Planning and Scheduling, What's the Difference Anyway?

Anonymous submission

Abstract

Scheduling solvers are one of the success story of constraint programming. They have been widely used for decades, both in academia and industry, and known for providing best in class performance for a range of planning, scheduling, routing and allocation problems. Despite such problems having a widespread representation in AI planning benchmarks, there has been little penetration of scheduling solvers and techniques in the field. In this talk, we propose to explore why that might be the case.

Planning & Scheduling By *planning* we adopt the restrictive definition of AI planning with PDDL-like models as first and foremost represented in the International Planning Competition. For the scheduling, we typically consider state-of-the-art scheduling engines such as CPOptimizer and broadly consider as *scheduling* classes of problems where those provide state of the art performance (shop scheduling, logistics, routing, ...), that typically feature an organization of actions in time often coupled by others aspects such as resource availability, allocation, action selection, ...

Different Domains? A first observation is that a very significant part of planning domains (as seen in the IPC) are ones where scheduling solvers have state-of-the-art performance. Many application inspired domains (satellites, transport, tpp, tms, crew planning, rovers, ...) are variants of very established scheduling or routing problems (TSP, shop scheduling, team-orienteering, ...).

This is certainly not true of all problems. In particular, there are many game inspired problems (sokoban, tetris, labyrinth, freecell, solitaire, ...) and few other application problem (protein folding, quantum routing, ...) where the correspondence is not so obvious.

Nevertheless, on these scheduling-like problems, one could reasonably expect scheduling solvers to widely outperform PDDL planners, and hence have at least a place of choice in a portfolio approach. A telling witness of this is the extreme over-representation of scheduling techniques in the application tracks of ICAPS over the years.

Domain Independence? A common misconception, is that, unlike PDDL planners, CP solvers are not domain-independent. While this was historically true, most CP solvers today also rely on a model-and-solve approach, without any necessary tuning.

Different Models? A first important source of explanation is that while problems may be very adapted to scheduling solvers, their PDDL models are not. Some of these are due to specificities in the PDDL language: boolean first representation, real numbers, lack of numeric parameters and global constraints. Others derive from choices to build models adapted for the dominant explicit state-space search: sparse representation of navigation graphs, actions splitting to facilitate grounding or avoid durative actions.

Most of these limitations would in theory be addressable by domain analysis and preprocessing but significantly raise the barrier to good performance.

Bias in Problem Selection One last and perhaps most pervasive problem is related to the selection of instances in standard planning benchmarks. In the planning community, we often consider that a planner *solves* an instance when it returns a valid plan. This translates naturally in the dominance of coverage as an evaluation metric, and leads us to systematically include in benchmarks instances that hard to satisfy.

This is of course not to say that the planning community is blind to quality measures. The satisfying track of the IPC accounts for plan quality in its evaluation, but in the presence of hard to satisfy instances, it mostly serves as tie-breaking between solvers with very close coverage. On the other hand the optimal track of the IPC is not concerned with solution quality but with an optimality proof.

On the other hand, the scheduling community has a very different culture in evaluating solvers, with the primary metric being the (normalized) distance to the optimal solution. For most scheduling problems, and one could argue for most planning problems as well, this makes sense: when finding a solution is polynomial, the focus should be placed on the hard combinatorial part: finding high quality solutions.

Such difference in evaluation between the communities naturally lead to different strength in solvers, with the ones of scheduling solvers being hard to leverage on planning benchmarks.

Conclusion The objective of this talk is to shine some light one those aspects. In particular, many elements provided will try to explain the absence of scheduling techniques in mainstream planning approaches but also justify why they remain extremely relevant when considering practical applications.