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-

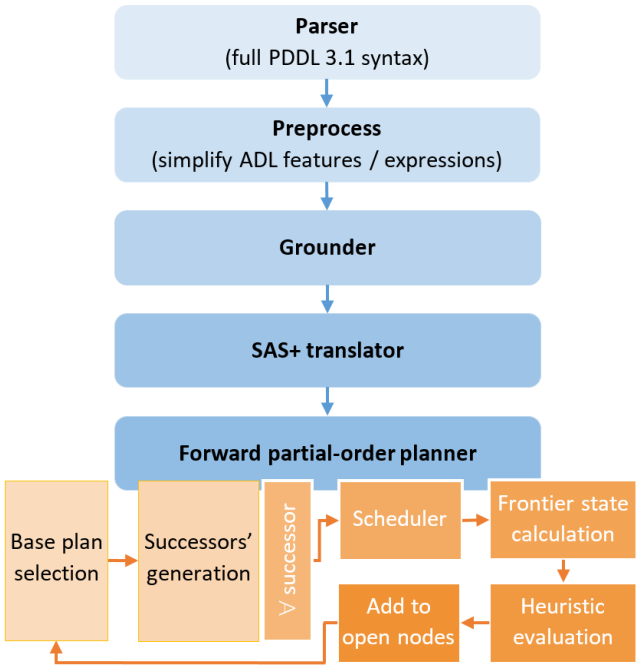

Figure 1: Working scheme of TFLAP.

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.