

# The FAR-OFF System: A Heuristic Search Case-Based Planning

Flavio Tonidandel<sup>1</sup> and Márcio Rillo<sup>1,2</sup>

<sup>1</sup> Universidade de São Paulo - Escola Politécnica  
Av. Luciano Gualberto 158, trav3, São Paulo, SP, Brazil, 05508-900

<sup>2</sup> UNIFEI  
Av. Humberto de A. Castelo Branco, 3972, São Bernardo do Campo, SP, Brazil, 09850-901  
E-mails: [flavio@lac.usp.br](mailto:flavio@lac.usp.br), [rillo@lsi.usp.br](mailto:rillo@lsi.usp.br)

## Abstract

This paper presents a case-based planning system, called FAR-OFF, as an alternative approach to heuristic search generative planners, like HSP and FF. The FAR-OFF system introduces the use of a new similarity metric, called ADG (*Action Distance-Guided*), that is more accurate than other similarity metrics, and it also incorporates techniques to decrease the retrieval time, like the Footprint-based Retrieval. Consequently, the FAR-OFF system can solve more AIPS'00 planning competition problems than the heuristic search planners in an acceptable amount of time.

## Introduction

Case-Based Planning (CBP) systems are alternative approaches to generative planning. Most of them use old plans stored in a case-base - memory with plans as cases - for helping to solve problems, while generative planning constructs plans from scratch. Although theoretically the complexity of both is *NP-hard* (Nebel and Koehler 1995), in practice CBP systems can outperform generative planning systems, because they avoid repeating the planning process for similar problems.

Since the appearance of the Graphplan system (Blum and Furst 1997), most generative planning systems are fast and solve a great number of planning problems. Some of them are the heuristic search based systems, like the HSP (Bonet and Geffner 2001) and the FF systems (Hoffmann and Nebel 2001), which are much faster than the old generative planning systems.

A CBP system that makes use of old case-based reasoning techniques can rarely outperform the current heuristic search planners. This is because CBP systems depend on the similarity rule accuracy and the number of cases that will be considered. The old similarity metrics are not accurate enough because they are based on the differences between states, and the old techniques of case-base reduction, such as indexing, do not consider a great part of important cases in its reduced set of cases.

However, new results in Case-Based Reasoning create a possibility to develop new and more efficient CBP systems. These systems can be based on new and more accurate similarity metrics as well as some methods that are based on cases competence - the range of problems that the cases can solve.

This paper introduces a case-based planning system called FAR-OFF (*Fast and Accurate Retrieval on Fast Forward*). It uses a generative planning system based on the FF system (Hoffmann and Nebel 2001) to adapt similar cases, and a new similarity metric, called ADG, that presents more accurate results than old similarity metrics (Tonidandel and Rillo 2001a).

Beyond the new ADG similarity metric, the FAR-OFF system uses a new competence-based method, called Footprint-based Retrieval (Smyth and McKenna 1999), to reduce the space of cases that will be searched.

Because a heuristic search planning system is used to adapt cases and the ADG uses a similar heuristic to determine similar cases to be retrieved, the FAR-OFF system can be called heuristic search case-based planner.

With all these features, the FAR-OFF system can outperform the heuristic search planning in number of problem solutions.

The next section presents a general view of the case-based planning and its relation with generative planning and new Case-Based Reasoning techniques. After the FAR-OFF system is described. In the following, some experiments about the performance and results of the FAR-OFF system are shown. A discussion and the conclusion are presented in the end of the paper.

## Case-Based Planning

Most planning systems are generative planners. They find a solution from scratch, i.e., they always start the planning process from the beginning, even when the same problem was previously solved.

The use of Case-Based Reasoning (CBR) techniques (Kolodner 1993) can avoid a planner to repeat previous executed planning processes for problems that were previously solved. The first application of CBR in planning was by the CHEF system (Hammond 1989), which received the denomination of Case-Based Planning (CBP) system.

The CBP systems are distinguished from generative planners by using a memory of previous executed plans<sup>1</sup>. Each previous plan is usually stored as a case with specific

---

<sup>1</sup> There are CBP systems that do not consider cases as old plans, like the Prodigy/Analogy system. They use derivational analogy where cases are traces of old planning solutions. However, in this paper, we will focus only on transformational analogy, that considers cases only as old plans.

features of the situation where the plan was applied. These features are used to indicate how much similar is a case to the solution of a new problem. Usually, a similar case is retrieved by a retrieval phase, which uses a similarity rule to rank cases from the most to the least similar one.

After the choice of the most similar case, a CBP system starts the adaptation process. In most of CBP systems, the adaptation phase is composed of a generative planner that permits the modification of a previous plan in order to find a new solution. This new solution, which is a plan, can be stored for future uses. These three processes – storing, retrieving and adapting – completes the cycle of a common CBP system.

Since this cycle, when suitably designed, can solve planning problems by itself, some CBP systems designed in the past do not consider any generative planning in their structure, and find a solution only by the cases stored in the case-base. These CBP systems are called *reuse-only* systems and. In this approach we can find CBP systems like CHEF (Hammond 1989) and CAPER (Kettler 1995).

However, *reuse-only* systems cannot find any planning solution from scratch. As consequence, they cannot find a solution if they do not find a proper case in the case-base that can be adapted through single rules.

An alternative approach to *reuse-only* systems is the *reuse-optional* approach, which uses a generative planning system that is responsible to adapt the retrieved cases. This feature allows a CBP system to solve problems that cannot be solved only by using stored cases and simple rules in the adaptation phase.

In fact, a *reuse-optional* case-based planning can be considered as an extension of a generative planning system by confining it in the adaptation phase and adding more the retrieval and the storing phases. Some CBP systems, like Prodigy/Analogy (Veloso 1994), SPA (Hanks and Weld 1995) e MRL (Koehler 1996), are examples of *reuse-optional* systems, because they use the generative systems NOLIMIT, SNLP e PHI respectively in their structure.

Empirically, a great number of *reuse-optional* CBP systems have shown that the use of a case-base can permit them to perform better in processing time and in number of planning solutions than the generative planning that they incorporate.

However, theoretically, any CBP system has the same complexity of a generative planning system. Nebel and Koehler (1995) proved that both are *NP-Hard*. They also conclude that “...*plan modification does not lead to a provable efficiency...*” and that if the generative planning is very efficient, “... *the costs for matching and modification can easily be higher than the costs for generating a plan from scratch*”. However, they emphasize that the efficiency bottleneck is in the retrieval phase and it is mainly caused by the *matching* problem.

The *matching* problem is caused by the similarity metric that, when not accurate, overloads the adaptation phase. One attempt to avoid the matching problem is by developing a similarity rule that anticipates the adaptation effort.

Some similarity rules based on adaptation were proposed in the past, such as RCR similarity in DIAL system (Leake et al. 1997) and AGR (*Adaptation-Guided Retrieval*) in DéjàVu system (Smyth and Keane 1998). However, all these similarity rules and techniques depend on the application domain, and sometimes require some additional domain information to estimate the effort of the adaptation phase. Consequently, they cannot be directly applied in domain independent case-based planning.

On the other hand, domain independent similarity rules are usually based on the number of common features between the current problem and the stored case. These common features are obtained by the intersections between the current state and the initial state of the stored case and between the goal state and the final state of the stored case. If the total number, or normalized number, of the common features between a stored case and the current problem is higher than a specific limit, the stored case can be considered similar.

However, this approach is not a suitable measure to improve the adaptation phase efficiency because it is not a good measure of the real adaptation effort. Most of state-space CBP systems use this kind of similarity rule.

An alternative approach to planning with states is the hierarchical case-based planning systems (Muñoz-Avila and Hüllen 1996). They have no goals, but only tasks to be achieved. Since tasks are semantically different from goals, the similarity metric designed for hierarchical CBP systems is also different from the similarity rules designed for state-space CBP systems.

An interesting similarity rule in hierarchical approach is presented in the CAPLAN/CBC system (Muñoz-Avila and Hüllen 1996). It extends the similarity rule introduced by the PRODIGY/ANALOGY system (Veloso 1994) by using feature weights in order to reduce the errors in the retrieval phase. These feature weights are learned and recomputed according to the performance of the previous retrieved cases.

However, all these approaches use either additional knowledge or learning from past retrieval to estimate the adaptation effort. On the other hand, the ADG (*Action-Distance Guided*) (Tonidandel and Rillo 2001a) similarity does not require any additional knowledge or learning process to present an accurate estimate of similarity between cases and problems. The ADG similarity is a process that uses the heuristic of the FF planning system to estimate the distance between states in number of possible actions.

The FAR-OFF system, presented in this paper, is a *reuse-optional* case-based planning that uses the ADG to estimate the similarity of cases in a case-base and a FF-based generative planning system to adapt old plans retrieved by a Footprint-based Retrieval (Smyth and McKenna 1999). In the following section, the FAR-OFF system will be presented in detail and it will be shown that it can present better performance than the heuristic search planning in some situations.

## The FAR-OFF system

The FAR-OFF system is a case-based planning system that uses new and more accurate methods of retrieval. This paper presents its Strips-version, with Strips-like states, actions and plans models formalized in Transaction Logic.

The Transaction Logic (TR) (Boner and Kifer 1995), in its serial-Horn version, is an extension of the first-order logic by the introduction of the serial conjunction operator ( $\otimes$ ), e.g.,  $\alpha \otimes \beta$  means "first execute  $\alpha$ , then execute  $\beta$ ".

TR is a suitable logic to describe actions and plans for planning systems (Tonidandel and Rillo 1998) (Tonidandel and Rillo 2000) (Santos and Rillo 1997). As TR is suitable for planning, it is suitable for CBP system components as well, because the case memory is a collection of plans.

It uses the following notation to describe a transaction:  $P, D_0, \dots, D_n \models \phi$ , where  $\phi$  is a transaction formula and  $P$  is a set of TR formulas called transaction base. Each  $D_i$  is a database state that is a set of first-order formulas. Intuitively,  $P$  is a set of transaction definitions,  $\phi$  is an invocation of some of these transactions;  $D_0, \dots, D_n$  is a sequence of databases that represents an updating made by  $\phi$ . This updating is performed by  $ins(\_)$  and  $del(\_)$  predicates, that inserts and deletes predicates of a state respectively. On the other hand, a situation of a query is not given by a sequence of databases, but by just one state. For example,  $P, D_k \models qry(c)$ , where  $c$  is true in  $D_k$ .

In order to delimit the planning components in a TR framework, some definitions are stated in accordance with Santos and Rillo's (1997) work. Considering  $L$  a language defined in serial-Horn version of TR, the components of a planning system can be defined as:

**Definition 1 (State):** *The state  $D$  is a finite set of first-logic predicates and it is represented in TR as a database state. Each  $d \in D$  is called fact.*

**Definition 2 (Strips-like Actions):** *Considering  $A \subseteq L$  as a set of action definitions, each  $\alpha, \alpha \in A$ , has the following structure:*

$$\alpha \leftarrow pre(\alpha) \otimes delete(\alpha) \otimes add(\alpha)$$

where

- $pre(\alpha)$  is a TR formula that is composed by  $qry(\_)$  predicates that represents the preconditions of the action
- $delete(\alpha)$  is the delete list of the action  $\alpha$ . It represents the facts that were true before and will not be true after the action execution.
- $add(\alpha)$  is the add list of the action  $\alpha$ . It represents the facts that were not true before and will be true after the action execution.

The result of an action execution is an updated state  $D'$  from  $D$  after the deletion and the insertion of the delete and add lists respectively. Any action  $\alpha$  only can be executed from a state  $D$  if  $pre(\alpha) \subseteq D$ .

With the definitions of actions and state, it is possible, following (Tonidandel and Rillo 1998), to define plans, goals, and cases with the use of the TR:

**Definition 3: (Plan)** *A plan  $\delta = \alpha_1 \otimes \dots \otimes \alpha_n$  is a TR formula, where  $\alpha_i \in A$ ;  $1 \leq i \leq n$ .*

**Definition 4: (Goal)** *A goal  $Df$  is a TR formula and it is a set of queries that represents the desirable final state.*

When the planner finds a desirable sequence of actions in order to reach  $Df$ , the plan can become a case to be stored for future uses.

### The Case-Base

A case-base, for case-based planners, is a collection of old plans. However, these old plans are stored in cases form. A case is a plan connected with initial and final states features:

**Definition 5: (Case)** *A case  $\eta$  is a TR rule:*

$$\eta \leftarrow Wi \otimes \alpha_1 \otimes \dots \otimes \alpha_n \otimes Wf,$$

where:

- $\eta$  is a TR rule that represents a stored case.
- $\alpha_i \in A$ ;  $1 \leq i \leq n$ , a plan defined by the planner that satisfies a proposed goal.
- $Wi$  is a set of queries in TR that represents the precondition of the case.
- $Wf$  is a set of queries in TR that represents the postcondition of the case

Intuitively,  $Wi$  is a set of those facts that must be in the initial state and represents the pre-condition of the plan. It can be obtained by analyzing the actions of the plan. Each fact, which is in the precondition of an action and that was not inserted by any other action before, can be considered in the  $Wi$  set.

With respect to  $Wf$ , it is a set of those facts that are inserted by the plan and will be presented in the final state after the plan execution. It can be obtained by a process that is similar to  $Wi$  process.

The  $Wi$  and the  $Wf$  are the most important features for the FAR-OFF efficiency, because they are the core part of the ADG similarity and influence the retrieval phase.

Each case-base has a limited number of cases. Each case is a complete plan with grounded actions and grounded predicates that correspond to  $Wi$  and  $Wf$ . The case-base is also composed of an extra feature that indicates the footprint cases and their respective Related Sets. Footprint cases and Related Sets are explained in the next section.

In fact, the FAR-OFF system has three phases: the retrieval, the adaptation and the storing phases. Each phase contributes to the efficiency of the entire system. In the following, these three phases will be further detailed.

### The Retrieval Phase

In case-based planning systems, the retrieval phase is the most time expensive phase. It influences directly and indirectly the performance of a CBP system. It directly influences when it searches the case memory in order to find a similar case to be considered. Indirectly, it

influences the system performance because the similar case that it chooses will require some effort to be adapted.

In order to minimize those influences, the FAR-OFF system applies two important methods. One is a method to reduce the retrieval processing time through a process that considers the cases competence: Footprint-based Retrieval (Smyth and McKenna 1999). Another method is a new similarity metric, called ADG (Tonidandel and Rillo 2001a), that chooses suitable similar cases to decrease the adaptation effort.

**The ADG Similarity Metric.** An accurate similarity metric that considers the effort of the adaptation phase is necessary to improve the system efficiency. It means that the most similar case must require less effort to be adapted.

The ADG (*Action-Distance Guided*) similarity (Tonidandel and Rillo 2001a) has as a purpose to estimate the adaptation effort by estimating the number of actions that is necessary to transform a case to a solution of a problem. This estimating result in a similarity metric is more accurate than the results from old similarity rules usually applied in CBP systems, like SNN (Kolodner 1993).

The ADG similarity is a process that uses the  $W_i$  and the  $W_f$  of the stored cases and calculates two estimate values of the distance between states. Both estimates are based on a number of actions. The first value is called *initial similarity value* ( $\delta_i$ ), and it estimates the distance between the current initial state, denoted by  $D_0$ , and the initial state features of the case, denoted by  $W_i$ . The second value is called *goal similarity value* ( $\delta_G$ ) and it is a distance estimate between the desirable final state ( $D_f$ ) of the goal (def. 4) and the final state features ( $W_f$ ) of the case.

The process of estimating the distance between two states is obtained by using the heuristic of the FF planner (Hoffmann and Nebel 2001). This heuristic is directly applied to determine the *initial similarity value* ( $\delta_i$ ), and a modified version of it is used to determine the *goal similarity value* ( $\delta_G$ )

The first step for determining the FF's heuristic is to create a graph for a relaxed planning problem. Hoffmann and Nebel (2001) explain that this relaxation is obtained by ignoring the delete list of actions. It allows the graph to find a relaxed fixpoint that is composed by all facts that are reached from the initial state.

As defined by Hoffmann and Nebel (2001), the graph is constituted by layers that comprise alternative facts and actions. The first fact layer is the initial state ( $D_0$ ). The first action layer contains all actions whose preconditions are satisfied in  $D_0$ . Then, the add lists of these actions are inserted in the next fact layer together with all facts from the previous fact layer, which leads to the next action layer, and so on. The process keeps going on until it finds a relaxed fixpoint, i.e., when there are no more fact layers that are different from previous fact layers.

Some useful information can be determined from the relaxed fixpoint process. Following (Hoffmann and Nebel 2001), they are:

---

```
relaxed_initial_length(G)
```

```
clear all Gi
for i:= 1 ... max do
  Gi := {g ∈ G | level(g) = i};
endfor
h:=0;
for i:= max ... 1 do
  for all g ∈ Gi, g False at time i do
    select αlevel=i-1; g ∈ add(α);
    h:=h+1;
  for all dlevel ≠ 0 ∈ pre(α), d not True
    at time i-1 do
    Glevel(d) := Glevel(d) ∪ {d};
  endfor
  for all d ∈ add(α) do
    mark d as True at time i-1 and i;
  endfor
endfor
endfor
return h;
```

---

**Figure 1.** The algorithm that computes the relaxed solution from a relaxed graph, where  $G$  is the target-state. It is extracted from (Hoffmann and Nebel 2001).

**Definition 6:**  $level(d) := \min \{i \mid d \in F_i, \text{ where } F_i \text{ is the } i^{\text{th}} \text{ layer of facts} \}$

**Definition 7:**  $level(\alpha) := \min \{i \mid \alpha \in O_i, \text{ where } O_i \text{ is the } i^{\text{th}} \text{ layer of actions} \}$

The definitions 6 and 7 provide the order number of the layer where each fact or action appears first. It means that each fact, or action, is a membership of the layer that it first appeared.

With the relaxed graph, it is possible to find a relaxed solution for any state that can be reached from  $D_0$ . This relaxed solution provides an estimate for the optimal length of the not-relaxed solution (Hoffmann and Nebel 2001). This estimate is suitable used to determine  $\delta_i$  and  $\delta_G$  values.

The *initial similarity value* ( $\delta_i$ ) is directly obtained by the determination of the relaxed solution from  $D_0$  to  $W_i$ . First, each fact in  $W_i$  is initialized as a goal in its correspondent layer, determined by  $level(\_)$  value. The process is then performed from the last layer to the first layer, finding and selecting actions in layer  $i-1$  which their add-list contains one or more of goals initialized in layer  $i$ . Then, the preconditions of the selected actions are initialized as new goals in their correspondent layer.

The process stops when all unsatisfied goals are in the first layer, which is exactly the initial state. The estimated number of actions between initial state and  $W_i$  is the number of action selected to satisfy the goals in each layer.

The algorithm to compute the relaxed solution, extracted from (Hoffmann and Nebel 2001), is shown in Figure 1, where the variable  $h$  is used to count the number of

selected actions. The *Initial Similarity Value* is the result  $h$  of the function  $relaxed\_initial\_length(Wi)$  after setting all marks of all facts as false:  $\delta_i = h$ .

In order to calculate the second value  $\delta_G$ , it is necessary to force the solution trace from  $D_0$  to consider the actions in the case. To do this suitably, it is necessary to maintain the values of each mark of each fact after the performing of the *Initial Similarity Value* calculation. It means that the function  $relaxed\_final\_length(Wi, Wf, Df)$  (Fig. 2) will be called using the marks changed by the  $relaxed\_initial\_length(Wi)$ .

As highlighted by Tonidandel and Rillo (2001a), the reason to keep all marks unchanged from  $\delta_i$  calculation is to avoid the re-calculation of the actions between  $D_0$  and  $Wi$ .

In addition, all marks of all facts in  $Wi$  are set as false and the marks of each fact in  $Wf$  are set as true in their correspondent layers. These are necessary because  $Wi$  are the predicates that will be deleted by the plan (case) and the  $Wf$  are the predicates that will be true after the plan (case) execution.

The result of the function  $relaxed\_final\_length(Wi, Wf, Df)$  of the algorithm presented in Figure 2 is the second part of the similarity metric:  $\delta_G = h'$ . Therefore, with  $\delta_i$  and  $\delta_G$  defined, the ADG similarity value can be determined by  $\delta_i + \delta_G$ .

The ADG is a domain independent approach and it is also designed to be used in any retrieval phase of a state-space CBP system with action-based cases, i.e., where cases and plans are sequence of actions.

The value of ADG can also be used to decide if the planning is better to be made by the adaptation of a case or by the direct application of the generative planning. The use of a case is useful when the ADG value is less than the direct distance between  $D_0$  and  $Df$ , that can be calculated with  $relaxed\_initial\_length(Df)$ . Otherwise, if the distance between  $D_0$  and  $Df$  is less than the ADG value of any stored case, the use of a generative planner is preferable.

**Competence-based Retrieval Phase.** Not only a new similarity metric is necessary to improve the system efficiency, but also a method that avoids the case-based planning to search all the cases in the case-base. Previous methods were mainly based on indexing of cases by relevant features, but they were domain dependent in most of the systems.

A domain independent alternative is a method based on the competence of the case-base. The competence is the range of problems that a case, or a case-base, can solve. As cases are good representation of problems, to perform any competence-based method we will consider the *representativeness assumption* (Smyth and McKenna 2001), and the following definitions extracted from (Smyth and McKenna, 1999):

**Definition 8:**  $CoverageSet(c) = \{c' \in C: Solves(c, c')\}$

**Definition 9:**  $ReachabilitySet(c) = \{c' \in C: Solves(c', c)\}$

---

```

relaxed_final_length(Wi, Wf, Df)

G := Df;
for each d ∈ Wi do
  mark d as False at all levels;
endfor
for each d ∈ Wf do
  mark True at level(d) and level(d)-1
endfor
h' := relaxed_initial_length(G);
return h';

```

---

**Figure 2.** The algorithm that extracts the distance between  $Wf$  and  $Df$  by considering the marks created by the  $relaxed\_initial\_length(Wi)$  algorithm.

$Solves(c, c')$  signifies that a case  $c$  can solve the case  $c'$ . A case can be consider as a new problem and some other cases can be adapted to solve it, because the *representativeness assumption* has been considered.

To reduce the space of cases, the FAR-OFF system uses the Footprint-based method in its retrieval phase. The Footprint-based Retrieval is a competence-based method for determining groups of footprint cases that represent a smaller case-base with the same competence of the original one. Each footprint case has a set of similar cases called Related Set (Smyth and McKenna 1999). The union of footprint cases and Related Set is the original case-base.

In order to calculate the footprint cases, the FAR-OFF system uses the ADG similarity in the *Solves* definition:

**Definition 10:**  $Solves(c, c')$  iff the value of the ADG of the case  $c$  for the case  $c'$  is  $< \rho$

The definition 10 shows that a case  $c$  solves a case  $c'$  if and only if the ADG similarity value of the case  $c$  for solving a problem defined as case  $c'$  is less than an upper limit  $\rho$ . This limit implies directly in the number of footprint cases of a case-base. It affects the number of cases in the *ReachabilitySet* and *CoverageSet* of each case. As a consequence, if the  $\rho$  value is high, one case could solve almost all cases and the case-base would have few footprint cases. On the other hand, if the value of  $\rho$  is small, one case could rarely solve another case and the case-base would have many footprint cases.

In accordance with Smyth and McKenna (1999), the footprint process considers a measure called *RelativeCoverage*. The definition of the Relative Coverage of a case  $c$ , according to Smyth and McKenna (1999), is done by:

$$RelativeCoverage(c) = \sum_{c' \in CoverageSet(c)} \frac{1}{|ReachabilitySet(c')|}$$

With the *RelativeCoverage* of each case in the case-base, the Footprint-based Retrieval uses a process to reduce the case-base that is similar to CNN (*Condensed Nearest Neighborhood*) and creates sets of footprint cases and their Related Sets. Other algorithms than CNN were investigated in a recent paper of Smyth and McKenna (2001). However, only the Footprint-based Retrieval with CNN algorithm is used in the FAR-OFF system.

With the footprint cases and their Related Sets, the Footprint-based Retrieval finds the similar case through two processes. First, it finds the similar footprint case among all footprint cases, and second, it looks for a similar case among the cases in the Related Set of the chosen footprint case. This method, as empirically shown by Smyth and McKenna (1999), finds the most similar case in most situations, and reduce drastically the process of case searching.

### The Adaptation Phase

The FAR-OFF system retrieves the most similar case, or the ordered  $k$  most similar cases, and shifts to the adaptation phase. Most case-base planning systems in the past investigate the modification of the retrieved case in the adaptation phase. However, this process, if not suitably designed, can be extremely time expensive.

The FAR-OFF system does not modify the retrieved case, but only completes it. The system will only find a plan that begins from the initial state and then goes to  $W_i$  of the case, and another plan that begins from the state where  $W_f$  of the case is satisfied and then goes to a state that satisfies the goal. Obviously, the completing of cases leads the FAR-OFF system to find longer solution plans than generative planners, but it avoids wasting time in manipulating case actions in order to find shorter solutions length.

To complete cases, the FAR-OFF system uses a FF-based generative planning system. It is an implementation of the FF planning system using some heuristics extracted from (Hoffmann and Nebel 2001), like added-goal deletion, helpful actions and relevant facts, and the ordering goals (Koehler and Hoffmann 2000). It is a simpler implementation of the original FF planning system because it does not incorporate the complete best first-search, that is called only when the Enforced Hill-climbing fails, as reported in (Hoffmann and Nebel 2001).

The FF-based generative planning is used by the FAR-OFF system to find a plan from the initial state to the  $W_i$  of the retrieved case. It is made possible by considering the  $W_i$  as a goal. Since the  $W_i$  is the precondition of the case, the FAR-OFF system can merge the plan found by the FF-based generative planning with the case and performs the actions of both in order to find a new intermediary state.

This intermediary state satisfies the  $W_f$  of the case and it will be considered as a new initial state for the next call of the FF-based generative planning. This calling is performed to find a plan for the goal.

The solution is then found by merging both plans that were found by the FF-based generative planning and the case. Since the retrieval phase properly chooses the most

similar case, the FF-based planning is called to find easy and short plans, what in general, is faster than finding the complete solution from scratch.

Although the FAR-OFF system considers one retrieved case at a time in the adaptation phase, it does not mean that the retrieval phase cannot choose  $k$  most similar cases and order them by their similarity. These  $k$  cases would be the cases obtained from the Related Set of the most similar footprint case.

So, if  $k$  most similar cases are retrieved and ordered from the most to the least similar one, the FAR-OFF system will try to adapt the first case. If the first case failed to be adapted for a solution, then the second case is attempted, and the process continues until the  $k$ -retrieved cases are attempted or one case can be adapted for a solution.

After finding a solution for a new problem, this new solution can be transformed to a case by incorporating the  $W_i$  and  $W_f$  features, and it can be stored for future uses by the Storing phase.

### The Storing Phase

When a new case is created, it must be stored for future uses. However, the insertion of a new case will request a new re-computation of footprint cases by updating the *CoverageSet* and the *ReachabilitySet* of all stored cases.

However, to perform the re-computation of the competence and the footprint cases, the case-base must store like the initial state and the final state of each case. They are necessary to perform the ADG similarity metric that is responsible for determining the *CoverageSet* and *ReachabilitySet*.

A case-base that stores these extra features are called *Extensible Case-Base*, because it permits the re-computation of footprint cases. However, an *Extensible Case-Base* occupies a large amount of memory to store cases and consequently requires methods for case-base maintenance.

One method that can be applied to maintain the *Extensible Case-Bases* is the minimal-injury method (Tonidandel and Rillo 2001b). It is a case-deletion approach that chooses cases to be deleted when the number of cases or the size of the memory exceeds an upper limit. A greedy algorithm chooses, at each step, a case that causes the minimal injury to the competence to be deleted (Tonidandel and Rillo 2001b).

The minimal-injury method is suitable for *Extensible Case-Bases* because it is specifically designed to control the used amount of memory and it guarantees a lower bound for the residual competence. Its application results in a case-base with a reduced number of cases, but preserving the competence.

Beyond the amount of used memory, another problem that arises from *Extensible Case-Bases* is the time to re-compute the footprint cases. The original footprint computation (Smyth and McKenna 1999) has complexity of  $O(n^2)$ , where  $n$  is the number of stored cases.

New methods for calculating the footprint cases with less or without extra stored features and a new process that

is more efficient than the original process must be investigated in the future.

## Empirical Tests

In order to validate the FAR-OFF system theory, some empirical tests must be performed. Using the same pattern as the old case-based planning systems, the tests are comparisons between the FAR-OFF system and the FF-based generative planning

For that, the AIPS'00 planning competition problems (Bacchus 2000) will be used in Blocks World and Logistic domains. Both domains are chosen because both are STRIPS domain models and they cause opposite performances in planning system. As reported in Hoffmann and Nebel (2001), the FF system has some difficulties in Blocks World domain that it does not have in Logistic domain.

Since the FF-based generative planning used by FAR-OFF is a similar version of the original FF planning, it is expected that it also performs well in Logistic domain and worse in Blocks World domain, though the implementation of the FF-based generative planning is in a different operational system and computer language. (Borland Delphi™ for Windows™).

However, in general, the FF-based generative planning has a similar performance to the original FF and can solve some AIPS'00 competition problems that original FF cannot solve and vice-versa. The main reason for different results is the generation of complete grounded facts and actions, and the order that the actions and facts are disposed in the relaxed graph generated by both systems.

The tests are performed in a computer with Windows™ Operational System, 450MHZ and 512 Mbytes of Ram Memory. Because of memory limitation, the tests will consider all regular AIPS'00 problems and only some additional AIPS'00 problems in both domains.

For each AIPS'00 problem in Blocks World and Logistic domains, the FAR-OFF system must have a case-base that provides enough cases to perform tests. A suitable case-base for generic tests is a case-base that is randomly created. For this reason a seeding system of case-bases is considered.

### The Case-Base Seeding

A Case-Base Seeding (CBS) system is used to create cases and case-bases randomly. The CBS system randomly generates an initial state and a goal state, and after it calls the FF-based generative planning to find the solution plan. Then, the found plan is stored as a case with  $W_i$  and  $W_f$  features.

The important feature for creating random cases is to create random initial and goal states properly. The initial state is constructed by following some rules that dictates which predicates must be together in a state and which cannot. The rules guarantee that the initial state is correct, and they permit that the CBS system discards all predicates that cannot be together with a predicate that was randomly

chosen to be in the initial state. For example, in Blocks World domain, the *handempty* predicate cannot be in the same state that has the *holding*(\_) predicate, so, if the CBS chooses the *handempty* predicate, it discards all *holding*(\_) predicates. The process continues until no more predicates can be considered in the state.

A similar process is used to create random goal states, however, some predicates are chosen as relevant predicates in each domain. Only relevant predicates can be in a goal state. The relevant predicates for the Block World domain are the *on*(\_,\_) and the *ontable*(\_) predicates, and for the Logistic domain only the *at*(\_,\_) predicate is considered relevant for goal state.

Following these random processes, the Case-Base Seeding creates a case-base for each type of planning problem. It means that one case-base is used for 4-Blocks problems and another is used for 5-Blocks problems. It is similar for Logistic problems.

The CBS process finishes with the calculation of the Footprint set and the Related Sets of the cases following the specifications in the definition 10. For a better performance of the FAR-OFF system, the footprint set was designed to be a percentage of the cases. Since each type of problems has different numbers of possible plans in the domain model, the CBS uses different values of  $\rho$  (definition 10) to reach approximately 25% of cases in footprint set.

The case-bases take time to be created by the Case-Base Seeding system, and this time is not considered in the experiments below.

## The Results

The results of the tests in Blocks World domain are reported in Table 1 and in Figure 3. The results of the tests in Logistic domain are reported in Table 2 and Figure 4. Each test considers a number of 5 retrieved cases by Footprint-based Retrieval, though most of Blocks World tests and all Logistic tests are solved by using the first retrieved case.

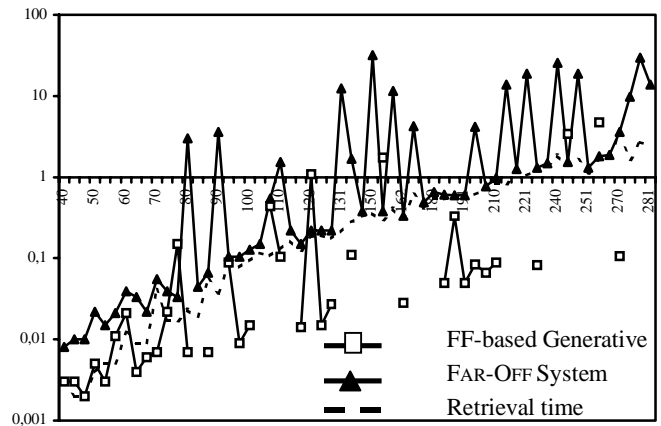
Since the purpose of the tests is to compare the performance of both generative and case-based planning processes, the FAR-OFF system is set to always consider a case to find a solution. It means that it does not use *a priori*, a heuristic to choose a planning process, generative or case-based, which may be applied to solve a specific problem faster. By the same reason, all case-bases created by CBS and considered in the tests are *not-Extensible*.

The results of the Blocks World tests show that the FAR-OFF system, sometimes, outperforms the FF-based generative planning. The FAR-OFF system solves all problems while the generative system solves only 62% of them. The original FF system solves 81% of the same problems (Bacchus 2000).

However, the possibility of solving all problems brings an adverse consequence to the FAR-OFF system: it finds longer solution plans than the FF-based generative planning.

BLOCKS WORLD	FF-based Generative		FAR-OFF					
			Solution		Retrieval		Case-Base	
	time	#steps	time	#steps	time	#fc	#cases	$\rho$
Problems								
BW-4-0	0,003	6	0,008	6	0,003			
BW-4-1	0,003	10	0,01	10	0,002	14	50	5
BW-4-2	0,002	6	0,01	6	0,002			
BW-5-0	0,005	12	0,022	18	0,004			
BW-5-1	0,003	10	0,015	14	0,005	37	150	7
BW-5-2	0,011	16	0,021	22	0,005			
BW-6-0	0,021	18	0,039	22	0,012			
BW-6-1	0,004	10	0,033	24	0,009	68	250	9
BW-6-2	0,006	20	0,022	30	0,009			
BW-7-0	0,007	20	0,055	22	0,041			
BW-7-1	0,022	22	0,039	42	0,017	126	350	11
BW-7-2	0,149	22	0,033	38	0,017			
BW-8-0	0,007	18	3,03	20	0,023			
BW-8-1			0,044	48	0,019	153	450	13
BW-8-2	0,007	16	0,066	32	0,054			
BW-9-0			3,63	76	0,037			
BW-9-1	0,088	28	0,104	32	0,081	222	550	15
BW-9-2	0,009	26	0,104	34	0,078			
BW-10-0	0,015	34	0,126	40	0,093			
BW-10-1			0,149	36	0,115	254	650	17
BW-10-2	0,44	34	0,55	34	0,108			
BW-11-0	0,104	34	1,54	38	0,128			
BW-11-1			0,22	40	0,164	249	750	19
BW-11-2	0,014	34	0,149	36	0,124			
BW-12-0	1,1	42	0,22	38	0,181	240	850	21
BW-12-1	0,015	34	0,22	42	0,188			
BW-13-0	0,027	42	0,22	48	0,174	321	950	23
BW-13-1			12,47	44	0,21			
BW-14-0	0,11	38	1,7	50	0,277	411	1050	25
BW-14-1			0,38	38	0,332			
BW-15-0			32,07	46	0,364	357	1150	27
BW-15-1	1,76	52	0,38	62	0,296			
BW-16-1			11,53	56	0,409	397	1250	29
BW-16-2	0,028	52	0,33	60	0,311			
BW-17-0			4,29	56	0,61	453	1350	31
BW-17-1			0,49	68	0,422			
BW-18-0			0,66	60	0,545	474	1450	33
BW-18-1	0,049	64	0,61	64	0,539			
BW-19-0	0,33	62	0,6	62	0,558	428	1550	35
BW-19-1	0,049	58	0,6	58	0,546			
BW-20-0	0,083	60	4,17	64	0,61	367	1650	37
BW-20-1	0,066	72	0,77	72	0,703			
BW-21-0	0,088	78	0,99	78	0,853	419	1750	39
BW-21-1			13,95	88	0,83			
BW-22-0			1,27	76	1,182	458	1850	41
BW-22-1			18,9	78	1,049			
BW-23-0	0,082	76	1,32	76	1,182	480	1950	43
BW-23-1			1,48	76	1,384			
BW-24-0			25,76	84	1,861	415	2050	45
BW-24-1	3,41	86	1,54	86	1,434			
BW-25-0			18,95	92	1,629	441	2150	47
BW-25-1			1,32	90	1,129			
BW-26-0	4,78	84	1,81	90	1,637	527	2250	49
BW-26-1			1,87	88	1,734			
BW-27-0	0,105	84	3,63	84	3,466	517	2350	51
BW-27-1			9,78	108	1,652			
BW-28-0			29,72	98	2,69	571	2450	53
BW-28-1			13,95	102	2,532			

**Table 1** – The results of the AIPS '00 Blocks World problems, the time to retrieve cases and some case-base features for each type of problem, as the number of footprint cases (#fc), the total number of cases (#cases) and the upper limit  $\rho$  used to determine the footprint cases. Blank spaces mean no solution was found.



**Figure 3** – Plotted results of the problems of Table 1. The axis Y is in logarithmic scale and represents the solution time in seconds.

The most important feature of the case-based approach is that, even using a random case-base, it solves all problems in the Blocks World tests in an acceptable amount of time.

The other domain considered in the tests is the Logistic domain, in which the FF system does not have any difficulty to work (Hoffmann and Nebel 2001). This domain was chosen in order to confront the FAR-OFF system with a domain where the FF's heuristic is usually close to optimal and guides the generative system search to the solution very quickly, as reported in (Hoffmann and Nebel 2001).

In fact, by the results in Table 2, the FF-based generative planning works well and performs better than the FAR-OFF system.

However, the time performance of the FAR-OFF system is close to the FF-based generative system, and the time to retrieve cases is less than the time to find a solution from scratch in most of situations. It means that if a problem recurs, the FAR-OFF system will solve the problem faster, because no adaptation will be necessary. The recurrence of a problem is one tenet about the nature of the world that case-based reasoning techniques are based.

An overall analysis shows that the main advantage of the FAR-OFF system is the possibility to have more than one chance to solve a problem. It is provided by the number of retrieved cases, if one retrieved case fails to be adapted, the system can try another one.

## Discussion

The FAR-OFF system is a complete domain independent system. It differs from hand-tailored planning systems (Bacchus 2000) because the additional knowledge used by it is automatically generated and is independent of the domain, while the additional knowledge applied in hand-tailored planning systems is specific to domain knowledge.

This first implementation of the FAR-OFF system is modeled in the STRIPS version. However, its natural evolution to the ADL version seems to be quite simple.



