

## An Accurate Adaptation-Guided Similarity Metric for 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  
05508-900 - São Paulo - SP - Brazil

<sup>2</sup> Faculdade de Engenharia Industrial  
Av. Humberto de A. Castelo Branco, 3972  
09850-901 - São Bernardo do Campo - SP - Brazil  
e-mails: flavio@lac.usp.br ; rillo@lsi.usp.br  
phone: +55 11 3818 5530

**Abstract.** In this paper, we present an adaptation-guided similarity metric based on the estimate of the number of actions between states, called ADG (*Action Distance-Guided*). It is determined by using a heuristic calculation extracted from the heuristic search planning, called FF, which was the fastest planner in the AIPS'2000 competition. This heuristic provides an accurate estimate of the distance between states that is appropriated for similarity measures. Consequently, the ADG becomes a new approach, suitable for domain independent case-based planning systems that perform state-space search.

### 1 Introduction

The purpose of Case-based Reasoning (CBR) systems is to improve efficiency by using stored cases to solve problems. However, there are many efficiency *bottlenecks* in CBR systems. Two of them, which are concerned with the retrieval phase, are: the space of cases where the search engine is performed, and the similarity metric that is responsible for determining how much similar the cases are to a new problem. The researches in former topic are concerned with locating the best case in a reduced space of search [13,15]. In the latter, the researches are concerned with how to design a suitable similarity metric that provides an accurate measure. Better approaches for this topic have been the measurement of the similarity of cases based on adaptation effort [8,11,16]. The less effort, the better is the case to be retrieved.

This paper focuses on developing a similarity metric that may predict the adaptation effort of a case by the estimate of the distance between states in case-based planning (CBP) systems. We mean by CBP a general problem solver where plans are a sequence of actions. It is different from route planning [14] and episode-based planning [8], which are particular samples of CBP systems, and consequently, they have specific features that can be explored in order to improve the system efficiency.

---

\* This work is supported by FAPESP under contract no. 98/15835-9.

As the plans are a sequence of actions, a suitable similarity can be found through the distance given by the possible number of actions between states. However, the similarity metrics, until now, would find a similarity by the number of differences between states [7,19].

The ADG (*Action Distance Guided*) similarity here proposed is a new similarity metric based on the estimate of the number of actions. It is extracted from the heuristic used by the FF planning system [4,6] to estimate the number of actions between the current state and the final state. The FF was the fastest planner in the AIPS'2000 competition [1] by using this heuristic method to guide the search engine at each step.

The proposed similarity metric is suitable to be applied in conjunction with any method that reduces the search space of cases, as [15], or with in any method of case-base maintenance based on the adaptation effort. For example, the ADG similarity can be used in RP-CNN [9], which is a measure based on the adaptation costs. It needs a suitable similarity metric to edit case-bases and to preserve the efficiency of the system. The RP-CNN assumes "*that the similarity metric will accurately select the most adaptable case for any problem*" ([9] p. 166), and the ADG can be this similarity metric.

This paper is organized as follow. Section 2 presents some related works. Section 3 presents the Transaction Logic that is used to formalize actions, plans and cases. Section 4 presents the relaxed fixpoint extracted from [4] and defines the ADG similarity. Section 5 presents some experiments. Finally, some discussions are presented in section 6 and conclusions in section 7.

## 2 Related Works

Although the case-base competence is important, the efficiency is really the main factor that affects the performance in environments where case-based systems have intensive domain knowledge, which can provide a solution *from scratch*.

The retrieval phase critically affects the systems performance. It must search in a space of cases in order to choose a proper one that will allow the system to solve a new problem easily. In order to improve efficiency in the retrieval phase, it is necessary either to reduce the search space or to design an accurate similarity metric. Reducing the search space will let available only a suitable subset of cases for the search process and an accurate similarity metric will choose the most similar case to decrease the adaptation phase effort. Some works present interesting approaches that reduce the search space of cases (e.g [13,15]).

In this paper, we are concerned with similarity metric. Much attention has been given to researches that design suitable similarity metrics. They focus on choosing the most adaptable case as the most similar one, such as DIAL [8], DéjàVu [16] and ADAPtER [11] systems. DIAL system is a case-based planner that works in disaster domains where cases are schema-based episodes. It uses a similarity assessment approach, called RCR, which considers an adaptability estimate to choose cases in the retrieval phase. Similarly, Déjà Vu system operates in design domains and uses an adaptation-guided retrieval (AGR) to choose cases that are easier to be adapted.

ADAPtER is a diagnostic system that calculates an estimate of adaptability of a solution to supply a pivoting-based retrieval with the capacity to choose a group of relevant cases between an upper and a lower bound.

The adaptation-based metric presented by this paper, called ADG (*Action Distance-Guide*), differs from those presented in ADAPtER and DéjàVu systems. It is designed to be applied in case-based planning systems that operate with actions. The action-based planning is a general problem solver that finds a sequence of actions to transform the initial state into a desirable final state [4,6,7,12,17,18,19]. This approach is slightly different from route planning [14] and episode-based planning [8], which are specific planners and have specific features that can be used to improve the efficiency of the system.

The ADG similarity differs from the AGR approach because it does not use any domain knowledge besides those obtained from actions and states, which are the minimal knowledge that is required to define a domain for planning systems. The AGR approach in DéjàVu system uses additional domain knowledge, called *capability knowledge*, which is similar to what *critics* used to determine and solve conflicts in partial-order planning systems. This additional knowledge allows identifying the type and the functionality of a set of transformations, which are performed by actions, through a collection of *agents* called *specialists* and *strategies*. It must be well specified in order to maximize the AGR performance.

The ADG similarity differs from RCR method, used by DIAL system. It is based on domain knowledge that is available in action definitions, while the RCR method uses the experience learned from the adaptation of previous utilization of cases. They also differ in their applicability because the RCR method considers cases as episodes in disaster domains; the ADG approach considers it as a sequence of actions, suitable for domain independent planning systems.

There are many domain independent case-based planning systems (e.g. [2,7,10,19]). A great number of them use a search engine based on a space of states [7,19]. Most of the state-space planning systems use a similarity rule that only confronts ( $I$ ) - the initial state and ( $G$ )- the final state of a stored case with ( $I'$ ) - current state and ( $G'$ ) - goal state of the new problem. If  $I$  subsumes  $I'$  and  $G$  subsumes  $G'$  under some limits, the case is appropriate to be retrieved. All retrieved cases are then ordered following this subsumption. However, this approach is not a suitable measure to improve the adaptation phase efficiency because it is not a good measure of the real similarity.

An alternative approach to planning with states is the plan-space planning systems [2,10]. They search in a space of plans and they have no goals, but only tasks to be achieved. Since tasks are semantically different from goals, the similarity metric designed for plan-space based CBP systems is also different from the similarity rules designed for state-space based CBP systems.

An interesting similarity rule in plan-space approach is presented in the CAPLAN/CBC system [10]. It extends the similarity rule introduced by the PRODIGY/ANALOGY system [19] 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. This approach is similar to RCR method used by DIAL system in disaster domains.

There are two important differences between ADG and CAPLAN/CBC's similarity rule. One difference is that the former is designed for state-space planning and the latter for plan-space planning. Another difference is that the ADG similarity does not need to learn any knowledge to present an accurate estimate. As highlighted before, the ADG similarity only needs the knowledge that can be extracted from actions.

The purpose of this paper is, therefore, to develop an adaptation-based similarity rule that estimates the adaptation effort of each case for general and domain independent case-based planning systems with state-space search.

### 3 The Transaction Logic in Planning

The Transaction Logic (TR) [3], in its serial-Horn version, is an extension of the first-order logic, by the introduction of the serial conjunction operator ( $\otimes$ ), e.g.,  $\alpha \otimes \beta$ , which means "first execute  $\alpha$ , and then execute  $\beta$ ".

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  $\phi$ . 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  $D_k$ .

To perform an updating, TR works with a transition oracle ( $O^t$ ) that is a mapping between two states and a set of atomic formulas. For our purposes, it is only necessary to define two formulas:  $ins(\_)$  and  $del(\_)$  in the transition oracle framework. To formally execute both formulas, for example  $ins(c)$  and  $del(d)$ , a possible satisfaction can be:  $P, D, D_1, D_2 \models ins(c) \otimes del(d)$ . It happens if and only if  $ins(c) \in O^t(D, D+\{c\})$ ;  $del(d) \in O^t(D, D-\{d\})$ ;  $D_1 = D+\{c\}$  and  $D_2 = D+\{c\}-\{d\}$ .

With all these formal features, TR is a suitable logic to describe actions and plans for planning systems, as can be seen with some uses of TR in planning formalization [12,17,18]. As TR is suitable for planning, it is suitable for CBP system components as well, because the case memory is a collection of plans.

In order to define the planning components in a TR framework, we will follow in accordance with Santos and Rillo's work [12]. 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 literal.*

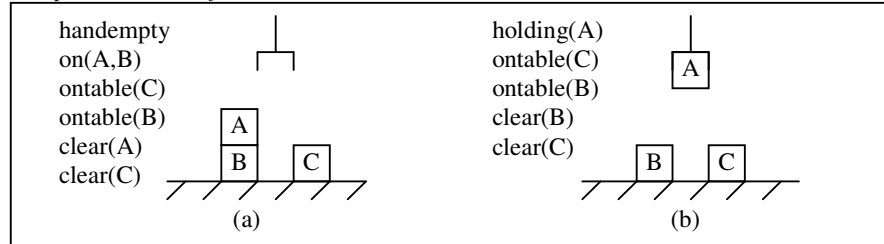
Two examples of state definition are given in Figure 1. States can be modified by actions, which are defined as:

**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  $\alpha$ .



**Fig. 1.** Two examples of state definition in block-world domain.

- $delete(\alpha)$  is the delete list of the action  $\alpha$ . It is a TR formula in accordance with transition oracle and it represents all predicates that will not be true after the action execution.
- $add(\alpha)$  is the add list of the action  $\alpha$ . It is a TR formula in accordance with transition oracle and it represents all predicates that 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 inserting of the delete and add lists respectively. Any action  $\alpha$  just can be executed from a state  $D$  if  $pre(\alpha) \subseteq D$ . For example, consider the following action in the well-known blocks-world domain:

$$unstack(x,y) \leftarrow qry(handempty(x)) \otimes qry(clear(x)) \otimes qry(on(x,y)) \otimes del(clear(x)) \otimes del(handempty) \otimes del(on(x,y)) \otimes ins(holding(x)) \otimes ins(clear(y)).$$

which means to get the block  $x$  from the top of block  $y$ . We can observe that:

$$\begin{aligned} pre(unstack(x,y)) &= qry(handempty(x)) \otimes qry(clear(x)) \otimes qry(on(x,y)). \\ delete(unstack(x,y)) &= del(clear(x)) \otimes del(handempty) \otimes del(on(x,y)). \\ add(unstack(x,y)) &= ins(holding(x)) \otimes ins(clear(y)). \end{aligned}$$

Consider now a state  $D_0$  as the state described in Figure 1a and the state  $D_f$  as the state described in Figure 1b. Thus,  $unstack(A,B)$  can be satisfied in TR as:

$$P, D_0 \dots D_f \models unstack(A,B)$$

Where  $P$  is the set of actions  $A$  and  $\langle D_0 \dots D_f \rangle$  is the state path from  $D_0$  to  $D_f$  by the application of  $del$  and  $ins$  predicates contained in formula  $unstack(A,B)$ . Therefore, TR permits to perform formally an action in planning.

With the definitions of actions and state, it is possible, following [17], 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. A case is a modified plan by the insertion of 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 pos-condition of the case.

Intuitively,  $Wi$  is a set of those literals that are deleted by the plan. It is equal to the result of the foot-printing method used by PRODIGY/ANALOGY system [19] and it represents the pre-condition of the plan. The process of foot-printing the initial state is described in [19].

With respect to  $Wf$ , it is a set of those literals that are inserted by the plan and will be presented in the final state after the plan execution. There is no correspondence to other cases features in any case-based planning system.

## 4 The Similarity Rule

Once we have  $Wi$  and  $Wf$ , we will be able to define values of similarity between a new problem, described by a goal formula (def. 4), and a case. It can be made through two comparisons. One is with the current initial state, denoted by  $D_0$ , and with the initial state features from the case, denoted by  $Wi$ . This comparison is called *initial similarity value* ( $\delta_i$ ). Another comparison is between the desirable final state ( $Df$ ) from the goal and the final state features ( $Wf$ ) from the case. This comparison is called *goal similarity value* ( $\delta_G$ ). Both similarity values are obtained after the determination of a state-fixpoint from a relaxed planning problem.

### 4.1 The State-Fixpoint

The process of estimating the distance between two states is the main part of a heuristic-search planning systems, as FF [4,6]. In fact, we use the same heuristic, which is used by FF planner to guide the search, to estimate the similarity between the current situation and stored cases.

The first step of heuristic determination is to create a graph for a relaxed planning problem. Hoffmann and Nebel [6] explain that this relaxing is obtained by ignoring the delete list of actions. It allows the graph to find a relaxed fixpoint that is composed by all predicates that are reached from the initial state.

As defined by Hoffmann [4], 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 predicates 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.

These layers show in how many steps, at least, a predicate can be reached, or an action can be applied, from  $D_0$ . The algorithm in Figure 2 shows this process, and it can be performed as an off-line process or as a startup process of the retrieval phase. It can be off-line because it uses only information from the initial state ( $D_0$ ) and the actions that are available before the retriever starts. In addition, it is performed only once, so it can be performed as a startup process of the retrieval phase.

Some useful information can be determined from the relaxed fixpoint process. Following [4,6], they are:

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

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

**Definition 8:**  $difficulty(\alpha) := \sum_{d \in pre(\alpha)} levelpred(d)$

The definitions 6 and 7 provide the order number of the layer where each literal or action appears first. It means that each literal, or action, is a membership of its first appearance in the fact layer or in the action layer respectively.

The definition 8 presents a heuristic that calculates the difficulty of an action in respect of its first preconditions appearance. As highlighted in Hoffmann and Nebel's work, "*this heuristic works good in situations where there are several ways to achieve one fact, but some ways need less effort than others.*" [6].

With the graph generated, it is possible to find a relaxed solution for any state that can be reached from  $D_0$ . This relaxed solution, improved by the difficulty measure, provides an estimate for the optimal length of the not-relaxed solution [6]. This relaxed solution will give us the *initial similarity value* ( $\delta_i$ ) and the *goal similarity value* ( $\delta_G$ ).

#### 4.2 The Initial Similarity Value

For a stored case, one important measure that must be estimated is the distance between the initial state ( $D_0$ ) and the features of the case initial state ( $W_i$ ). This distance estimation can be obtained by the second step of the FF's estimate heuristic: the relaxed solution [6].

The process of obtaining the relaxed solution for a target-state ( $W_i$ ) from initial state ( $D_0$ ), as described in [6] uses the generated graph. First, each predicate in the target-state is initialized as a goal in its correspondent layer, determined by  $levelpred(\_)$  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.

---

```
F0 := D0;
k:=0;
fixpoint:=False;
while fixpoint=False do
    Ok := {α ∈ A | pre(α) ⊆ Fk}
    Fk+1 := ∪α ∈ ok add(α)
    if Fk+1 = Fk then fixpoint:=True;
    k:=k+1;
endwhile
max:=k;
```

---

**Fig. 2.** The algorithm that computes the relaxed fixpoint from the initial state. It is extracted from [4].

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

The algorithm to compute the relaxed solution is shown in Figure 3. The variable  $h$  is used to count the number of selected actions. The initialization of each goal is made by an array and each predicate has a mark that can be true or false.

With the algorithm in Figure 3, it is possible to find the *Initial Similarity Value*, that is the result  $h$  from the function *relaxed\_initial\_length* ( $W_i$ ) after *setting all marks of all predicates as false*:

$$\delta_i = h$$

The value of  $\delta_i$  is the estimate of the distance between  $D_0$  and  $W_i$ , and configures the first part of our similarity metric. The second part is obtained by the estimate of the distance between  $W_f$  and  $D_f$ , which is presented below.

### 4.3 The Goal Similarity Value

So far, the similarity measure only used the FF's estimate heuristic without any changes in its original concept. In fact, we had needed only to calculate the distance from  $D_0$ , and the direct application of the FF's heuristic is enough. Now, we have to

introduce some modifications in the FF's heuristic concept in order to calculate the distance from  $Wf$  to  $Df$  using the generated graph.

The generated graph is suitable to estimate the distance from  $D_0$ . In this case, the distance between  $Wf$  and  $Df$  cannot be estimated by simply calculating the distance of  $Df$  from  $D_0$  and then diminishing the number of actions in the case and the  $\delta_i$  value. This is because the solution trace from  $D_0$  is different from the other solution traces that consider the actions in the case. It leads us to force the solution trace from  $D_0$  to use the sequence of actions in the case.

---

```
relaxed_initial_length(G)  
  
clear all Gi  
for i:= 1 ... max do  
  Gi := {g ∈ G | levelpred(g) = i};  
endfor  
h:=0;  
for i:= max ... 1 do  
  for all g ∈ Gi, g not marked TRUE at time i do  
    select  $\alpha_{\text{levelpred}=i-1}$ ; minimal difficulty; g ∈ add( $\alpha$ );  
    h:=h+1;  
    for all  $d_{\text{levelpred} \neq 0} \in \text{pre}(\alpha)$ , not marked True at i-1 do  
       $G_{\text{levelpred}(d)} := G_{\text{levelpred}(d)} \cup \{d\}$ ;  
    endfor  
    for all d ∈ add( $\alpha$ ) do  
      mark d as True at time i-1 and i;  
    endfor  
  endfor  
endfor  
return h;
```

---

**Fig. 3.** The algorithm that computes the relaxed solution from a relaxed fixpoint, where G is the target-state. It is extracted from [6].

To force the estimate of the distance of  $Df$  from  $D_0$  considering the case actions, it is necessary to maintain the values of each mark of each predicate, after the performing of the *Initial Similarity Value* calculation. It means that we will call the function *relaxed\_final\_length(Wi,Wf,Df)* (Figure 4) using the marks changed by the *relaxed\_initial\_length(Wi)*.

However, keeping the marks unchanged is not enough. It is also necessary to change some truth predicate values of  $Wi$  and  $Wf$ . First, we must set all marks of all predicates in  $Wi$  as false. Then, we have to mark as true each predicate in  $Wf$  in their correspondent layer. All these changes are necessary because  $Wi$  indicates the predicates that the case will delete, and the  $Wf$  indicates the predicates that will be true after the case execution.

In addition, it is also necessary to initialize  $Wi$  as a goal, because the trace must be calculated through the actions of the case.

The reason to keep unchanged all marks from  $\delta_i$  calculation is to avoid that the calculation of the *Goal Similarity Value* incorporates redundant values like the number of actions between  $D_0$  and  $W_i$ .

With the result  $h'$  from the function *relaxed\_final\_length(Df)* of the algorithm presented in Figure 4, we can define the *Goal Similarity Value*, that is the second part of our similarity metric:

$$\delta_F = h'$$

With all parts defined, it is possible to determine the similarity value of a case.

---

```
relaxed_final_length (Wi, Wf, Df)

G:=Df;
for each d  $\in$  Wi do
    mark d as False at all levels;
    G:=G+{d};
endfor
for each d  $\in$  Wf do
    mark True at levelpred(d) and levelpred(d)-1
endfor
h':=relaxed_initial_length(G);
return h';
```

---

**Fig. 4.** The algorithm that extracts the distance between  $Wf$  and  $Df$  by considering the marks from *relaxed\_initial\_length(Wi)*.

#### 4.4 The Action Distance-Guided Similarity

Finally, the similarity metric can be defined joining the two values:  $\delta_i$  and  $\delta_F$ . This similarity metric, that we call *Action Distance-Guided (ADG)*, is a measure of the adaptation effort for constructing a plan from  $D_0$  to  $W_i$  and from  $W_f$  to  $D_f$ . It is the sum of both values:

$$ADG = \delta_i + \delta_F$$

It is important to note that 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.

A case is useful when the ADG value is less than the direct distance between  $D_0$  and  $D_f$ , that can be calculated with the FF's heuristic. If this distance is less than the ADG value of any stored case, a generative planner, as FF [4,6], can be performed without the use of any retrieved case.

Otherwise, if the ADG value is less than the direct distance between  $D_0$  and  $D_f$ , it is preferable to use a retrieved case than plan *from scratch*. It means that, for example, the ADG value of a case with 10 actions is 7 and the distance between  $D_0$  and  $D_f$

directly is 15. It is preferable to avoid any kind of modification in the case structure and to plan a sequence of 7 actions, and finding a solution with 17 actions, than plan *from scratch* and have to find a sequence of 15 actions. This preference is based on the generative planner effort, which is less if it is requested to find a sequence of 7 actions than a sequence of 15 actions.

Some CBP systems incorporate a modification phase that can change the actions in the case in order to find a solution near to optimal. However, this process is more time expensive than the approach that does not perform any modification in the case structure. Therefore, the generative planning is used to *complete* the retrieved case by finding a sequence of actions that links  $D_0$  to the case and another sequence that links the case to  $D_f$ . It is considered in the experiments below.

## 5 Experiments

Some empirical tests are necessary in order to analyze the performance and results that the ADG similarity can produce. The experiments are performed with two case-bases with 100 cases each, in two different domains: Block-world and Logistic domains. The former is a well know domain in the planning area, and the latter is a simplified version of UCPOP domain. Both were used in AIPS'2000 competition [1].

In order to compare the results from ADG, we will consider the similarity rules used by some case-based planning systems, such as MRL's subsuming similarity [7] and Prodigy's footprint similarity [19]. We will also consider the standard normalized Nearest Neighborhood (NN) rule as a generic similarity rule.

However, the NN rule and MRL's subsuming similarity used in the tests are improved by  $W_i$  and  $W_f$  information from the stored cases. It means that the accuracy improvement obtained by Prodigy's footprint similarity over the standard Nearest Neighborhood does not exist anymore. It happens because the improvement was mainly based on the consideration of information about initial and goal states of the cases, and this information is already described in  $W_i$  and  $W_f$ . On the other hand, the time used by Prodigy's similarity to compute the footprint initial states, which takes  $O(n^2)$  for a case with  $n$  actions [19], is not necessary because  $W_i$  is already computed in all stored cases. Consequently, Prodigy's similarity takes the same processing time as MRL's subsuming and standard Nearest Neighborhood.

Considering the fact that Prodigy's similarity rule does not calculate normalized values of similarities, and that MRL's subsuming does not consider the case initial state features to rank similar cases, it is expected that the normalized standard NN performs better than both in the experiments, but worse than AGR.

The experiments are performed in a *Pentium*<sup>®</sup> III computer with 450 MHz and 128 Mbytes of memory, in *Microsoft*<sup>®</sup> *Windows* environment.

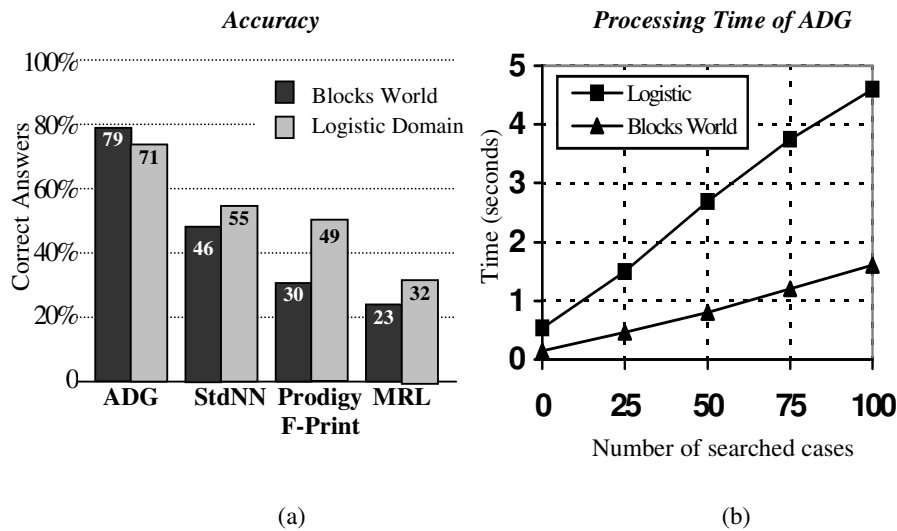
### 5.1 Experiment 1 – The accuracy of each similarity rule

In the first experiment (Figure 5a), each test considers a couple of cases. We performed 600 tests with couple of different cases in different initial and final states.

Initial states, final states and cases are chosen randomly for each case base in blocks and logistic domains.

At each test, the time used by a generative planning to *complete* the cases is annotated. The case that requested less effort to find a solution is considered the most similar one. The cases are considered equally similar if the difference time between both is less than 0.3 seconds. Then, each similarity rule is applied. The results of each method are confronted with the correct answer. The percentage of correct results from each similarity rule is presented in Figure 5a.

The results show that the ADG is better than the common similarity rules used by some CBP systems. It leads us to conclude that ADG is suitable for any adaptation-guided similarity rule applied in CBP systems with state-space search.



**Fig. 5.** The performance of the ADG compared with other methods by accuracy (a) and its processing time (b).

However, this accuracy is obtained to the detriment of processing time, as we can see in the processing time experiment.

## 5.2 Experiment 2 – Processing Time

This experiment shows the processing time behavior of ADG similarity in different domains. As the ADG is a process that calculates a relaxed solution, it is more time expensive than the other methods, which take a processing time less than 0.3 second for a case-base with 100 cases.

The purpose of this experiment is to verify if the time curve from ADG similarity calculation presents any exponential average time behavior that can invalidate the benefit obtained by the similarity accuracy. The curve is plotted in Figure 5b.

The test is performed by determining the processing time to calculate the ADG value for different quantities of cases in each domain. The result is an average time value.

The result of the tests lead us to conclude that the ADG similarity calculation presents a linear behavior in the average, even though it varies from domain to domain.

The difference in processing times occurs because domains have different complexities. Any generic system presents different processing times for different complexities, because it does not have any specific domain features or any particular heuristic methods that can improve the system efficiency in such domains.

For being a generic similarity rule, ADG presents the same different processing times for different domains like any domain independent generative planner. Therefore, we claim that the time expended to find a similar case in the case-base is proportional to that presented by a generic generative planner as a result from planning from scratch. The accuracy of the ADG can provide efficiency to the system by the fact that the time wasted in the retrieval phase can be compensated in the adaptation phase.

The efficiency of the system can be improved by a combination of an accurate similarity rule, such as ADG, and a method that reduces the space of cases in the retrieval searching process [15].

## 6 Discussion & Future Works

The actions defined in section 3 are STRIPS-like. However, as the FF's heuristic can also be calculated from actions in ADL-style [6], our similarity metric can be applied to ADL-style as well.

The meaningful improvement introduced by the ADG similarity metric is the use of FF's heuristic that calculates a solution length, not necessarily optimal, for the relaxed problem. It provides an informative estimate of the difficulty for planning. This heuristic is well used by FF planning to escape from local minima and plateaus in a local search engine combined with a breadth first search [4,5,6]. The use of this heuristic permits that the FF works well in most of planning domains, as shown by the results in AIPS'2000 competition [1]. An excellent and deep analysis of the reasons that lead the FF to work well in planning domains is presented in [5].

As ADG is an estimate of the difficulty in planning to complete the retrieved case in order to solve a problem, it is suitable for systems that have the purpose to use the case without any or with less modification. The idea is to retrieve a case that can decrease the number of actions that the system has to find and arrange to solve the problem.

For a little group of cases, it is possible to investigate the intermediary states of each case, and decide if any modification, like adding, deleting or replacing actions, can be performed. The ADG similarity can be used to estimate the distance between

the current initial state and these intermediary states, and consequently, it can discover if there are superfluous actions and states in a case composition. If they exist, they can be eliminated.

However, only ADG similarity is not sufficient to perform a modification phase, because this phase can become a time expensive process, and in most situations, some of heuristic-search planning systems, such as FF planner, could be faster than it. Therefore, other heuristic methods should be researched in order to perform the modification phase efficiently.

The ADG similarity is suitable to be used in CBP systems with state-space search, and it cannot be applied in plan-space approaches as it is presented in this paper. However, with some modifications, the ADG similarity could operate with tasks instead of states and be appropriate to be used in domain independent case-based planning with plan-space search. In addition, it can be mixed with techniques of learning and feature weights, as CAPLAN/CBC's similarity rule. All these advances will be left for future works.

## 7 Conclusions

This paper presents a similarity metric for case-based planning systems based on the number of actions necessary to transform one state into another state. We called this similarity rule ADG (*Action Distance-Guided*).

The ADG similarity estimates the distance among the stored case and the initial and final states. Consequently, the ADG similarity presented a better accuracy than the usual similarity rules applied in generic case-based planning.

Although the processing time is higher than in other methods, the retrieved case is much more accurate, and consequently, it will require less effort from the adaptation phase.

As far as the processing time is concerned, many methods can reduce the space search of cases in the case-base, reducing the processing time [15]. As mentioned before, this paper is concerned with the accuracy of the similarity rule, which, if applied with any reducing space search method, can provide a CBP system with a fast and accurate retrieval phase.

## References

1. Bacchus, F. AIPS-2000 Planning Competition Results. Available in: <http://www.cs.toronto.edu/aips2000/>.
2. Bergmann, R., Wilke, W. Building and Refining Abstract Planning Cases by Change of Representation Language. *Journal of Artificial Intelligence Research*, 3. (1995). 53-118.
3. Bonner, A.J., Kifer, M.: Transaction logic programming. Technical Report, CSRI-323, Department of Computer Science, University of Toronto (1995).
4. Hoffman, J. A Heuristic for Domain Independent Planning and its Use in an Enforced Hill-climbing Algorithm. In: *Proceedings of 12<sup>th</sup> International Symposia on Methodologies for Intelligent Systems*. North Carolina, USA. (2000).

5. Hoffman, J. Local Search Topology in Planning Benchmarks: An Empirical Analysis. In: Proceedings of the 17<sup>th</sup> International Joint Conference on Artificial Intelligence IJCAI'01. Morgan Kaufmann Publishers (2001).
6. Hoffman, J., Nebel, B. The FF Planning System: Fast Plan Generation Through Heuristic Search. In: Journal of Artificial Intelligence Research. To appear.
7. Koehler, J. Planning from Second Principles. Artificial Intelligence, 87. Elsevier Science. (1996).148-187.
8. Leake, D., Kinley, A., Wilson, D. Case-Based Similarity Assessment: Estimating Adaptability from Experience. In: Proceedings of 14<sup>th</sup> National Conference on Artificial Intelligence – AAAI'97. AAAI Press. (1997).
9. Leake, D., Wilson, D. Remembering Why to Remember: Performance-Guided Case-Base Maintenance. In: Blanzieri,E., Portinale, L. (Eds.) Proceedings of the 5<sup>th</sup> European Workshop on Case-Based Reasoning (EWCBR2K). Lecture Notes in Artificial Intelligence, Vol 1898. Springer-Verlag. (2000) 161-172.
10. Muñoz-Avila, H.,Hüllen, J. Feature Weighting by Explaining Case-Based Planning Episodes. In: Smith, I., Faltings, B. (Eds) Proceedings of 3<sup>rd</sup> European Workshop on Case-Based Reasoning (EWCBR'96). Lecture Notes in Artificial Intelligence, Vol 1168. Springer-Verlag. (1996) 280-294.
11. Portinali, L., Torasso, P., Magro, D. Selecting Most Adaptable Diagnostic Solutions through Pivoting-Based Retrieval. In: Leake,D.,Plaza,E. (Eds) Proceedings of the 2<sup>nd</sup> International Conference on Case-Based Reasoning – ICCBR'97. Lecture Notes in Artificial Intelligence, Vol 1266. Springer-Verlag. (1997). 393-402.
12. Santos, M., Rillo, M. Approaching the *Plans are Programs* Paradigm using Transaction Logic. In: Steel, S., Alami,R (Eds) Proceedings of 4<sup>th</sup> European Conference on Planning – ECP'97. Lecture Notes in Artificial Intelligence, vol. 1348. Springer-Verlag. (1997) 377-389.
13. Schaaf, J. Fish and Shrink: A Next Step Towards Efficient Case Retrieval in Large-Scale Case-Bases. In: Smith, I., Faltings, B. (eds): Advances in case-Based Reasoning. Lecture Notes in Artificial Intelligence, Vol 1168. Springer-Verlag. (1996) 362-376.
14. Smyth, B. , McGinty, L. Personalised Route Planning: A Case-Based Approach. In: Blanzieri,E., Portinale, L. (Eds.) Proceedings of the 5<sup>th</sup> European Workshop on Case-Based Reasoning (EWCBR2K). Lecture Notes in Artificial Intelligence, Vol 1898. Springer-Verlag. (2000).431-442.
15. Smyth, B., McKenna, E. Footprint-Based Retrieval. In: Althouff, K., Bergmann, R., Branting, K. (Eds.) Proceedings of the 3<sup>rd</sup> International Conference in Case-Based Reasoning. ICCBR'99. Lecture Notes in Artificial Intelligence, Vol 1650. Springer-Verlag. (1999).343-357.
16. Smyth, B., Keane, M. Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning. In: Journal of Artificial Intelligence, 102(2). (1998). 249-293.
17. Tonidandel, F, Rillo, M. Case-Based Planning in Transaction Logic Framework. In: Proceedings of Workshop on Intelligent Manufacturing Systems (IMS'98). Elsevier Science. (1998). 281-286.
18. Tonidandel, F., Rillo, M. Handling Cases and the Coverage in a Limited Quantity of Memory for Case-Based Planning Systems. In: Sichman, J., Monard, C. (Eds). Proceedings of IBERAMIA/SBIA 2000. Lecture Notes in Artificial Intelligence, Vol 1952. Springer-Verlag. (2000) 23-32.
19. Veloso, M. Planning and Learning by Analogical Reasoning. Lecture Notes in Artificial Intelligence, Vol 886. Springer-Verlag. (1994).