy.

4 Conclusions and future work

By focusing on robot strategies, we have made an initial attempt at inducing logic programs with optimal resource complexity. We intend to generalise the approach to a broader class of logic programs and non-logic programs.

In [Cropper and Muggleton, 2015], the resource complexity of a hypothesised strategy is maintained in the state description. Each dyadic action has an input state as the first term and an output state as the second term. Executing a dyadic action increments the resource cost in the input state to form the output state. However, predicates in logic programs do not not necessarily have *input* and *output* arguments, for instance when learning a monadic predicate. Therefore, to generalise the approach to arbitrary logic programs, we need a more general representation to calculate the resource complexity.

In addition, we have assumed a user-provided function to assign resource costs to each robot action. However, such information is not necessarily available and in future work we intend to investigate whether we can learn efficient time-complexity algorithms without user-provided costs.

We believe that the ideas proposed in this paper open exciting avenues in the emerging field of program synthesis, applicable to variety of AI domains. We also believe that this work raises the potential of discovering novel algorithms which are more efficient than existing ones.

References

[Ahlgren and Yuen, 2013] John Ahlgren and Shiu Yin Yuen. Efficient program synthesis using constraint satisfaction in inductive logic programming. *The Journal of Machine Learning Research*, 14(1):3649–3682, 2013.

[Cropper and Muggleton, 2015] Andrew Cropper and Stephen H. Muggleton. Learning efficient logical robot strategies involving composable objects. In *IJCAI 2015*, pages 3423–3429, 2015.

[Cropper and Muggleton, 2016] Andrew Cropper and Stephen H. Muggleton. Learning higher-order logic programs through abstraction and invention. In *IJCAI 2016*, 2016. To appear.

[Eiter *et al.*, 2003] Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. Answer set planning under action costs. *Journal of Artificial Intelligence Research*, pages 25–71, 2003.

[Laird, 2008] J. E. Laird. Extending the soar cognitive architecture. *Frontiers in Artificial Intelligence and Applications*, pages 224–235, 2008.

[Lin *et al.*, 2014] D. Lin, E. Dechter, K. Ellis, J.B. Tenenbaum, and S.H. Muggleton. Bias reformulation for one-shot function induction. In *Proceedings of the 23rd European Conference on Artificial Intelligence (ECAI 2014)*, pages 525–530, Amsterdam, 2014. IOS Press.

[Muggleton *et al.*, 2014] S.H. Muggleton, D. Lin, N. Pahlavi, and A. Tamaddoni-Nezhad. Meta-interpretive learning: application to grammatical inference. *Machine Learning*, 94:25–49, 2014.

[Muggleton *et al.*, 2015] Stephen H. Muggleton, Dianhuan Lin, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Machine Learning*, 100(1):49–73, 2015.

[Xing *et al.*, 2006] Zhao Xing, Yixin Chen, and Weixiong Zhang. Optimal strips planning by maximum satisfiability and accumulative learning. In *Proceedings of the International Conference on Autonomous Planning and Scheduling (ICAPS)*, pages 442–446, 2006.