RIFO within IPP

Jana Koehler

Institute for Computer Science
Albert Ludwigs University
Am Flughafen 17
79110 Freiburg, Germany
koehler@informatik.uni-freiburg.de

Abstract

The paper describes the implementation of the RIFO family of heuristics to detect irrelevant operators and initial facts within the IPP planning system. It discusses the main differences to the original RIFO method that was developed by Bernhard Nebel for STRIPS operators. For IPP, this method was extended to handle conditional effects and negation. Furthermore, a metastrategy was added, which allows the planner to decide when to activate RIFO depending on the number of objects and ground operators in a planning problem. A summary of results from the 1998 planning competition shows the effectiveness of this strategy.

Technical Report No. 126

1 Introduction

RIFO [NDK97] is a family of heuristics that enables a STRIPS planner to detect irrelevant initial facts and operators in a planning problem. It was originally developed as a preprocessing module for the GRAPHPLAN system. This means, it reads in a GRAPHPLAN operator and fact file and returns a GRAPHPLAN operator file probably containing less operators and a fact file containing probably less objects in the type declaration and less initial facts. Then GRAPHPLAN is run on these RIFO output files.

For IPP, RIFO was extended to conditional effects and integrated into the planning system in two different ways:

1. by providing switches in IPP that allow to select a specific RIFO heuristic *after* the instantiation process has determined the set of actions satisfying inertia and *before* the planning process starts.

2. as a metastrategy that is activated by the planner when a planning problem satisfies predefined threshold values for two different parameters.

2 The RIFO Family of Heuristics

RIFO uses a backchaining technique from the goals to the initial facts. During backchaining from the goals it applies all possible operators, but ignores any conflicts between operator applications. The result is an AND-OR tree, called *fact generation tree* where AND nodes are goals or preconditions of instantiated operators and the OR nodes are single ground atoms which can be generated by different instantiated operators.

Definition 1 *[NDK97] An OR-node of a fact generation tree is considered to be solved if it is an initial fact or if one of its immediate children is solved. An AND-node is solved if all of its children are solved. The entire fact generation tree is solved if the AND-node corresponding to the goals is solved.*

Usually, a fact generation tree is generated until it is solved. But for unsolvable planning problems it can happen that the tree remains unsolved. In this case, generation proceeds until a preset depths of the tree is reached in order to guarantee termination.

To detect the operators and initial facts that are *relevant* for the planning process, a *minimum set* of initial facts, i.e., a set with a minimal number of elements, should be determined that solves the tree. RIFO uses an approximation strategy to find such a set as a solution of the fact generation tree.

For every AND- and OR-node, the set of sets of initial facts that could be used to generate this specific node are determined. These sets are recorded in so-called *possibility sets*. Given the planning problem

initial state: a, b, c
goal: x, y

together with the STRIPS operators

O1: a, $b \Rightarrow x$, z, $\neg\ a$
O2: b, $c \Rightarrow x$
O3: $c \Rightarrow y$, $\neg\ c$

RIFO generates the fact generation tree shown in Figure 1. The possibility set for the goals is $\{\{a, b, c\}, \{b, c\}\}$, and the minimum set of initial facts that solves the fact generation tree is $\{b, c\}$.

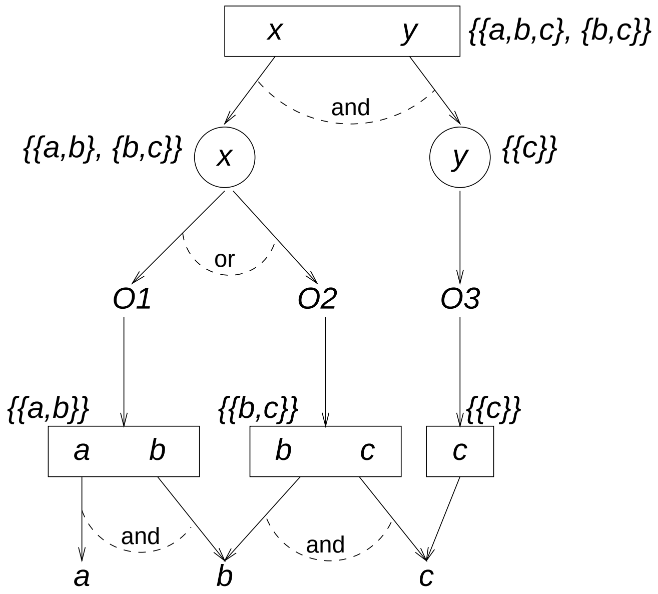

Figure 1: A RIFO fact generation tree. Cited with friendly permission from [NDK97]

During the development of RIFO [NDK97] it turned out that the computation of possibility sets is computationally expensive because the number of elements in these sets grows exponentially with the depth and breadth of the tree. Even worse, computing the minimum set of initial facts is a generalization of the NP-hard *hitting set* problem. Therefore, not all elements of the possibility sets are computed, i.e., at every node only the 10 smallest sets computed so far, are stored.

After having determined the approximation of the possibility set, several heuristics are available that select and merge the elements of the possibility set in different ways [NDK97] :

m 1: Use the union over all elements in the possibility set.

m 2: Use the union over all set-inclusion minimal sets.

m 3: Use the union over all minimum sets.

m 4: Pick a minimum set

Each of these heuristics returns a possibly reduced set of initial facts in which only the supposedly relevant objects are contained. In the example, **m 1** and **m 2** would return $\{a, b, c\}$, and **m 3** and **m 4** would return $\{b, c\}$.

Based on the selected set of initial facts, another set of heuristics decides which operators and objects will be selected and forwarded to the planner:

l 1: Select all objects in the set of probably relevant initial facts as relevant. Select initial facts and ground operators that use only these relevant objects.

l 2: Select initial facts appearing in the possibility set. Operators are selected as with the **l 1** heuristic.

l 3: Use only ground operators that appeared in the fact generation tree, i.e., that have been selected while backchaining from the goals and that can be generated by the set of probably relevant initial facts.

In the example, **l 1** and **l 2** select all three operators as relevant, while **l 3** only selects **O2** and **O3** as relevant.

Unfortunately, RIFO can remove too many operators and initial facts and is therefore not completeness-preserving. But for almost all planning problems, there is a combination of heuristics that allows to filter out some irrelevant information while preserving enough initial facts and ground operators. This means, although theoretically incomplete, RIFO is a practically valuable tool for a planning system, which allows it to find a solution more quickly. In many situations, RIFO enables a theoretically complete planning system to find a solution at all, because the original problem formulation generates a search space than cannot be completely searched given a limited amount of time and memory.

3 RIFO within IPP

Since RIFO proved to be very powerful as a separate preprocessing module for GRAPHPLAN, the heuristics were directly embedded into the IPP planner. For this purpose, the backchaining process had to be extended such that it can handle conditional effects in operators. This is rather straightforward: whenever an operator is applied to match a particular goal node, the various effects that match this goal node are considered separately. This means, an OR-node is not only generated when different operators can achieve a goal, but also when different conditional effects of the same operator can achieve it. The preconditions and effect conditions together create the new AND-node. The detection of irrelevant initial facts has also been accomodated to deal with conditional effects. When simply deleting facts from the initial state, one risks that certain conditional effects do no longer "fire" in the plan, although they are contained in the actions and would "fire" when planning from the complete initial state, i.e., invalid conditional plans could be generated. IPP, however, selects a set of relevant objects from the initial facts and repeats the instantiation procedure of operators with the set of relevant objects, i.e., such conditional effects are not contained in the actions, cf. the description of the **l 1**, **l 2**, and **l 3** selection methods above.

Second, the backchaining process is no longer taking a set of operators as input and instantiates them when building the fact generation tree, but instead IPP forwards the set of actions to RIFO. This means, RIFO is used as a filter after the instantiation process and before the planning graph is built. It turned out that the inertia handling of IPP has a very positive influence on RIFO because it reduces the size of the fact generation tree. Furthermore, the number of relevant facts and actions that are selected is often smaller than in the RIFO preprocessor for GRAPHPLAN because inertia are no longer considered and all operators that violate inertia have already been eliminated by the IPP instantiation process. This avoids that invalid sets of operators are selected during RIFO's backchaining process because they have already been filtered out by the inertia handling. In contrast to this, GRAPHPLAN's instantiation procedure can generate inertia-violating operators, which can then possibly be selected by RIFO making the planning problem unsolvable.

To activate RIFO within IPP, the above listed heuristics are activated by the corresponding switches of the planner, for example by calling IPP with option -rm 1, -rl 3 etc. The selected set of actions and initial facts is directly forwarded to the planning module. The combination of the heuristics **m 3** (switch -rm 3) and **l 1** (switch -rl 1) is the default setting in the planner. The combination of **m 4** together with **l 3** provides the strongest possible filter leading to a very powerful heuristic because the number of operators and this way the size of the graph is significantly reduced.

The implementation of RIFO within IPP allows for very flexible experimentation. The planner can be run on individual problems with or without RIFO, and by combining the different RIFO heuristics in various ways. This way, one can systematically explore the impact of the heuristics on a particular planning problem. Usually, one will start with the strongest and most selective combination of the heuristics and then relax it gradually if the planner returns no solution. Unfortunately, the stronger a heuristic the more likely it is to fail, i.e., it can make the planner incomplete because too many actions or initial facts have been removed. In particular, incompleteness of the action set occurs quite frequently because RIFO ignores potential conflicts between actions when building the backchaining tree. This can cause the planner to be confronted with unsolvable action conflicts in the subsequent planning process, i.e., no solution is found although the planning problem was solvable in its original formulation. In such a situation, one cannot tell if RIFO made the planner incomplete or if the planning problem is indeed unsolvable. Therefore, one has to run the planner without RIFO in order to be able to answer this question.

The following ordering describes a good "meta heuristic" that tries RIFO with a decreasing strength of parameters and that is used frequently for manual testing: [1]

1. ipp -rr 1 -rm 4 -rl 3
2. ipp -rr 1 -rm 3 -rl 3
3. ipp -rr 1 -rm 3 -rl 2
4. ipp -rr 1 -rm 3 -rl 1
5. ipp -rr 1 -rm 1 -rl 1
6. ipp -rr 0

By experimenting this way, one can gradually weaken the selection process and try the planner on an increasing search space. If all heuristics have led to an unsolvable planning problem, the original problem formulation is given to IPP.

4 The RIFO Metastrategy

The idea of gradually weaken the RIFO selection process and trying the planner on an increasing search space provides the basis for the RIFO metastrategy. Instead of invoking the RIFO heuristics by hand, IPP decides under which conditions they should be activated.

[1] The -rr switch activates (value 1) or deactivates (value 0) RIFO. The -rm and -rl switches select the desired combination of heuristics as described above.

The RIFO metastrategy within IPP takes advantage of RIFO's capability to drastically reduce the search space while still preserving completeness of the planner. IPP now decides to activate RIFO if a planning problem turns out to be "rather large", i.e., one can assume that it might generate a search space that is too huge, such that IPP can search it in reasonable time. Practical experience with planning systems has shown that the number of objects and the number of ground actions has a significant influence on the search space. Therefore, IPP uses these parameters to decide if RIFO should be activated:

```
if ( (ground_ops_count > OPS_THRESH && objects_count > OBJ_THRESH))
        strategy_count = 2; /* strong filtering */
      else if ( objects_count > OBJ_THRESH )
        strategy_count = 1; /* weak filtering */
      else
        strategy_count = 0; /* no filtering */
```

The threshold values have to be determined empirically and vary quite significantly in different domains. For example, in the blocksworld one can usually expect that more than 15 blocks make a planning instance very hard for IPP. In other domains, this number can be quite higher, e.g., it grows as large as 150 objects in the manhattan domain. Similarly, the number of actions, which can generate a search space that lies beyond the capabilities of IPP, can be very different in the various domains.[2]

Two different heuristics are combined in the RIFO metastrategy. *Strong filtering* uses the strongest possible combination of the heuristics **m 4** and **l 3**. This means, one best cardinality-minimal set is chosen and only those actions are selected that appeared in the fact generation tree and that can be generated by this set. Although this filtering process has a very high risk of determining too many actions as irrelevant, it is extremely powerful in reducing the search space size significantly. This means, if completeness is preserved under this strong combination of heuristics, the planner has a good chance of scaling to very large planning problems. An example for this effect occurred in the final round of the planning competition and is discussed in the subsequent section. *Weak filtering* combines the heuristics **m 4** and **l 2**, i.e., the fact selection process is still very restrictive, but the selection of actions is much weaker, i.e., all actions using relevant objects remain in the action set that is forwarded to the planner. Empirical observations suggest this particular combination of the two heuristics:

1. Selecting one best cardinality minimal set greatly reduces the number of objects for the planning system and very often it still preserves completeness. Using other selection heuristics might preserve completeness in more cases, but then the number of objects is only marginally reduced, which means that also more actions result during operator instantiation and thus the planner will more likely fail in solving the problem because it exceeds the available computational resources.

2. Since the metastrategy is only activated when a critical value of the threshold parameters is reached, it is important to try the strongest possible filtering of actions first. However in some cases, only selecting the actions that appeared in the fact

[2]Note that the number of actions depends indirectly on the number of objects because of the combinatorics during the instantiation process.

generation tree risks that all actions that will enable the planner to do conflict resolution are also removed. This problem is overcome by the weaker heuristics that selects all actions manipulating relevant objects.

The metastrategy first tests both threshold values. This means, the strong filtering is activated only if the number of objects and the number of actions exceed the preset parameters. If the reduced fact and action set makes the planner incomplete, weak filtering is tried. If again the planner returns no solution, IPP is called on the original problem formulation. Figure 2 summarizes the RIFO metastrategy.

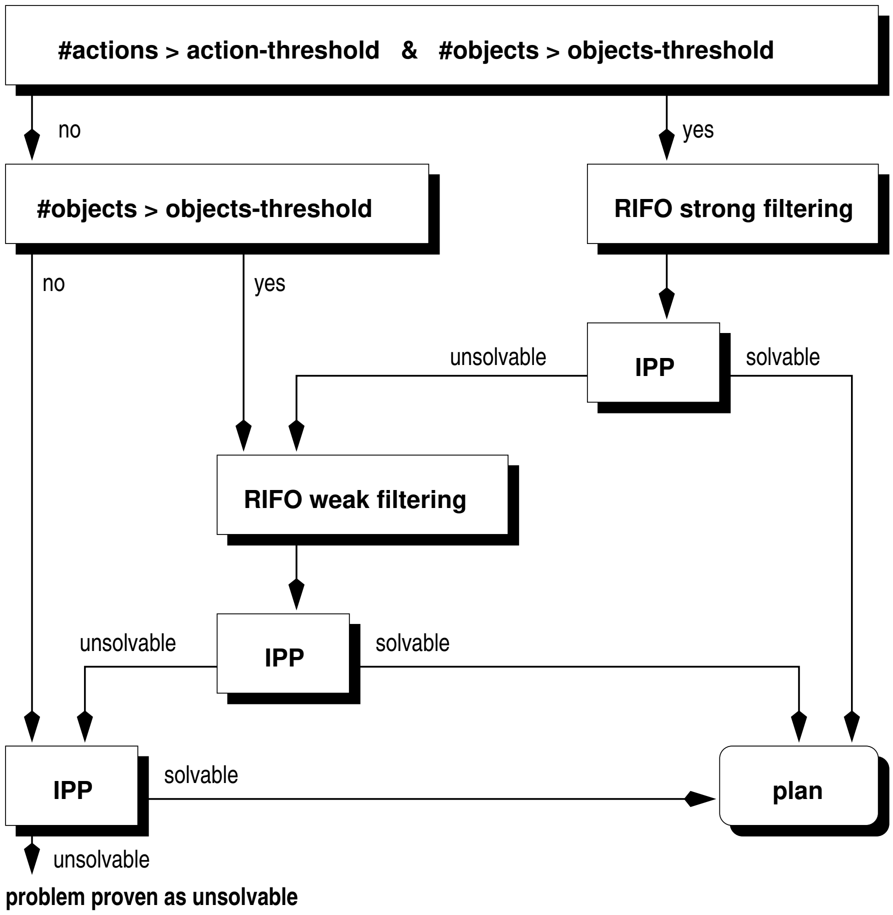

Figure 2: The RIFO metastrategy.

If a planning problem has a value for one of the parameters that remains below the threshold, the metastrategy tests if only the number of objects exceeds the preset value. In this situation, there is still a large number of objects involved in the planning problem, but the combinatorics of the instantiation process is moderate as the threshold parameter for the set of actions is not exceeded. In this case, just trying weak filtering seems to be the most appropriate, i.e., the number of objects is reduced, but action selection is less restrictive.

If both strategies have led to an unsolvable planning problem or if none of the parameters is exceeded, the planner is tried on the original problem formulation. This cascade-line metastrategy preserves completeness of the planning process. It can lead to a significant improvement of IPP's performance if one of the RIFO filtering processes enables the planner to solve a problem more quickly as without any filtering. But it can also introduce an additional overhead if an unsolvable, but huge planning problem is posed to the planner. In this case, RIFO would be activated to reduce the search space for the planner, but the failure of the planning process could also have been caused by the too restrictive RIFO filtering. This means, the planner would try to solve the planning problem three times until unsolvability can be definitely decided. Fortunately, IPP fails very quickly when RIFO has selected too few actions, i.e., usually the goals remain exclusive until the fixpoint of the graph is reached and no search needs to be performed.

5 Empirical Evaluation

It is well known that computational problems occur when conditional effects are translated into sets of STRIPS operators in order to motivate the direct treatment of conditional effects in IPP. Using the example of a roundtrip problem in the briefcase domain, one can easily demonstrate that GRAPHPLAN shows a very bad performance on the translated problem formulation, cf. Figure 3.

#objects	#locations	#operators	#actions	plan length	cpu time in s
1	2	4	12	3	0.02
2	3	6	48	5	0.11
3	4	10	152	7	1.14
4	5	18	440	9	173.48
5	6	34	1212	11	15656.58

Figure 3: Dramatic performance decrease of GRAPHPLAN on briefcase roundtrip problems caused by the translation of conditional effects into sets of STRIPS operators.

One might argue if using RIFO to select only the relevant operators could help in improving the effectiveness of the translation. Unfortunately, it turns out that running the original RIFO implementation on these problems as a predecessor for GRAPHPLAN, all initial facts and operators are selected as relevant even under the strongest combination of heuristics. Thus, no improvement of GRAPHPLAN's performance can be achieved in this case.

In contrast to this, RIFO within IPP using the strongest combination of heuristics and backchaining with the original ADL actions containing conditional effects can eliminate all **take-out** actions and a large fraction of the **move** actions by deciding that the roundtrip should be performed in a specific order, cf. Figure 4

In Figure 4, the first column lists the number of objects that have to be collected in the roundtrip problems. The second column shows the number of relevant actions that were selected out of the total number of actions resulting from the instantiation process. Column 3 shows the number of initial facts, which could not be reduced by RIFO. The last column shows the cpu time that is needed to solve the planning problem including the time to run RIFO.

As a side-effect of the filtering process, the graph not only contains less actions, but also more mutex relations, which reduce the search space significantly. Without RIFO, the combinatorial explosion is encountered when more than 5 objects have to be collected, cf. Figure 5. With RIFO, this number is extended to 7 objects.

objects	actions selected/total actions	initial facts	problem solving time
1	3/8	6	0.01
2	7/18	10	0.02
3	12/36	14	0.08
4	18/60	18	0.43
5	25/90	22	3.53
6	33/126	26	27.03
7	42/168	30	199.43
8	52/216	34	1314.3

Figure 4: Improved scalability of IPP in the briefcase roundtrip example after reduction of the action set with RIFO.

objects	time steps	actions	UCPOP	Prodigy	IPP 2.0	IPP 4.0
1	3	3	0.04	0.24	0.01	0.01
2	5	5	0.14	0.71	0.07	0.03
3	7	7	0.58	0.55	0.26	0.08
4	9	9	24.80	0.89	3.83	0.67
5	11	11	-	1.1	285.14	44.50

Figure 5: UCPOP, Prodigy, and IPP on briefcase roundtrip problems.

Many other planning problems could be listed to show the effectiveness of RIFO. For example, the bw_large.b problem from the SATPLAN test suite contains 242 actions, 26 initial facts and is solved by IPP 3.3 in 129 s. Using RIFO strong filtering, only 72 actions remain as relevant, the number of initial fact is reduced to 22 and planning time reduces to 0.89 s. Even more impressive, for the manhattan problem the number of actions is reduced from 1948 down to 108. Planning time is reduced from 210 s down to 40 s, but RIFO itself needs 34 s for analysis.

A very impressive demonstration of RIFO took place during the second round in the planning competition. Here, the RIFO metastrategy enabled IPP to solve more problems than any other of the planners in this test set. After a few trials on the test set for round 2, the threshold values were set to 3500 for the number of actions and to 35 for the number of objects. The influence of the metastrategy on the search space of each problem that had to be solved in this round is shown in detail in Figure 6.

The first column shows the problem name. The second column lists the number of actions and objects in the original problem formulation. The third and fourth columns list the number of actions and objects selected by strong and weak filtering. In parentheses follows the length of the plan if IPP could find one. ($\perp$) means the planning problem became unsolvable, - means RIFO was inactive, (-) means no plan was found because the planner either exceeded the cpu time or memory limit.

The results were obtained by repeating the competition tests from round 2 on a Sun Ultra 1/170 under a 10 minutes cpu time limit and a 120 Mb memory limit. During the competition, IPP could not be watched solving the problems and no recording of the RIFO parameters was possible.

Weak filtering is activated after a failure of strong filtering as in the case of the *grid3*

problem	original	strong filtering	weak filtering
log1	571/25 (13)	-	-
log2	502/21 (20)	-	-
log3	958/26 (27)	-	-
log4	3561/42 (-)	189/30 (-)	-
log5	4985/53 (-)	119/26 (31)	-
mprime1	7809/36 (5)	7/9 ($\perp$)	11/9 ($\perp$)
mprime2	3281/32 (8)	-	-
mprime3	97259/68 (-)	-	-
mprime4	8485/42 (5)	7/10 (4)	-
mprime5	6773/22 (6)	-	-
grid1	2610/38 (14)	-	80/11 (20)
grid2	4501/50 (-)	69/19 ($\perp$)	260/19 (31)
grid3	7256/64 (-)	315/21 ($\perp$)	557/21 (-)
grid4	11151/80 (-)	135/24 (47)	-
grid5	16240/98 (-)	1481/49(-)	-

Figure 6: Influence of the RIFO metastrategy on IPP's search space in the second round of the planning competition.

problem or when the number of actions remains below the threshold parameter of 3500 and only the number of objects exceeds 35 as in the *grid1* problem. As the analysis shows, only 8 problems can be solved by IPP without RIFO, but in using the metastrategy 3 more solutions are found.

References

[NDK97] B. Nebel, Y. Dimopoulos, and J. Koehler. Ignoring irrelevant facts and operators in plan generation. In S. Steel, editor, *Proceedings of the 4th European Conference on Planning*, volume 1348 of *LNAI*, pages 338–350. Springer, 1997.