Fast (Diagonally) Downward

Malte Helmert
Institut für Informatik, Albert-Ludwigs-Universität Freiburg
Georges-Köhler-Allee, Gebäude 052, 79110 Freiburg, Germany
helmert@informatik.uni-freiburg.de

Abstract

Fast Downward is a propositional planning system based on heuristic search. Compared to other heuristic planners such as FF or HSP, it has two distinguishing features: First, it is tailored towards planning tasks with *non-binary* (but finite domain) state variables. Second, it exploits the *causal dependency* between state variables to solve relaxed planning tasks in a hierarchical fashion.

Fast Downward won the propositional satisficing track of the 4th International Planning Competition (IPC4). At the 5th International Planning Competition (IPC5), a (mostly) unchanged version of the planner was entered to provide a reference point for comparing to the earlier state of the art.

Introduction

Fast Downward is a planning system based on heuristic state space search, in the spirit of HSP or FF (Bonet & Geffner 2001; Hoffmann & Nebel 2001). It makes use of the *causal graph* (or CG) heuristic, introduced in an ICAPS 2004 paper (Helmert 2004). Fast Downward itself is described in much detail in a JAIR article (Helmert 2006). For this reason, we only provide a very brief overview in the following.

Structure of the planner

Fast Downward consists of three separate programs:

1. the *translator* (written in Python),
2. the *knowledge compilation* module (written in C++), and
3. the *search engine* (also written in C++).

To solve a planning task, the three programs are called in sequence; they communicate via text files.

Translator

The purpose of the *translator* is to transform the planner input, specified in the propositional fragment of PDDL (including ADL features and derived predicates, but not the preferences and constraints introduced for IPC5), into a multi-valued state representation similar to the SAS^+ formalism (Bäckström & Nebel 1995).

The main components of the translator are an efficient grounding algorithm for instantiating schematic operators and axioms and an invariant synthesis algorithm for determining groups of mutually exclusive facts. Such fact groups are consequently replaced by a single multi-valued state variable encoding *which* fact (if any) from the group is satisfied in a given world state, rather than encoding the truth of each fact in a separate state variable.

The translator can be used independently from the rest of the planner, and has already proved to be a useful component in planning systems not related to Fast Downward (van den Briel, Vossen, & Kambhampati 2005).

Knowledge Compilation

Using the multi-valued task representation generated by the translator, the *knowledge compilation* module is responsible for building some data structures which play a central role in Fast Downward's search engine.

First and foremost, it determines the *causal graph* of the input task, whose purpose is to encode the information which state variables are relevant to changing which other state variables of the task. Together with *domain transition graphs* for each state variable, which encode the ways in which a given state variable may change its value through operator applications, it forms the basis for the recursive computation of the CG heuristic.

The knowledge compilation module also generates *successor generators* and *axiom evaluators*, data structures for efficiently determining the set of applicable actions in a given state of the planning task and for evaluating the values of derived state variables.

Search Engine

Using these data structures, the *search engine* attempts to find a plan using greedy best-first search with some enhancements such as the use of *preferred operators* (similar to helpful actions in FF) and *deferred heuristic evaluation*, which mitigates the impact of large branching factors in planning tasks with accurate heuristic estimates.

The search engine can also be configured to use several heuristic estimators (namely, the CG heuristic and the FF heuristic) in tandem within an algorithm called *multi-heuristic best-first search*. This search algorithm attempts to exploit strengths of the utilized heuristics in different parts of the search space in an orthogonal way. The planner configuration using multi-heuristic best-first search is called *Fast Diagonally Downward*, because it combines the "downward" thrust of the CG heuristic with the "forward" thrust of the FF heuristic.

Fast Downward also includes a third search algorithm called *focused iterative-broadening search*, but as of this writing, it is not clear whether or not this algorithm will be used for the IPC5 benchmark set.

New Developments

Since IPC4, apart from some bug fixes, we made only one modification to Fast Downward.

In some planning domains, it is clear from the multi-valued task description that some state variables can change their value in essentially arbitrary ways without further conditions on other state variables. For example, state variables which encode vehicle locations in transportation domains such as LOGISTICS or DEPOTS never have causal dependencies on other state variables in the task (i. e., they are source nodes in the causal graph) and can never assume a value from which the initial value cannot be restored anymore (i. e., their domain transition graphs are strongly connected).

In tasks where such state variables are present, Fast Downward can now apply *safe abstraction* to remove the state variables from the planning task altogether and plan in the resulting abstract planning task. After planning, the necessary actions to convert the generated abstract plan to the concrete level can be inserted in a straight-forward manner.

Safe abstractions can substantially speed up planning; indeed, in very simple cases like the LOGISTICS domain, repeated application of safe abstraction can completely solve the planning task, without requiring any search. However, there is often a price to pay in plan quality.

At IPC5, we have run Fast Downward both in a version that does use safe abstractions where possible and in a version which never uses them. The latter version of the planner is identical to Fast Downward at IPC4 (modulo bug fixes) and thus serves as a useful reference point.

References

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

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1):5–33.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proc. ICAPS 2004*, 161–170.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research*. Accepted for publication.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

van den Briel, M.; Vossen, T.; and Kambhampati, S. 2005. Reviving integer programming approaches for AI planning: A branch-and-cut framework. In *Proc. ICAPS 2005*, 310–319.