Towards Automated Planning Domain Models Generation

Lukáš Chrpa and **Mauro Vallati**
School of Computing and Engineering
University of Huddersfield, UK
{l.chrpa,m.vallati}@hud.ac.uk

Federico Cerutti
School of Natural & Computing Science
University of Aberdeen, UK
f.cerutti@abdn.ac.uk

Abstract

It is a common practice in Automated Planning to evaluate algorithms on existing benchmark domains. The number of domain models is limited, since they are encoding simplified versions of real-world domains and the generation of a new planning domain is a complex task.

The limited number of domain models does not allow to have a complete overview of the performances of automated planning engines. It would then be useful to have a generator of planning domain models for improving the evaluation of planning algorithms.

In this paper we introduce the requirements that an automatic generator of random domain models should fulfill, and we discuss the related works and the main issues that a domain models generator will have to face.

Introduction

In AI planning, algorithms are commonly evaluated only on a limited number of benchmark domain models, usually those that have been designed and used in an International Planning Competition (IPC) (Coles et al. 2012). They are inspired by real world domains with the ultimate aim of testing planning algorithms in everyday applications. However, usually the resulting models are very simplified, they share several similarities, and they are provided in a very limited number.

In this paper we introduce the requirements for a generator of planning domain models that will be able to overcome the limits stated above. These requirements will consider four situations where the usage of such a generator can be envisaged: (i) for achieving a better comprehension of algorithms performance, (ii) for configuring learning-based planners on large classes of domains; (iii) for configuring domain-independent portfolios on very large sets of domains, and (iv) for improving the current existing techniques for comparing planning models.

In the next section we introduce the existing techniques for generating domain models for planning. Then in the subsequent we discuss the main issues related to the automated generation of AI planning domains models. Finally we provide conclusions and future work.

Related works

The generation of a new planning domain model is a complex task. The traditional method involves an AI planning expert that uses a text editor for manually hand coding a set of previously gathered requirements that represent a real world domain. Recently, knowledge engineering tools such as GIPO (Simpson, Kitchin, and McCluskey 2007), itSIMPLE (Vaquero et al. 2012) and PDDL Studio (Plch et al. 2012) have been developed. These tools usually include techniques for analysis of structures of domain models. It is the case of itSIMPLE, where Petri Nets are exploited for analysing dynamic properties of the model, or GIPO, in which it is possible to check the correctness of invariants. It is clear that these KE tools are designed for supporting users in the models generation task, while in this paper we are introducing the idea of an automated models generator. It is also worth to consider that, for humans, it is hard to design a new domain which is not somehow related to a real-world application. Moreover, a user will probably re-use solutions that he has previously adopted for encoding similar constraints, which will result in models with significant similarities. These facts represent a clear limit to the generation of new models by human users.

Considering the automatic generation of planning domain models, there exist several techniques for handling this problem, but all of them require the exploitation of some sort of existing knowledge. LOCM (Cresswell, McCluskey, and West 2013) is able to generate domain models from sample plan traces, for instance, LOCM is able to learn a Freecell domain model just by observing legal moves in several games. Opmaker2 (McCluskey et al. 2010) learns domain models from sample plans and partial domain models. Domain models can be learnt also from existing formal models (Barták, Fratini, and McCluskey 2010).

Differently from the existing automatic approaches for generating planning domain models, the final outcome of this research project will be a system able to automatically create new models that will be mainly exploited for improving the comprehension of planners performances (encoding domain models described in a non-formal language or encoding real-world problems are not considered at this stage).

To this aim, in the next section we introduce the design requirements for such a system.

Requirements for an Automated Planning Domain Models Generator

An automated planning domain models generator should address the following three issues: (i) how to define a well structured domain; (ii) the parameters that can be safely randomized, and the ones that have to be selected by humans; and (iii) the equivalence between models.

Insights into how actions, instances of planning operators, might be ordered in plans can be get by investigating operators' preconditions and effects. This mainly influences planning domain structures. Following Chapman's terminology (Chapman 1987) we can define a *possible achiever* and a *possible clobberer*. An operator o is a possible achiever (resp. clobberer) for another operator o' (o' does not have to be necessarily different from o) if and only if o creates (resp. deletes) a predicate for o' (operators o and o' must share corresponding arguments). Straightforwardly, a predicate in a precondition of some operator must be achieved by some other operator or be present in an initial state. Similarly, an operator should be an achiever for some other operator or a goal state. In other words, operators should be reachable (their instances should be applicable at some point) and useful (their instances should somehow contribute to the goal). Intuitively, achiever and clobberer relations among operators in a domain should be somehow balanced. These aspects might somehow ensure that the domain is well structured. However, it is also important to investigate computational complexity issues of determining whether a domain is well structured.

Regarding the second aspect, the system must provide configuration parameters that humans would select for testing planning algorithms on domain models with specific characteristics. It can determine, for instance, whether planning operators, when instantiated into actions, have a high level of interference between each other, or what restrictions the domain models would have. Specific restrictions of planning domains influence how difficult is to solve corresponding planning problems in terms of computational complexity (Bäckström and Nebel 1995; Bäckström et al. 2012). Computational complexity of known IPC benchmarks has also been studied (Helmert 2003; 2006). Despite proven tractability of some classes of problems planning engines tend to struggle with them. Therefore, a possibility of generating domains with proven tractability can somehow point to issues which might be characteristic for current domain-independent planning engines. Given the fact that the problem of determining how to design an optionally constrained domain have never been carefully analyzed, we believe that having an opportunity to generate domains with specific constraints might contribute to ongoing research dealing with structural analysis of planning domains/problems which might result in new planning techniques, heuristics and theoretical results. We will address this specific problem by studying the relations between (Roberts and Howe 2009)'s metrics of existing domain models that have been demonstrated to be constrained, thus providing also an innovative usage of these metrics.

Another important aspect of planning domain models is the reversibility of all the operators. Reversibility implies that is whether at any state, an operator's application can be reversed by another operator to result in the original state (Wickler 2011). If one operator is not reversible, this leads to dead ends in the search space of the planning problems. This is an aspect that is important to address in particular both for comparing planning models and for configuring domain-independent portfolios because planning domain models which have many dead ends in the search space are especially problematic for heuristic search based planners (Helmert 2004).

The third introduced aspect is related to the equivalence of domain models. Referring to the definitions given in (Shoeeb and McCluskey 2011), there are two different types of equivalence: strong and weak. The former defines two domain models that are identical up to naming, while the latter implies that the functional behaviour of the domain is the same for both models. This means that the same task can be formulated in both models and then the same solutions can be generated. It is clear that a domain models generator must avoid to generate strong equivalent models: indeed the analysis of the performance of planning algorithms on strong equivalent domains do not improve the knowledge on their behaviour. Therefore, if different strong equivalent domain models are used for testing planners performance, the resulting statistical model of their behaviour would be misleading. On the other hand, the impact of weak equivalent domain models on the planners evaluation is not clear as the strong one. Intuitively, it seems reasonable that the generation of weak equivalent models should be avoided by an automated system but more analysis are needed for a complete understanding of this issue.

Conclusions

The evaluation of planning algorithms is currently limited to existing benchmark domain models. This approach does not allow to have a complete overview of the performance of planners. An automated domain model generator would then be useful in several situations, which include algorithms evaluation and configuration of learning-based planners.

In this paper we introduced three general requirements for a domain models generator. First it has to build well-structured and exploitable domain models. Then, it has to be open to user customization, in order to generate planning domains with certain chosen characteristics (e.g. tractability, operators' reversibility, ...) that are believed to be useful for given purposes. Finally, it must be able to generate domains not equivalent to a given one.

Our future work includes more specific analysis of the domain models generation problem, such as the best way for representing and handling new models, and the development of a prototype of a domain generator. It will be also important to study the techniques that can be used for evaluating the quality of the new generated models, and the performance of the proposed system.

References

Bäckström, C., and Nebel, B. 1995. Complexity results for sas+ planning. *Computational Intelligence* 11:625–656.

Bäckström, C.; Chen, Y.; Jonsson, P.; Ordyniak, S.; and Szeider, S. 2012. The complexity of planning revisited - a parameterized analysis. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI-12)*, 1735–1741.

Barták, R.; Fratini, S.; and McCluskey, T. L. 2010. The 3rd competition on knowledge engineering for planning and scheduling. *AI Magazine* 31(1):95–98.

Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32(3):333–377.

Coles, A. J.; Coles, A.; Olaya, A. G.; Jiménez, S.; López, C. L.; Sanner, S.; and Yoon, S. 2012. A survey of the seventh international planning competition. *AI Magazine* 33(1).

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2013. Acquiring planning domain models using locm. *The Knowledge Engineering Review* FirstView:1–19.

Helmert, M. 2003. Complexity results for standard benchmark domains in planning. *Artificial Intelligence* 143(2):219–262.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 161–170.

Helmert, M. 2006. New complexity results for classical planning benchmarks. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS-06)*, 52–62.

McCluskey, T. L.; Cresswell, S.; Richardson, N.; and West, M. 2010. Action knowledge acquisition with opmaker2. In *Agents and Artificial Intelligence*, volume 67 of *Communications in Computer and Information Science*. Springer. 137–150.

Plch, T.; Chomut, M.; Brom, C.; and Barták, R. 2012. Inspect, edit and debug pddl documents: Simply and efficiently with pddl studio. In *System Demonstration – Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12)*.

Roberts, M., and Howe, A. 2009. Learning from planner performance. *Artificial Intelligence* 173(56):536 – 561.

Shoeeb, S., and McCluskey, T. L. 2011. On comparing planning domain models. In *The 29th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG-11)*.

Simpson, R.; Kitchin, D. E.; and McCluskey, T. 2007. Planning domain definition using GIPO. *Knowledge Engineering Review* 22(2):117–134.

Vaquero, T. S.; Tonaco, R.; Costa, G.; Tonidandel, F.; Silva, J. R.; and Beck, J. C. 2012. itSIMPLE4.0: Enhancing the modeling experience of planning problems. In *System Demonstration – Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12)*.

Wickler, G. 2011. Using planning domain features to facilitate knowledge engineering. In *Knowledge Engineering for Planning and Scheduling workshop (KEPS)*.