The FF Planning System

Jorge A. Baier
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada

CSC2542 – Topics in Knowledge Representation and Reasoning

The Fast Forward (FF) Planning System

- Was proposed by Hoffmann & Nebel (2001).
- Was the winner of the 2000 planning competition.
- Its novel elements are the following:
  - Heuristic based on relaxed plans.
  - Enforced Hill Climbing Used as the Search Strategy.
  - Its core ideas have had substantial impact.

The Relaxed Plan Heuristic: Basic Definitions

Definition (STRIPS planning problem)
Let $P = (\text{Init}, \text{Ops}, \text{Goal})$ be a STRIPS planning problem where:
- $\text{Init}$ is the initial state.
- $\text{Goal}$ is the goal condition.
- Each $o \in \text{Ops}$ of the form $o = (\text{prec}(o), \text{add}(o), \text{del}(o))$

Definition (Delete-Relaxation)
The delete relaxation of $P$, denoted $P^+$, is a instance just like $P$ but in which operators in $\text{Ops}$ have an empty delete list.

Definition (Relaxed Plan)
A relaxed plan for $P$ is any plan for $P^+$.

Computing a Relaxed Plan

For a planning state $s$:
$h_{FF}(s) = \text{“number of actions in a relaxed plan from } s\text{”}$

The relaxed plan computed by FF:
- Is obtained using a version of Graphplan on $P^+$.
- Is not a shortest relaxed plan (since this is already NP-hard).

Computing a Relaxed Plan: Algorithm

Extraction algorithm (Hoffmann & Nebel, 2001)

1: function EXTRACTPLAN(plan graph $P_0P_1\cdots P_n$, goal $G$)
2:    for $i = n \ldots 1$ do
3:        $G_i \leftarrow$ goals reached at level $i$
4:    end for
5:    for $i = n \ldots 1$ do
6:        for all $g \in G_i$ not marked TRUE at time $i$ do
7:            Find min-cost $a \in A_{i-1}$ such that $g \in add(A_{i-1})$
8:            $RP_{i-1} \leftarrow RP_{i-1} \cup \{a\}$
9:        end for
10:        for all $f \in prec(a)$ do
11:            $G_{\text{layer}}(f) = G_{\text{layer}}(f) \cup \{f\}$
12:        end for
13:        for all $f \in add(a)$ do
14:            mark $f$ as TRUE at times $i-1$ and $i$.
15:        end for
16:    end for
17:    return $RP$
18: end function

Highlights of the relaxed plan extraction algorithm:
- Plan is extracted by regressing the goals (i.e. backwards)
- Iterates from the highest to the lowest level.
- Earliest achievers are always preferred.
“Min-Cost” Actions

The “min-cost” action referred to in line 7 is the one that minimizes the following function:

\[ \text{Cost}(a) = \sum_{p \in \text{Prec}(a)} \text{level}(p). \]

where \( \text{level}(p) \) is the first layer at which \( p \) appears, and \( \text{Prec}(a) \) are the preconditions of \( a \).

Helpful Actions

Helpful actions are essential for FF’s performance. Helpful actions are those that appear at the first level of the relaxed plan.

**Definition (Helpful action)**

An action \( a \) of a relaxed plan from \( s \) is helpful iff it is a member of \( \text{RP}_0 \).

Note that helpful actions are a subset of the actions executable in \( s \).

Enforced Hill Climbing

**Enforced Hill Climbing (EHC) (Hoffmann & Nebel, 2001)**

1: function EHC(initial state \( I \), goal \( G \))
2: plan ← EMPTY
3: \( s \leftarrow I \)
4: while \( h(s) \neq 0 \) do
5: from \( s \), search for \( s' \) such that \( h(s') < h(s) \).
6: if no such state is found then
7: return fail ⊲ discard and continue the iteration
8: end if
9: if \( h(s') < h(s) \) then
10: \( s' \leftarrow t \)
11: break ⊲ better state found, exit loop
12: end if
13: \( \text{push}(\text{queue}, \{ \text{helpful successors of } s \}) \)
14: \( \text{closed} \leftarrow \text{closed} \cup \{ t \} \)
15: end while

FF’s search strategy

EHC is an incomplete search algorithm and thus prone to failure. If EHC fails, FF falls back into best-first search (A search), in which the evaluation function for a state is:

\[ f(s) = h_{FF}(s) \]

Note that this search is complete but greedy since the length of the plan is not considered.

Now let’s see how FF works in practice!

References I