FDP: Filtering and Decomposition for Planning

Stéphane Grandcolas et Cyril Pain-Barre
LSIS – UMR CNRS 6168
Domaine Universitaire de Saint-Jérôme
Avenue Escadrille Normandie-Niemen
13397 MARSEILLE CEDEX 20 France
{stephane.grandcolas,cyril.pain-barre}@lsis.org

Overview

FDP is a planning system based on the paradigm of planning as constraint satisfaction, that searches for optimal sequential plans. The input langage is PDDL with typing and equality. FDP works directly on a structure related to Graphplan's planning graph: given a fixed bound on the length of the plan, the graph is incrementally built. Each time the graph is extended, a search for a sequential plan is made.

FDP does not use any external solver. The reason is that using an up-to-date CSP solver allows to take benefits from recent advances in the CSP field, but has also the disadvantage that the resulting system can not take into account the specificities of planning nor the structure of the problem. Hence, as the DPPLAN system (Baioletti, Marcugini, & Milani 2000), FDP integrates consistency rules and filtering and decomposition mechanisms suitable for planning.

A structure that represents the planning problem is incrementally extended until a solution is found or a fixed bound of the number of steps is reached. The current implementation extends the structure with one step more. Each time a depth-first search is performed, based on problem decomposition with actions sets partitioning. Nevertheless, it is basically *Depth-First Iterative Deepening* (Korf 1985) (or IDA^* with admissible heuristic of constant cost 1).

FDP does not detect unsolvability of problems, as many other similar approaches (Rintanen 1998; Baioletti, Marcugini, & Milani 2000; Lopez & Bacchus 2003). Then, it must be given a fixed bound of plan length in order to stop on unsolvable instances of problems. This weakness of the algorithm will be adressed in future work.

The search procedure is complete. Then if a solution is found, it is minimal in terms of plan length. On the other hand, the current search procedure of FDP requires that any solution must contain only one single action per step. Hence, solutions returned by FDP are optimal in terms of the number of actions.

Problem representation

FDP works on a structure that resembles the well-known GRAPHPLAN planning graph (Blum & Furst 1995). It is a leveled graph that alternates *propositions levels* and *actions levels*. The i-th propositions level represents the validity of the propositions at step i. The i-th actions level represents the possible values for the action that is applied at step i. Since FDP searches for optimal sequential plans, FDP structures do not contain no-ops actions.

Consistent FDP-structures

FDP makes use of consistency rules to remove from FDP-structures some values of proposition variables or actions that cannot occur in any valid plan. For example an action whose one precondition is not valid should not be considered, and then can be removed without loss of completeness. The search procedure maintains the consistency of the FDP-structure, so as to discard as soon as possible invalid litterals or actions. A consistent structure in which each action level contains a single action and such that the first proposition level corresponds to the initial state of the planning problem and the last level contains the goals, represents a solution plan.

FDP consistency rules are the following. A litteral l at level i is inconsistent (cannot be *true*) if one of the following situations hold:

1. **(forward persistency)**
   l is not true at level $i-1$ and no possible action at level $i-1$ has l as effect,
2. **(all actions delete)**
   any possible action at level $i-1$ deletes l,
3. **(backward persistency)**
   l is not true at level $i+1$ and no possible action at level i deletes l,
4. **(opposite always required)**
   any possible action at level i has $\neg l$ as precondition.

A possible action a at step i is inconsistent (cannot occurs) if one of the following situations hold:

1. **(falsified precondition)**
   a precondition of a is inconsistent at level i,
2. **(falsified effects)**
   an effect of a is inconsistent at level $i+1$,
3. **(effect required)**
   there exists a litteral l such that l is inconsistent at level i, $\neg l$ is inconsistent at level $i+1$, and l is not an effect of a.

Maintaining consistency

Making a FDP-structure consistent consists in removing inconsistent values and actions until none exists or a domain becomes empty. The mechanism is similar to arc consistency enforcing procedures in the domain of constraint satisfaction (Dechter 2003; Mackworth 1977). One major aspect of the procedure is that the removals are propagated forward and backward through the FDP-structure. Propagation stops with failure if a domain becomes empty and the procedure returns FALSE. In the other case the procedure stops with the consistent FDP-structure S.

Search procedure

To find an optimal plan, FDP starts with a one step FDP-structure, and extends it until a plan is found or a given fixed bound is reached. Each time the FDP-structure is extended, a depth-first search is performed. This ensures the optimality of the solution plan if one exists. FDP employs a *divide and conquer* approach to search for a plan of a given length: the structure is decomposed into smaller substructures and the procedure searches recursively each of them. The substructures are filtered so as to detect failures as soon as possible.

The decomposition mechanism currently performed is *splitting action sets*. It consists in partitionning the set of actions at a given step i so as to put together actions which have common deletions: The procedure searches for the undefined proposition variable p at step $i+1$ for which the number of actions that delete it and the number of actions that do not are the closest. The FDP-structure is then decomposed into two substructures, one containing the actions at step i which delete p, the other containing the remaining actions at step i. The two substructures are then filtered.

When searching for a plan of length k, FDP uses a FDP-structure S:Initially each action set of S is set to A and each proposition variable is undefined. Then, the values which are not in the initial state and the opposites of the goals are removed and a preliminary filtering is performed on S. If S is inconsistent then the search stops with failure, there are no plans of length k. In the other case, FDP starts searching with the consistent structure S, which is decomposed into two substructures according to the splitting of an actions set. Nevertheless, the search procedure remains a depth first iterative deepening search, since it always chooses the first non singleton actions set for splitting, starting from the initial state. To produce each of the two substructures by actions set splitting, FDP just removes from the actions set the actions belonging to other actions subset. Then, each resulting substructure is filtered so as to remove inconsistent values and actions. If it is consistent, the search is recursively performed. These transformations continue until the (sub)structure becomes inconsistent or a valid plan.

Improving performance

FDP uses several techniques to avoid search efforts and then improve performance. They are: recording nogoods, evaluation of minimal plan length, avoidance of redundant actions sequences, elimination of literals and actions that are not relevant. These techniques are briefly discussed below.

Nogoods recording. Whenever the system produces a totaly defined state at a level i such that the recursive search from that state returns failure, this state and its distance to the golas are recorded as a nogood. Later, if the same state is reached but its distance to the goal step is less than or equal to the memorized distance, then there is no need to pursue the search. Recording nogoods improves drastically the performances of the search.

Minimal plan length. Anytime a propositional level F_i is completely instantiated, FDP performs a greedy evaluation of the length of a plan to achieve the goals from that state. It consists in choosing at each of the following steps the action which adds the most unsatisfied goals. In the best case these actions will constitute a valid plan. This heuristic is admissible: The number of steps needed to achieve the goals with this evaluation process cannot be greater than the number of steps actually needed in any valid plan. If at step k some goals are not achieved by the selected actions, then the search from the current state is aborted.

Redundant actions sequences. Since FDP searches sequential plans, it can generate equivalent permutations of "independent" actions and perform as many redundant processings. To avoid these useless processings, FDP discards the sequences of independent actions that do not verify an arbitrary total order on the actions denoted $\prec$.

Definition 1 (Ordered 2-Sequences) *The actions a_1 and a_2 are* independent *if the following situations hold:*

1. *no precondition of a_1 is an effect of a_2 and no precondition of a_2 is an effect of a_1*[1]*,*
2. *no deletion of a_2 is a precondition of a_1 and no deletion of a_1 is a precondition of a_2.*

The sequence (a_1, a_2) is an ordered 2-sequence *if either a_1 and a_2 are independent and $a_1 \prec a_2$, or a_1 and a_2 are not independent.*

FDP discards unordered 2-sequences. Besides, it also discards sequences whose actions have exactly opposite effects, as such sequences are useless in a plan.

To avoid sequences that do not verify the order, the following rules are added to the definition of inconsistent actions:

4. **(no backward ordered 2-sequence)**
   a is inconsistent at level i if there exists no action a' at level $i-1$ such that (a', a) is an ordered 2-sequence,
5. **(no forward ordered 2-sequence)**
   a is inconsistent at level i if there exists no action a' at level $i+1$ such that (a, a') is an ordered 2-sequence.

Relevant literals and actions. FDP searches optimal sequential plans. Then actions which do not help effectively to achieve the goals are useless and should not be considered. Basically relevant actions are the ones which add goals at the

[1]If a_1 requires a fact which is added by a_2, it is possible in some situations that the sequence (a_2, a_1) must be authorized. Then a_1 and a_2 should not be considered as independent.

last level. This property can be propagated backwards iteratively introducing the notion of *relevant literals and actions* at some steps:

1. a literal l is relevant at level i if there exists an action a at level i such that l is a precondition of a and a is relevant at level i,
2. an action a is relevant at level i if one of its effects is relevant at level $i + 1$.

At any moment during the search, actions that are not relevant at a given level can be removed from this step as it could not serve in any minimal solution.

Mutually exclusive propositions and actions FDP does not implement any specific processing for mutual exclusion relations, in particular those handled in GRAPHPLAN. Indeed, they are useless since FDP produces only sequential plans, and the effects of mutual exclusions of propositions are redundant with FDP inconsistency rules.

Conclusion and perspectives

Compared to other optimal sequential planners FDP seems to be competitive. Its advantage is its regularity: maintaining consistency, memorizing invalid states, and discarding redundant sequences, in addition with a fast and light search procedure, let FDP quickly detect deadends.

Its consistency rules and its decomposition strategies allow to operate backward chaining search or bidirectional search and more generally undirectional search. FDP could be improved with other evaluations of the minimal distance to the goals (Haslum, Bonet, & Geffner 2005) and concurrent bidirectional searches which could cooperate through valid or invalid states. The lack of termination criterion will be also addressed in future work. Finally FDP could be extended to handle valued actions and to compute plans of minimal costs. Also, planning with ressource will be a matter of development.

References

Baioletti, M.; Marcugini, S.; and Milani, A. 2000. Dpplan: An algorithm for fast solutions extraction from a planning graph. In *AIPS*, 13–21.

Blum, A., and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, 1636–1642.

Dechter, R. 2003. *Constraint Processing.* Morgan Kaufmann, San Francisco.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In Veloso, M. M., and Kambhampati, S., eds., *AAAI*, 1163–1168. AAAI Press AAAI Press / The MIT Press.

Korf, R. 1985. Macro-operators: A weak method for learning. *Artificial Intelligence* 26(1):35–77.

Lopez, A., and Bacchus, F. 2003. Generalizing graphplan by formulating planning as a CSP. In Gottlob, G., and Walsh, T., eds., *IJCAI*, 954–960. Morgan Kaufmann.

Mackworth, A. 1977. Consistency in networks of relations. In *Artificial Intelligence*, 8:99–118.

Rintanen, J. 1998. A planning algorithm not based on directional search. In *KR*, 617–625.