TFD: A Numeric Temporal Extension to Fast Downward

Gabriele Röger and **Patrick Eyerich** and **Robert Mattmüller**
University of Freiburg, Germany
{roeger,eyerich,mattmuel}@informatik.uni-freiburg.de

Abstract

Temporal Fast Downward (TFD) is a variant of the well-known propositional Fast Downward planning system and supports the temporal and numeric features of PDDL. Like the propositional system, TFD is based on heuristic search but makes use of a different heuristic that is a generalization of Fast Downward's original causal graph heuristic.

Introduction

Fast Downward (Helmert 2006) is a well-known planning system that won the "classical" (i.e., propositional, non-optimizing) part of the 4th International Planning Competition in 2004. The system is based on heuristic state space search using the causal graph heuristic (Helmert 2004) that exploits causal dependencies of the planning domains.

A limitation of Fast Downward is that it only deals with the propositional fragment of PDDL. For this reason, we have extended the system to cover the numeric and temporal features of PDDL resulting in a new planning system called *Temporal Fast Downward* (TFD).

Temporal Fast Downward copes with the numeric features by partially decoupling them from the operators (by introducing two new types of axioms) and handling them separately. The temporal aspects are primarily handled by an adaption of the search space to concurrent actions.

Besides the adjustments that were necessary to cope with the wider PDDL fragment, the most important difference is that TFD does no longer use the causal graph heuristic. Instead, it makes use of the context-enhanced additive heuristic (CEA) recently proposed by several Geffner (2007). This heuristic is a generalization of both the causal graph heuristic and the additive heuristic (Bonet, Loerincs, & Geffner 1997).

Fast Downward works in three steps: First it translates the input to a planning formalism with multi-valued state variables. In a second step, the so-called *knowledge compilation* component generates several data structures that are used during search. The third component of the system implements the actual search. Temporal Fast Downward retains this structure of Fast Downward. In the following, we address the several components (translation, knowledge compilation, and search) in more detail.

Translation

The *translator* has the following responsibilities:

- Compiling away (most) ADL features and object fluents.
- Moving numeric features from the operators to axioms.
- Grounding the operators and axioms.
- Converting the resulting (binary) representation to one with multi-valued state variables.

Most of these steps are also done by the translator of the propositional Fast Downward system and were adapted to the temporal formalism in a straightforward manner. Probably the only peculiar aspects are the treatment of object fluents and the handling of numeric features.

One of the most remarkable features of the original translator is that it transforms the binary representation of a task to one with multi-valued state variables. The main change from PDDL 3.0 to version 3.1 is that one can use object fluents, which are quite similar to multi-valued state variables. Thus, it might seem silly that the first thing we do is to compile away these objects fluents. The main reason for this is that object fluents can be nested which cannot directly be carried over to Fast Downward's multi-valued state variables. Experiments have shown that usually the conversion in the last step of the translation introduces a multi-valued state variable for each unnested object fluent but often also finds additional propositional variables that can be combined to multi-valued variables.

The temporal aspects (like conditions and effects being annotated with time specifiers) are handed over to the knowledge compilation component in a straightforward way.

Besides the temporal aspects, the major difference from the original propositional translator is that we have to handle numeric features. As mentioned in the introduction we introduced two new types of axioms for this purpose: *numeric axioms* map numeric expressions to *derived numeric variables* and *comparison axioms* map a comparison of two numeric variables to a binary *comparison variable*. After these substitutions numeric expressions occur in operators only as derived numeric variables and comparisons as comparison variables. All this may seem overly complicated but the approach has the advantage that it somewhat uncouples the numeric features from the operators and makes it possible to handle them separately.

Knowledge Compilation

The purpose of the knowledge compilation component is to build some data structures that are used by the context-enhanced additive heuristic and that facilitate efficient state expansion during search. In detail, the responsibilities of this preprocessing step are:

- Computing the *causal graph* of the planning task which encodes dependencies between different state variables.
- Computing the *domain transition graph* for each state variable that encodes how operators affect the variable.
- Computing the *successor generator*, a data structure that supports efficiently computing an over-approximation of the set of applicable operators for a state.

The *causal graph* encodes dependencies between different state variables. Conceptually, the computation of the causal graph in TFD remains the same as in Fast Downward: The causal graph contains an arc from a source variable to a target variable if changes in the value of the target variable can depend on the value of the source variable. However, in contrast to Fast Downward we do not require the causal graph to be acyclic, so there is no need for a relaxation of the planning task at this level.

For each state variable a *domain transition graph* (DTG) is computed that encodes under which circumstances the variable can change its value. While it is sufficient to use only one kind of DTGs in Fast Downward (since there is only one variable type), we have to distinguish different forms of DTGs for the different kinds of variables in a temporal multi-valued planning task. While the DTGs for comparison variables only differ from the original ones in minor details, we cannot use the same concept for numeric variables since their domain range is infinite. Actually, a DTG for a numeric variable not even *is* a graph. Instead, it contains the information necessary for creating a graph structure online during the calculation of the heuristic value.

Derived variables are handled using a straightforward adaption of the *axiom evaluator* from Fast Downward to numeric and comparison axioms.

Unlike the causal graph heuristic, the context-enhanced additive heuristic no longer requires an explicit representation of the causal graph of the problem or the DTGs of the state variables. Nevertheless, the implementation can be optimized at several points by using the information given by these data structures, so we decided to compute them anyway.

Search

The search component is responsible for the systematic exploration of the state space of the given problem. We use a greedy best-first search approach enhanced with deferred heuristic evaluation. The main differences between Temporal Fast Downward and Fast Downward with respect to search are:

1. **Search space:** Besides the values of the state variables, the *time-stamped states* in the search space contain a real-valued time stamp as well as information about scheduled effects and conditions of currently executed actions. This representation is very similar to the one chosen by Do and and Kambhampati in the SAPA system (Do & Kambhampati 2003). A transition from one time-stamped state to another is accomplished by either (a) adding an applicable action starting at the current time point, applying its start effects and scheduling its end effects as well as its over-all and end conditions, or (b) letting time pass until the next scheduled happening and adapting the new time-stamped state accordingly, i.e., applying effects scheduled for the new time point and deleting expired conditions.
2. **Heuristic:** The search is guided by a variant of the context-enhanced additive heuristic adapted to handle numeric variables. The original CEA heuristic is a generalization of both the additive heuristic and the causal graph heuristic used in Fast Downward and does not require the causal graph to be acyclic. The heuristic does not work on time-stamped states but on a relaxed version of them. For example, the information about *when* and *under which conditions* scheduled effects will occur is neglected. Similarly, scheduled conditions are not considered at all.

   The heuristic computation uses search techniques on DTGs based on Dijkstra's algorithm and takes into account the current context of the relevant state variables. The basic idea of the extension to numeric state variables is to construct the graph on which Dijkstra's algorithm is performed on-the-fly during the heuristic computation.

References

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A robust and fast action selection mechanism for planning. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*.

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

Geffner, H. 2007. The causal graph heuristic is the additive heuristic plus context. In *Proc. 2007 ICAPS Workshop on Heuristics for Domain-Independent Planning*.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In Zilberstein, S.; Koehler, J.; and Koenig, S., eds., *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 161–170. AAAI Press.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.