CP4TP: A Classical Planning for Temporal Planning Portfolio

Daniel Furelos-Blanco and **Anders Jonsson**
Department of Information and Communication Technologies
Universitat Pompeu Fabra
Roc Boronat 138, 08018 Barcelona, Spain
{daniel.furelos, anders.jonsson}@upf.edu

Abstract

CP4TP is a portfolio temporal planner that sequentially runs different temporal planning algorithms until a solution is found. All the constituent temporal planners rely on a compilation to classical planning. The complexity of the compilations increases as we advance in the queue: from exclusively solving sequential problems to solving problems requiring simultaneous events.

This portfolio planner participates in the satisficing track of the International Planning Competition 2018.

Introduction

The International Planning Competition (IPC) has always played a major role in setting new milestones in the development of planning algorithms. In the case of temporal planning, the first domains were temporal versions of existing classical planning domains. Therefore, they could be sequentially solved.

The first temporal domains requiring concurrency at the IPC were in the form of single hard envelopes (Coles et al. 2009), which can be solved using specialized algorithms (Jiménez, Jonsson, and Palacios 2015). However, outside the IPC, there have been publications introducing more intricate kinds of concurrency (Cushing et al. 2007).

The fact that temporal problems can be divided in different types of increasing difficulty motivates the portfolio planner we present, CP4TP. The portfolio is formed by planners that rely on a compilation from temporal to classical planning. The complexity of the constituent planners increases as we advance in the queue: from solving sequential problems to solving problems requiring simultaneous events.

Background

This section briefly introduces the two forms of planning used by our planner: classical and temporal planning. Besides, the different kinds of temporal problems that CP4TP is able to solve are described, as well as how their compilations to classical planning work.

Classical Planning and Temporal Planning

A *classical planning* problem is of the form $\Pi = \langle F, A, I, G \rangle$ where F is a set of fluents, A is a set of actions, $I \subseteq F$ is an initial state and $G \subseteq F$ is a goal condition. Each action $a \in A$ has a precondition $\mathsf{pre}(a) \subseteq F$, an add effect $\mathsf{add}(a) \subseteq F$ and a delete effect $\mathsf{del}(a) \subseteq F$, each a subset of fluents. A plan for Π is a sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$.

A *temporal planning* problem is also a tuple $\Pi = \langle F, A, I, G \rangle$, where F, I and G are defined as for classical planning. However, each element $a \in A$ is a *temporal action* composed of:

- $d(a)$: duration,
- $\mathsf{pre}_s(a), \mathsf{pre}_o(a), \mathsf{pre}_e(a)$: preconditions of a at start, over all, and at end, respectively,
- $\mathsf{add}_s(a), \mathsf{add}_e(a)$: add effects of a at start and at end,
- $\mathsf{del}_s(a), \mathsf{del}_e(a)$: delete effects of a at start and at end.

The semantics of temporal actions can be defined in terms of two discrete *events* start_a and end_a, each of which is naturally expressed as a classical action as follows (Fox and Long 2003):

start_a: $\mathsf{pre} = \mathsf{pre}_s(a)$, $\mathsf{add} = \mathsf{add}_s(a)$, $\mathsf{del} = \mathsf{del}_s(a)$
end_a: $\mathsf{pre} = \mathsf{pre}_e(a)$, $\mathsf{add} = \mathsf{add}_e(a)$, $\mathsf{del} = \mathsf{del}_e(a)$

The duration $d(a)$ and precondition over all $\mathsf{pre}_o(a)$ impose restrictions on this process: end_a has to occur exactly $d(a)$ time units after start_a and $\mathsf{pre}_o(a)$ has to hold in all states between start_a and end_a.

A temporal plan is a set of action-time pairs $\pi = \langle (a_1, t_1), \ldots, (a_n, t_n) \rangle$. We say that π has *concurrent actions* if there exist two pairs (a_i, t_i) and (a_j, t_j) in π such that $t_i < t_j < t_i + d(a)$, i.e. a_j starts after a_i starts but before a_i ends. A *joint event* is composed of one or more individual events of π that all have the same associated time. We say that π has *simultaneous events* if at least one joint event is composed of multiple individual events.

The quality of a temporal plan is given by its *makespan*, i.e. the duration from the start of the first action execution to the end of the last action execution.

Sequential Temporal Problems

A temporal plan π is *sequential* if it does not contain concurrent actions, i.e. each action ends before

the next action starts. Formally, a temporal plan $\pi = \langle (a_1, t_1), \ldots, (a_n, t_n) \rangle$ is sequential if $t_{i-1} + d(a_{i-1}) < t_i$ for each $i, 1 < i \leq n$. A temporal planning problem Π is *sequential* if there exists a sequential plan π solving Π, and Π is *inherently sequential* if each plan π solving Π can be rescheduled to form a sequential plan.

To solve a sequential temporal problem, we can map each temporal action $a \in A$ to a classical action c_a that simulates all of a at once (Coles et al. 2009). The precondition of c_a is the union of the precondition at start of a with the preconditions over all and at end not achieved by the add effect at start. The effect of c_a is the effect at start of a followed immediately by its effect at end. Formally, the compressed action c_a is defined as follows:

- $\mathsf{pre}(c_a) = \mathsf{pre}_s(a) \cup ((\mathsf{pre}_o(a) \cup \mathsf{pre}_e(a)) \setminus \mathsf{add}_s(a))$,
- $\mathsf{add}(c_a) = (\mathsf{add}_s(a) \setminus \mathsf{del}_e(a)) \cup \mathsf{add}_e(a)$,
- $\mathsf{del}(c_a) = (\mathsf{del}_s(a) \setminus \mathsf{add}_e(a)) \cup \mathsf{del}_e(a)$.

By converting all temporal actions to classical actions, we get a classical planning problem $\Pi_c = \langle F, A_c, I, G \rangle$ where $A_c = \{c_a : a \in A\}$ is the set of resulting classical actions. A temporal plan π can be obtained from the resulting classical plan $\pi_c = \langle c_{a_1}, \ldots, c_{a_n} \rangle$ by choosing $t_1 = 0$ and $t_i = t_{i-1} + d(a_{i-1}) + u$ for each i such that $1 < i \leq n$, where u is a slack unit of time whose purpose is to separate the end of a temporal action from the start of the next action. Finally, the makespan of π can be optimized by rescheduling its actions.

Many instances of domains in the previous edition of IPC (Vallati et al. 2015) were of this kind (DRIVERLOG, FLOORTILE, MAPANALYSER, PARKING, RTAM, SATELLITE and STORAGE).

Single Hard Envelope Problems

A *single hard envelope* (Coles et al. 2009) (SHE) defines a form of action concurrency. Formally, a SHE is a temporal action a that adds a fluent f at start and deletes it at end.

Jiménez, Jonsson, and Palacios (2015) proposed the TPSHE planner to address temporal planning instances where concurrency is exclusively in the form of SHEs that provide the precondition over all of other temporal actions. In this case the temporal actions can be organized in a stack.

Several instances of domains in the previous IPC edition had concurrency in precisely this form (MATCHCELLAR, TMS and TURN&OPEN). Moreover, the TPSHE planner was shown to outperform all temporal planners in the temporal track of IPC-2014.

Other Forms of Concurrency

In general, temporal planning problems can have other forms of concurrency not captured by the SHE concept. Jiménez, Jonsson, and Palacios (2015) proposed TP, an adaptation of the TEMPO algorithm (Cushing et al. 2007), which adopts a forward search approach to generate temporal plans with general forms of concurrency. To reduce the branching factor of forward search, the authors proposed a parameterized version TP(k), $k \geq 2$, that allows at most k concurrent actions.

Although TP supports more forms of concurrency, it does not support simultaneous events such as in the Allen's interval algebra domain (Jiménez, Jonsson, and Palacios 2015). To solve problems requiring simultaneous events, Furelos-Blanco et al. (2018) proposed STP, an extension of TP which is capable of producing temporal plans with simultaneous events, as well as the parameterized versions STP(k), $k \geq 2$.

Methodology

The constituent planners are organized in a queue following this order:

1. Sequential temporal planner (SEQ).
2. TPSHE to solve problems with single hard envelopes (SHE).
3. TP(K), for $K \in \{2, 3, 4\}$, to solve problems with other kinds of concurrency different from SHE (except for simultaneous events).
4. STP(K), for $K \in \{2, 3, 4\}$, to solve problems with simultaneous events.

If a planner in the queue is not capable of solving a problem, then the next planner will be responsible for solving that problem. For example, if SEQ cannot solve a problem Π, TPSHE will try to do so.

Note that the later a planner appears in the queue, the more general it is. For instance, TPSHE can solve both sequential and SHE problems, TP(K) can solve the previous ones and problems with other kinds of concurrency. Then, why do not we try to solve everything using STP(K)? The answer is that the more general a planner is, the more machinery it involves. Therefore, the amount of time required by STP(K) to solve a sequential problem can be higher than for SEQ.

This trial and error approach is convenient because if problem Π cannot be solved using a certain planner P, then P fails immediately and the portfolio quickly tries to get a solution with the next planner. For example, if SEQ tries to solve a SHE problem, it quickly tells that no solution can be found.

Figure 1 shows the order of the planners and the time allocated to each of them:

- SEQ is given all time available to find a solution (t_{total}).
- If the problem cannot be sequentially solved, TPSHE is executed and it is given all the remaining time possible (t_{rem}).
- If the problem cannot be solved using TPSHE, TP(2), TP(3) and TP(4) are all given $t_{rem}/3$. That is, the remaining time is equally divided among them.
- If the problem cannot be solved using TP(K) planners, STP(2), STP(3) and STP(4) are all given $t_{rem}/3$. Again, the remaining time is equally divided.

The code of the different planners can be found at https://github.com/aig-upf/temporal-planning.

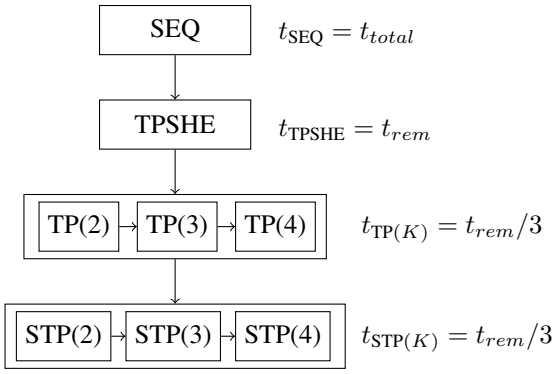

Figure 1: Schema of the portfolio planner.

Conclusions

We have introduced a portfolio planner formed by temporal planners which deal with problems of increasing temporal complexity. All constituent planners use a compilation from temporal to classical planning.

In future work, it would be interesting to find strategies for selecting the appropriate planner depending on the problem instead of using a trial and error approach.

Acknowledgments

This work has been partially supported by the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502).

References

Coles, A.; Fox, M.; Halsey, K.; Long, D.; and Smith, A. 2009. Managing concurrency in temporal planning using planner-scheduler interaction. *Artif. Intell.* 173(1):1–44.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. S. 2007. When is Temporal Planning Really Temporal? In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 1852–1859.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.* 20:61–124.

Furelos-Blanco, D.; Jonsson, A.; Palacios, H.; and Jiménez, S. 2018. Forward-Search Temporal Planning with Simultaneous Events. In *Proceedings of the 13th Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS) at ICAPS 2018, Delft, Netherlands, June 24-29, 2018.*

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.* 26:191–246.

Jiménez, S.; Jonsson, A.; and Palacios, H. 2015. Temporal Planning With Required Concurrency Using Classical Planning. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, 129–137.

Vallati, M.; Chrpa, L.; Grześ, M.; McCluskey, T. L.; Roberts, M.; and Sanner, S. 2015. The 2014 International Planning Competition: Progress and Trends. *AI Magazine* 36(3):90–98.