28th International Conference on
Automated Planning and Scheduling

June 24–29, 2018, Delft, the Netherlands

IPC 2018 – Temporal Tracks

Planner Abstracts for the

Temporal Tracks

in the

International Planning Competition 2018

Edited by:

Moisés Martínez, Andrew Coles and Amanda Coles

Organization

Moisés Martínez
King's College London, United Kingdom

Andrew Coles
King's College London, United Kingdom

Amanda Coles
King's College London, United Kingdom

Foreword

The International Planning Competition (IPC) is organized in the context of the International Conference on Planning and Scheduling (ICAPS). It empirically evaluates state-of-the-art planning systems on a number of benchmark problems. The goals of the IPC are to promote planning re-search, highlight challenges in the planning community and provide new and interesting problems as benchmarks for future research. In 2018 different tracks were organized more or less independently by different groups of organizers. This booklet describes the planners that participated in the temporal track of the IPC 2018.

There were two temporal track in total: the satisficing track where the goal was to find a (not necessarily optimal) plan with high quality. The on-board planning track where the goal was to find a plan with limited resources available, but this track was canceled because only two teams registered for it. A multi-core track was announced but unfortunately had to be canceled because only two teams registered for it. In parallel to the temporal track there were also a classical and probabilistic tracks organized separately.

This year planner were submitted as software containers using singularity (singularity.lbl.gov). We hope that this step will increase the reproducibility of results and simplify reusing the planners for future experiments. The competition was run on the cluster of the King's College London.

The competition results were announced at ICAPS, in June 2018, in Delft (Netherlands). Results, planners, log files, and detailed information on all tracks is available on our homepage

https://ipc2018-temporal.bitbucket.io/

Moisés Martínez, Andrew Coles and Amanda Coles
June 2018

Contents

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.

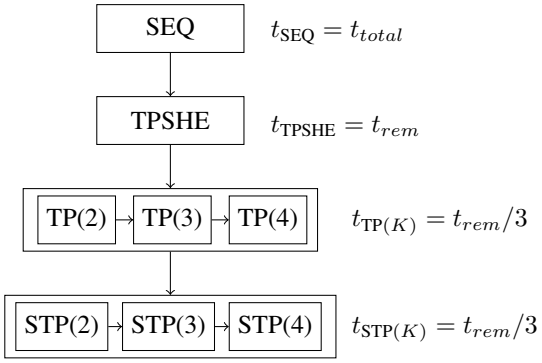

Figure 1: Schema of the portfolio planner.

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.

TFLAP: a temporal forward partial-order planner

Oscar Sapena, Eliseo Marzal, Eva Onaindia
Information Systems and Computation Dept.
Universitat Politècnica de València, Spain
E-mail: osapena,emarzal,onaindia@dsic.upv.es

Abstract

TFLAP is a forward-chaining temporal planner that follows the principles of the classical Partial-Order Causal-Link (POCL) paradigm. Working with partial-order plans is very advantageous when doing temporal reasoning as it allows to manage the parallelism of the plans in a natural way. This way, TFLAP can deal directly with concurrent tasks, where a total ordering on the actions of the plans is unfeasible, durative actions and timed initial literals. The main limitation of POCL planners is, however, the high computational effort required to cope with the interactions that arise among actions. On the other hand, the use of state-based heuristics embedded in the partial-order framework of TFLAP proves to be a good guidance, thus partially alleviating the burden of a complex machinery for handling action interaction.

Introduction

Over the last years, different extensions of classical planning to solving temporal planning problems have been proposed. One approach relies on parallelizing the plans obtained by a classical planner that applies a sequential reasoning on durative actions. The YAHSP planners family, DAE$_{YAHSP}$ (Khouadjia et al. 2013), YAHSP2 (Vidal 2011) and the two versions of YAHSP3 that participated in the IPC-2014 (Vidal 2014), follow this separation of action selection and scheduling, which prevents these planners from handling temporally expressive features as concurrent temporal domains. This was evidenced in the IPC-2014 results where these planners were not able to solve problems of the concurrent domains `TurnAndOPen` or `TMS`. Despite this, YAHSP3-MT was the winner of the temporal satisficing track of IPC-2014

The approach taken by TFD (Temporal Fast Downward) (Eyerich, Mattmüller, and Röger 2009) lies in the application of a forward search in the space of time-stamped states. TFD was the runner-up in the temporal satisficing track of IPC-2014. It ranked second in quality score and third in number of solved problems. TFD showed an impressive performance in some domains in which it was capable of solving all of the problems but on the contrary it could not solve any problem of five domains.

Using satisfiability checking is another important trend of classical planning research. ITSAT (Rankooh and Ghassemi-Sani 2015) encodes a given planning problem into a SAT formula, which is given as the input to an off-the-shelf SAT solver. ITSAT performed well in quality score at IPC-2014, obtaining almost the same results of YAHSP3, but performed poorly in the number of solved problems, solving only 70% of the problems solved by YAHSP3.

Few temporal planners work directly with partial-order planning (POP). OPTIC (Benton, Coles, and Coles 2012), the last version of POPF2 (Coles et al. 2010), follows this approach and it is able to deal with time, numeric fluents, continuous effects, soft constraints and preferences. OPTIC obtains high quality plans, although its performance is usually below the sequential planners.

Our planner TFLAP is based on FLAP2 (Sapena, Torreño, and Onaindia 2016), a partial-order forward-chaining planner that follows the principles of POP. TFLAP works similarly to OPTIC, but it introduces two important differences:

- while OPTIC adds temporal constraints between actions to ensure that preconditions of the new actions are met in the frontier state, TFLAP does not commit to an action ordering if this is not required, just like traditional POCL planners do. This makes TFLAP be more flexible.
- TFLAP is able to incorporate new actions in any point of the current plan. OPTIC, however, only allows to add actions that are applicable in the frontier state so that the newly added actions do not threaten the preconditions of earlier actions.

These two differences lead to a more flexible temporal partial-order planner, although this improvement entails a higher computational effort to deal with the interactions among actions.

TFLAP's working scheme

TFLAP is a modular planner implemented in C++. Its main components can be seen in Figure 1. Initially, the domain and problem files of the planning task are sent to the parser module. Then, the information is preprocessed, grounded and translated to the SAS+ formalism (Bäckström and Nebel 1995). Although these modules can handle all the features of PDDL3.1, the planning component only supports PDDL2.1 and timed initial literals (TILs) for now.

TFLAP implements an A* search guided by an evaluation function. A search node is a partial-order plan and the start-

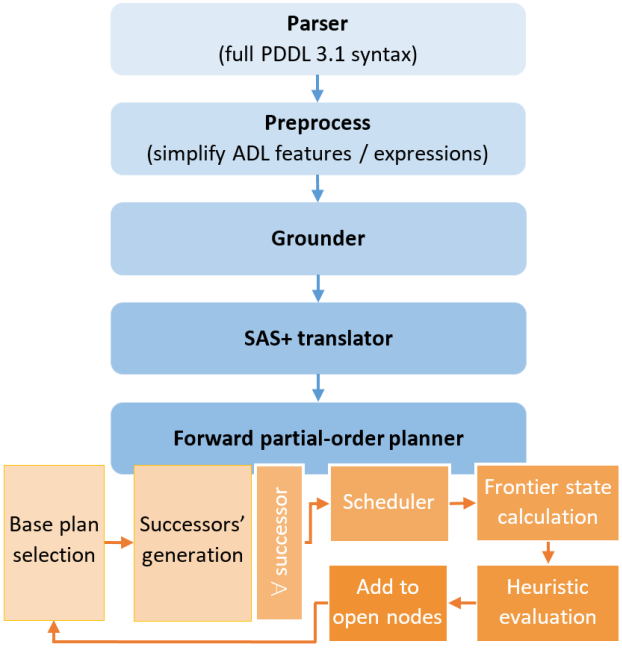

Figure 1: Working scheme of TFLAP.

ing node of the search tree is a plan with one fictitious action that represents the initial state. Additionally, a fictitious action is included for each TIL defined in the problem: the action begins at time 0, its duration is the programmed time, and the given literal is its only `at-end` effect. The planner applies the following six steps at each iteration of the search process until a solution plan is found:

1. It selects the best node, called *base plan*, from the set of open nodes according to the evaluation function.
2. It generates all possible successors of the base plan. TFLAP considers that a plan is a valid successor of the base plan if the following conditions are met:
   - The successor adds a new action to the base plan.
   - The conditions of the new action are supported with existing actions in the base plan by inserting the corresponding causal links.
   - All threats are solved through promotion or demotion by adding new ordering constraints: the result is a flaw-free successor plan.
3. TFLAP schedules the actions in each successor node, determining the start time and end time of each action in the plan. If no valid schedule is found, the successor does not meet the temporal constraints and it is rejected.
4. The schedule of a node is used to calculate the frontier state; i.e. the state resulting from executing the plan comprised in the node.
5. Successor nodes are evaluated using state-based heuristics, which are computed on their frontier states.
6. Finally, the successor plans are added to the set of open nodes.

The current version of TFLAP uses two different classical heuristics to evaluate nodes:

- h_{FF} (Hoffman and Nebel 2001), which builds a relaxed plan by ignoring the delete effects of the actions and returns its number of actions.
- h_{Land}, a heuristic that computes a landmark graph and estimates the goal distance through the number of landmarks that still need to be achieved from the state being evaluated (Hoffmann, Porteous, and Sebastia 2004; Sebastia, Onaindia, and Marzal 2006).

The set of open nodes are stored in two different queues, one per heuristic function. The nodes in the queues are sorted by $f(n) = g(n) + 2 * h(n)$, where $g(n)$ is the cost of node n in number of actions and $h(n)$ is the corresponding heuristic value of the node. TFLAP applies an alternation of heuristics: the most promising states are selected according to the currently used heuristic, completely ignoring the other heuristic estimate (Röger and Helmert 2010), and changing the heuristic when no improvement in two consecutive base plans is obtained.

TFLAP uses a sophisticated machinery to handle least commitment of durative actions and the scheduling of a node. This grants TFLAP a great flexibility to be able to deal with all types of concurrency problems (Cushing et al. 2007). Particularly, the scheduling process of a node is optimized by leveraging the topological ordering of the plan graph contained in the node. This process yields the frontier state and the makespan of the plan.

TFLAP does not make use of temporal information as search guidance and only uses classical heuristic functions, as commented above. This is a major constraint in domains with dead-ends that provokes TFLAP to encounter plateaus during search. Nonetheless, we can say that TFLAP is a simple and versatile temporal planner that exhibits on the whole a good performance.

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS+ planning. *Computational Intelligence* 11(4):625–655.

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. *International Conference on Automated Planning and Scheduling (ICAPS)* 2–10.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. *International Conference on Automated Planning and Scheduling (ICAPS)* 42–49.

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.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2009. Using the context-enhanced additive heuristic for temporal and numeric planning. *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*.

Hoffman, J., and Nebel, B. 2001. The FF planning system: Fast planning generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22:215–278.

Khouadjia, M. R.; Schoenauer, M.; Vidal, V.; Dréo, J.; and Savéant, P. 2013. Multi-objective AI planning: Evaluating dae YAHSP on a tunable benchmark. In *Evolutionary Multi-Criterion Optimization - 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*, 36–50.

Rankooh, M., and Ghassem-Sani, G. 2015. ITSAT: An efficient SAT-based temporal planner. *Journal of Artificial Intelligence Research (JAIR)* 53:541–632.

Röger, G., and Helmert, M. 2010. The more, the merrier: Combining heuristic estimators for satisficing planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, 246–249.

Sapena, O.; Torreño, A.; and Onaindia, E. 2016. Parallel heuristic search in forward partial-order planning. *The Knowledge Engineering Review* 31(5):417–428.

Sebastia, L.; Onaindia, E.; and Marzal, E. 2006. Decomposition of planning problems. *AI Communications* 19(1):49–81.

Vidal, V. 2011. YAHSP2: Keep it simple, stupid. *Proceedings of the 7th International Planning Competition (IPC-2011)* 83–90.

Vidal, V. 2014. YAHSP3 and YAHSP3-MT in the 8th international planning competition. *Proceedings of the 8th International Planning Competition (IPC-2014)* 64–65.

TemPoRal: Temporal Portfolio Algorithm

Isabel Cenamor
Universidad Carlos III de Madrid
icenamorg@gmail.com

Mauro Vallati
University of Huddersfield.
m.vallati@hud.ac.uk

Lukáš Chrpa
Czech Technical University in Prague &
Charles University in Prague
chrpaluk@fel.cvut.cz

Tómas de la Rosa
Universidad Carlos III de Madrid
trosa@inf.uc3m.es

Fernando Fernández
Universidad Carlos III de Madrid
ffernand@inf.uc3m.es

Abstract

This paper describes *TemPoRal*, a Temporal Planning Portfolio submitted to the satisficing temporal track of the International Planning Competition 2018. This portfolio performs a static equal time assignment of the available time to each of the portfolio components, which where empirically selected from the state-of-the art temporal planners.

Introduction

A planning portfolio is an automated planning system made of a set of individual planners for which one assigns a slot of the available time in order to solve a planning task. In the context of classical planning, the community has shown for several years that planning portfolios are very powerful in exploiting the complementary strength of different automated planners. For instance, portfolio-based approaches delivered outstanding performance in the 2014 edition of the International Planning Competition (IPC) (Vallati et al. 2015; Vallati, Chrpa, and McCluskey 2018). Even a simple equal-time assignment on a diverse set of planners can outperform most of the single planners (Fawcett et al. 2014; Cenamor, de la Rosa, and Fernández 2016). More elaborated approaches create per-instance configurable portfolio mainly based on some form of *Empirical Performance Models* (EPMs), for instance, *AllPaca* (Malitsky, Wang, and Karpas 2014) or *IBaCoP2* (Cenamor, de la Rosa, and Fernández 2016; 2014), the winner of the sequential satisficing track of IPC-2014.

Interestingly, no portfolio-based approach has participated in the temporal track of IPC so far. At the moment, the only work dealing with temporal planning portfolios appears in Cenamor's dissertation (2017). For the temporal track of this competition our proposal consists of a static portfolio with equal-time assignments. The creation of EPMs is less appealing for this case for several reasons. First, the number of temporal planners is fairly low compared to the available planners for the classical setting, therefore it does not make much sense to have a per-instance selection of a subset of planners when they are actually a manageable set in terms time slot assignment. In addition it is still not clear how the PDDL-level features related to temporal annotations will help to generalize across domains without falling in overfitting. Thus, we keep it simple until looking for deeper insights for this regard. In the next section we describe the details of the proposed static portfolio.

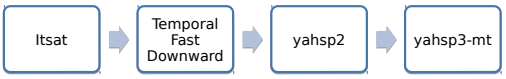

Figure 1: Planner's execution order in the temporal portfolio

TemPoRal Details and Components

For the creation of the portfolio we have considered all the planners that took part in any temporal track of past IPCs. In a first selection we remove planners showing issues during source code compilation, basically due to the use of old or deprecated libraries. Then, after running the candidate planners on a large set of instances, including all the available temporal benchmarks from previous competitions, we discarded the planners that were always dominated by another planner from the group. At the end, the portfolio comprises 4 planners, where each planner has 450 seconds to run in a specific order shown in Figure 1. If a component fails during its execution the remaining time equally re-assigned to the the planners that have not been executed yet.

The TemPoRal portfolio includes the followings component planners:

- *itsat* (Rankooh, Mahjoob, and Ghassem-Sani 2012) translates the problem into a sequence of SAT instances, corresponding to different time horizons.
- *Temporal Fast Downward (TFD)* (Eyerich, Mattmüller, and Röger 2012) is based on the Fast Downward planning system (Helmert 2006) and uses an adaptation of the context-enhanced additive heuristic to guide the search in the temporal state space induced by the given planning problem.
- *yahsp2* and *yahsp2-mt* (Vidal 2011) compute look-ahead plans from delete-relaxed plans and use them in the state-space heuristic search.
- *yahsp3* and *yahsp3-mt* (Vidal 2014) are the latest version of the Yahsp planner, which took part in IPC 2014.

Acknowledgements

This portfolio is based on previous temporal systems. Thus, we would like to acknowledge and thank the authors of the individual planners for their contribution and hard work.

References

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2014. IBaCoP and IBaCoP2 planner. *Proceedings of the 8th International Planning Competition (IPC-2014)*.

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2016. The IBaCoP planning system: Instance-based configured portfolios. *J. Artif. Intell. Res.* 56:657–691.

Cenamor, I. 2017. *Creating Planning Portfolios with Predictive Models*. Ph.D. Dissertation, Departamento de Informatica, Universidad Carlos III de Madrid.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2012. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Towards Service Robots for Everyday Environments*. Springer. 49–64.

Fawcett, C.; Vallati, M.; Hutter, F.; Hoffmann, J.; Hoos, H. H.; and Leyton-Brown, K. 2014. Improved features for runtime prediction of domain-independent planners. In Chien, S. A.; Do, M. B.; Fern, A.; and Ruml, W., eds., *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26(1):191–246.

Malitsky, Y.; Wang, D.; and Karpas, E. 2014. The allpaca planner: All planners automatic choice algorithm. *International Planning Competition (IPC)* 71–73.

Rankooh, M. F.; Mahjoob, A.; and Ghassem-Sani, G. 2012. Using satisfiability for non-optimal temporal planning. In *Logics in Artificial Intelligence*. Springer. 176–188.

Vallati, M.; Chrpa, L.; Grześ, M.; McCluskey, T. L.; Roberts, M.; Sanner, S.; et al. 2015. The 2014 international planning competition: Progress and trends. *Ai Magazine* 36(3):90–98.

Vallati, M.; Chrpa, L.; and McCluskey, T. L. 2018. What you always wanted to know about the deterministic part of the international planning competition (IPC) 2014 (but were too afraid to ask). *Knowledge Eng. Review* 33:e3.

Vidal, V. 2011. Yahsp2: Keep it simple, stupid. *The 2011 International Planning Competition* 83–90.

Vidal, V. 2014. YAHSP3 and YAHSP3-MT in the 8th international planning competition. In *Proceedings of the 8th International Planning Competition*.

Temporal-Numeric Planning with Infinite Domain Action Parameters

Emre Savaş and **Stefan Edelkamp** and **Maria Fox** and **Derek Long** and **Daniele Magazzeni**
Department of Informatics, King's College London, London, UK, WC2B 4BG
firstname.lastname@kcl.ac.uk

Abstract

A limitation of task planning as a search problem is that action parameters are restricted to take their values from a finite collection of objects defined in a problem instance. In real-world, there are parameters that have freedom of taking their values from infinite domains, which we call *control parameters*. Planning with control parameters are deemed to be more than just a modelling choice; but it is a modelling requirement in real-world applications. POPCORN is a partially-ordered forward state space search planner that can reason with multiple *typed* control parameters declared in action schemes. In this paper, we briefly introduce the planning with control parameters paradigm.

1 Introduction

Over the last decade, significant progress has been made in task planning to explore temporal and numerical features of the real world, including the work of (Do and Kambhampati 2003; Coles et al. 2012; Eyerich, Mattmüller, and Röger 2012; Benton, Coles, and Coles 2012; Piotrowski et al. 2016). Interaction between time and metric fluents and temporal coordination problems have been well-addressed by innovative approaches in planning and scheduling. A standardised language, PDDL, has enabled modelling the conceptual models and benchmarking the work of numerous researchers since it was released.

Although PDDL is an expressive modelling language, a significant limitation is imposed on the structure of actions: the parameters of actions are restricted to values from finite (in fact, explicitly enumerated) domains. There is one exception to this, introduced in PDDL2.1 (Fox and Long 2003), which is that durative actions may have durations that are chosen (possibly subject to explicit constraints in the action models) by the planner. A motivation for this limitation is that it ensures that the set of grounded actions is finite and, ignoring duration, the branching factor of action choices at a state is therefore finite. Although the duration parameter can make this choice infinite, very few planners support this possibility, but restrict themselves to durative actions with fixed durations.

In this paper, we briefly introduce planning with control parameters paradigm in a temporal setting. We provide brief details of the effects of control parameters in the forwards search space. We conclude the paper with a set of real-world examples where modelling using control parameters is more than just a modelling choice. This work is described in detail in work of Savaş et al. (2016).

2 A Motivating Case

We start with a very simple example motivating the use planning with control parameters. Suppose we are given two jugs and we want to pour *a bit* of water from the one into the other as illustrated in Figure 1. We know that the water level in the one jug will decrease by the amount the other one will increase, and we also observe obvious side constraints on the pouring actions. However, during modelling we do not yet know the numerical amount of water being poured, as this will be dependent on the partial plan executed so far and has to be optimised for plan improvement.

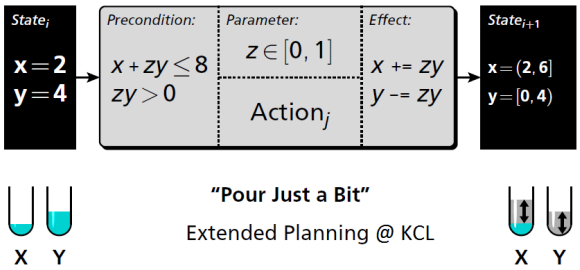

Figure 1: Water pouring action with control parameters.

3 Planning with Control Parameters

Planning with control parameters is a generalisation of the duration parameter of durative actions to other types, allowing actions to take multiple duration-independent infinite domain parameters. We define the temporal and numeric planning problem with control parameters as follows:

Definition 1 *A temporal and numeric planning problem with control parameters is a tuple* $\langle F, \vec{v}, I, A, G, M \rangle$*, where:*

- F *is a finite set of grounded literals,*
- $\vec{v}$ *is a finite set of numeric variables,*

- *I is the initial state,*
- *A is a set of actions. Each action, $a \in A$, is a tuple:*
$$a = \langle dur,\ cont,\ pre^n_x,\ pre_x,\ eff^n_y,\ eff_y \rangle$$
  - *dur is a set of duration constraints of action a, where each is expressible in the form of $\langle g(\vec{v}, \vec{d^a}, \texttt{?duration}), comp, c \rangle$, where $c \in \mathbb{R}$.*
  - *pre_x and pre^n_x are the propositional and numeric conditions, respectively, that must hold at a state in which the action starts ($\vdash$), at a state in which the action ends ($\dashv$), or between the start and the end of an action a ($\leftrightarrow$): $x \in \{\vdash, \leftrightarrow, \dashv\}$. The numeric conditions are in the form:*
$$pre^n_x = \langle f(\vec{v}, \vec{d^a}), comp, c \rangle$$
  - *eff_y are the propositional start (or end) effects of an action a that are to be added to (or deleted from) the world state. eff^n_y are the numeric effects that acts on variable $v \in \vec{v}$. We denote the decrease effects by eff^-_y and the increase effects by eff^+_y; where $y \in \{\vdash, \dashv\}$. The numeric effect eff^n_y is expressible in the form:*
$$eff^n_y = \langle v, op, g(\vec{v}, \vec{d^a}, \texttt{?duration}) \rangle,$$
- *G is the goal described by a set of propositions, $p(G) \subseteq F$, and a set of numeric conditions, $v(G)$, over the state variables,*
- *M is the metric objective function.*

A solution to such a problem is an extended version of the plans described in (Coles et al. 2012):

Definition 2 (Controlled Plan (π)) *is a list of timestamped actions with duration and specified values for control parameters of actions in π, $\{\langle a, t, dur, \vec{d^a} \rangle, \dots\}$, where $t \in \mathbb{R}$ is a timestamp (the time at which the action a is applied), $dur \in \mathbb{R}$ is the duration and $\vec{d^a} \in \mathbb{R}^{m^a}$ is the vector of values of the control parameters of a. The duration constraints of each action in π are satisfied, and all pre^n_x of actions in π hold in states in which the actions are applied. The trajectory of states generated by application of the actions yields a final state that satisfies the goal.*

POPCORN is a forward state space heuristic search planner. It builds a *constraint space* search alongside the forwards search as follows. It collects temporal-numeric constraints that appears in the conditions and effects of actions chosen during forwards search. It uses Simple Temporal Network (STN) to check temporal consistency at a finitely branching state. However, the existence of control parameters creates infinite branching factor in forward state space search. To avoid infinite branching decision, POPCORN planner uses a Linear Programming (LP) solver to check the temporal/numeric consistency at a state. It leaves infinite branching choice to the constraint space, relying on LP-solving. LP solver helps to prune states with infeasible solutions. Figure 2 illustrates the resulting search space.

4 Conclusion

We introduced a way of modelling infinite domain action parameters in expressive temporal-numeric PDDL language. The approach accumulates constraints during planning and uses them to determine the values of these parameters and to prune inconsistent branches from the search. The use of control parameters expands the real-world applicability of task planning.

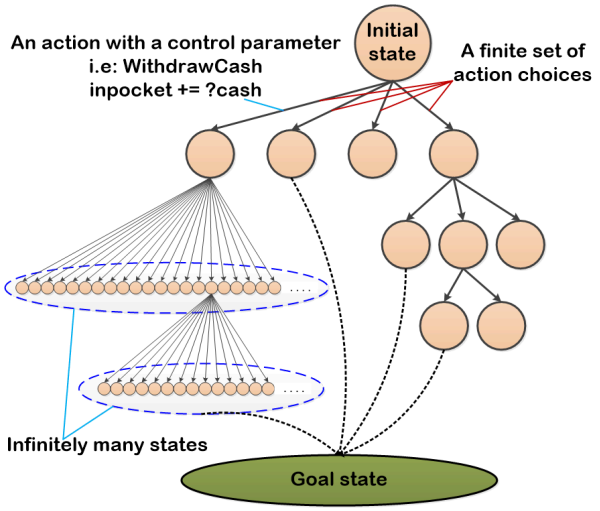

Figure 2: Schematic representation of the search space where there is a control parameter effect. The nodes represent the state reached, and the edges represent the action applied to reach the next state. The graphs in black boxes represent the LP constraint space, which is used to avoid complex branching choice.

References

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling, ICAPS*.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2012. COLIN: Planning with Continuous Linear Numeric Change. *Journal of Artificial Intelligence Research (JAIR)* 1–96.

Do, M. B., and Kambhampati, S. 2003. Sapa: A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research* 20:155–194.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2012. Using the context-enhanced additive heuristic for temporal and numeric planning. *Towards Service Robots for Everyday Environments* 76:49–64.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research (JAIR)* 20:61–124.

Piotrowski, W.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic Planning for PDDL+ Domains. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press.

Savaş, E.; Fox, M.; Long, D.; and Magazzeni, D. 2016. Planning Using Actions with Control Parameters. In *Proceedings of the Twenty-Second European Conference on Artificial Intelligence (ECAI 2016)*, 1185–1193. IOS Press.