1

One-Shot Learning of Temporal Action Models with Constraint Programming

Antonio Garrido and **Sergio Jiménez**

Abstract. This work presents a constraint programming (CP) formulation for the learning of temporal planning action models. This paper focus on the extreme scenario where just a single partial observation of the execution of a temporal plan is available (i.e. one-shot) because it is closely related to plan synthesis, plan validation and plan recognition. Our CP formulation models time-stamps for actions, causal link relationships (conditions and effects), threats and effect interferences that appear in planning. It also accommodates a different range of expressiveness, subsuming the PDDL2.1 temporal semantics. The formulation is solver-independent, meaning that an arbitrary CSP solver can be used for its resolution and is not only valid for learning, but also for plan validation.

1 Introduction

Temporal planning is a expressive planning model that relaxes the assumption of instantaneous actions of classical planning [5]. Actions in temporal planning have then durations and conditions/effects that must hold/happen at different times. This means that temporal actions can be executed in parallel and overlap in several ways [4] and that valid solutions for temporal planning problems need to indicate the precise time-stamp when an action starts and ends [6].

Despite the potential of state-of-the-art planners, its applicability to the real world is still somewhat limited because of the difficulty of specifying correct and complete planning models [8]. The more expressive the planning model is, the more evident becomes this knowledge acquisition bottleneck, which jeopardizes the usability of AI planning technology. This has led to a growing interest in the planning community for the learning of action models [7]. Most approaches for learning planning action models are purely inductive and often require large datasets of observations, e.g. thousands of plan observations to compute a statistically significant model that minimizes some error metric over the observations [11, 10, 12, 9]. Defining model learning as an optimization task over a set of observations does not guarantee completeness (the learned model may fail to explain an observation), nor correctness (the states induced by the execution of the plan generated with the model may contain contradictory information).

This paper analyzes the application of *Constraint Programming* for the *one-shot learning* of temporal action models, that is, the extreme case of learning action models from a single and partially specified model observed from the execution of a temporal plan. While learning an action model for classical planning means computing the actions' conditions and effects that are consistent with the input observations, learning temporal action models extends this to: i) identify how these conditions and effects are temporally distributed in the action execution, and ii) estimate the action duration. As a motivating example, let us assume a logistics scenario. Learning the temporal planning model will allow us: i) to better understand the insights of the logistics in terms of what is possible (or not) and why, because the model is consistent with the observed data; ii) to suggest changes that can improve the model originally created by a human, e.g. redistributing the actions' conditions, provided they still explain the observations; and iii) to automatically elaborate similar models for similar scenarios, such as public transit for commuters, tourists or people in general in metropolitan areas —*a.k.a.* smart urban mobility.

Contributions of our CP formulation are two-fold:

1. The first approach for learning actoin models for temporal planning and from observation that can refer to the execution of overlapping actions. Learning classical action models has been addressed by different approaches [2]. Since pioneering learning systems like ARMS [11], we have seen systems able to learn action models with quantifiers [1, 15], from noisy actions or states [10, 12], from null state information [3], or from incomplete domain models [13, 14]. But, to our knowledge, none of these systems learns the temporal features. This means that observations may now refer to the execution of overlapping durative actions, which makes our approach appealing for learning in multi-agent environments.
2. We evidence that the *one-shot learning* of planning action models is strongly related to the tasks of synthesizing, validating and recognizing plans. Our CP formluation allows that an arbitrary CSP solver can be used for the plan validation. This validation capacity is beyond the functionality of VAL (the standard plan validation tool [6]) because we can address *plan validation* of a partial (or even an empty) action model with a partially observed plan trace (VAL requires a full plan and a full action model for plan validation).

2 Background

3 Learning of Temporal Action Models with Constraint Programming

4 Results

5 Conclussions

REFERENCES

[1] Eyal Amir and Allen Chang, 'Learning partially observable deterministic action models', *Journal of Artificial Intelligence Research*, **33**, 349–402, (2008).
[2] Ankuj Arora, Humbert Fiorino, Damien Pellier, Marc Métivier, and Sylvie Pesty, 'A review of learning planning action models', *The Knowledge Engineering Review*, **33**, (2018).
[3] S. N. Cresswell, T.L. McCluskey, and M.M West, 'Acquiring planning domain models using LOCM', *The Knowledge Engineering Review*, **28(2)**, 195–213, (2013).

[1] Universitat Politècnica de València
Camino de Vera s/n. 46022 Valencia, Spain
{agarridot,serjice}@dsic.upv.es.

[4] William Cushing, Subbarao Kambhampati, Daniel S Weld, et al., 'When is temporal planning really temporal?', in *Proceedings of the 20th international joint conference on Artifical intelligence*, pp. 1852–1859. Morgan Kaufmann Publishers Inc., (2007).

[5] Maria Fox and Derek Long, 'Pddl2.1: An extension to pddl for expressing temporal planning domains', *Journal of artificial intelligence research*, **20**, 61–124, (2003).

[6] Richard Howey, Derek Long, and Maria Fox, 'VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL', in *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pp. 294–301. IEEE, (2004).

[7] Sergio Jiménez, Tomás De la Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo, 'A review of machine learning for automated planning', *The Knowledge Engineering Review*, **27**(4), 433–467, (2012).

[8] Subbarao Kambhampati, 'Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models', in *Proceedings of the National Conference on Artificial Intelligence (AAAI-07)*, volume 22(2), pp. 1601–1604, (2007).

[9] Jirí Kucera and Roman Barták, 'LOUGA: learning planning operators using genetic algorithms', in *Pacific Rim Knowledge Acquisition Workshop, PKAW-18*, pp. 124–138, (2018).

[10] Kira Mourão, Luke S. Zettlemoyer, Ronald P. A. Petrick, and Mark Steedman, 'Learning STRIPS operators from noisy and incomplete observations', in *Conference on Uncertainty in Artificial Intelligence, UAI-12*, pp. 614–623, (2012).

[11] Qiang Yang, Kangheng Wu, and Yunfei Jiang, 'Learning action models from plan examples using weighted MAX-SAT', *Artificial Intelligence*, **171**(2-3), 107–143, (2007).

[12] Hankz Hankui Zhuo and Subbarao Kambhampati, 'Action-model acquisition from noisy plan traces', in *International Joint Conference on Artificial Intelligence, IJCAI-13*, pp. 2444–2450, (2013).

[13] Hankz Hankui Zhuo and Subbarao Kambhampati, 'Model-lite planning: Case-based vs. model-based approaches', *Artificial Intelligence*, **246**, 1–21, (2017).

[14] Hankz Hankui Zhuo, Tuan Anh Nguyen, and Subbarao Kambhampati, 'Refining incomplete planning domain models through plan traces', in *International Joint Conference on Artificial Intelligence, IJCAI-13*, pp. 2451–2458, (2013).

[15] Hankz Hankui Zhuo, Qiang Yang, Derek Hao Hu, and Lei Li, 'Learning complex action models with quantifiers and logical implications', *Artificial Intelligence*, **174**(18), 1540–1569, (2010).