Feature Weighting by Explaining Case-Based Problem Solving Episodes

Héctor Muñoz-Avila & Jochem Hüllen
University of Kaiserslautern, Dept. of Computer Science
P.O. Box 3049, D-67653 Kaiserslautern, Germany
E-mail: {munioz|huellen}@informatik.uni-kl.de

Abstract

We present a similarity criterion based on feature weighting. Feature weights are recomputed dynamically according to the performance of cases during problem solving episodes. We will also present a novel algorithm to analyze and explain the performance of the retrieved cases and to determine the features whose weights need to be recomputed. We will perform experiments and show that the integration in a feature weighting model of our similarity criterion with our analysis algorithm improves the adaptability of the retrieved cases by converging to best weights for the features over a period of multiple problem solving episodes.

1 Introduction

An essential factor influencing the effectiveness of case-based problem solving is the retrieval phase (Aamodt and Plaza, 1994). In the context of planning and design, retrieval means searching for adaptable cases (Smyth and Keane, 1994; Marir and Watson, 1995; Smyth and Keane, 1995). Thus, any similarity criterion should measure the adaptation effort of the cases with respect to a new problem. Because the adaptation effort is difficult to determine *a priori*, learning from previous retrieval episodes has been proposed (Veloso, 1992; Fox and Leake, 1995; Ihrig and Kambhampati, 1995).

In *Robbie* (Fox and Leake, 1995), introspective reasoning (Leake et al., 1995) is used to determine the features that should be considered during retrieval. The validity of pre-defined assertions related to indexing criteria is tested after each retrieval episode. If a failure occurs, the criteria is restated resulting in a refinement of the index.

In the context of domain independent planning, EBL (Minton, 1988) has been used in a system called *derSNLP+EBL* (Ihrig and Kambhampati, 1995) to *explain* retrieval failures. An EBL-rule that indicates combinations of features causing a failure is constructed. The rule is used as a filter to avoid selecting the case when a given problem matches the indicated combination of features.

Another approach toward learning from retrieval failures in the context of domain independent planning was proposed in PRODIGY/ANALOGY (Veloso, 1992; Veloso, 1994a): each feature is assigned a so-called relevance-bias, which is a number indicating the preference of the feature. The relevance-bias of a feature in the case is incremented dynamically when a failure occurs and the feature was not matched in the new problem. PRODIGY/ANALOGY divides features into relevant and non relevant. The *foot-printed similarity metric* counts features of the problem matched by relevant features in the cases. The relevant features are indexed so that features with higher relevance-bias are matched first.

In this paper we will extend PRODIGY/ANALOGY's similarity metric by explicitly incorporating feature weights in a new similarity metric, *the weighted foot-printed similarity metric*. The weighted foot-printed similarity metric counts the weights of relevant features matching features in the new problem and can be integrated with a feature weighting model. We will a also present a novel algorithm to analyze and explain the performance of the retrieved cases. This algorithm serves as a bridge between the similarity metric and the weighting model. We will evaluate our approach with experiments and will conclude that the integration in the feature weighting model of the weighted foot-printed similarity metric with the algorithm improves the adaptability of the retrieved cases by *converging* to best weights for the relevant features over a period of multiple problem solving episodes.

This paper is organized as follows: the next section presents the weighted foot-printed similarity metric and a model for feature weighting. Section 3 presents the algorithm for analyzing and explaining the performance of the cases retrieved. Then, the results of the experiments performed are shown. Limitations of our work are discussed in section 5. Finally, we make concluding remarks of our work in the last section.

2 Weighting Relevant Features

The foot-printed similarity metric compares relevant features of a case with features of a new problem. Informally, a feature is considered relevant to a goal with respect to a solution, if the feature contributes to achieve the goal in the solution (Veloso, 1994b). This notion can be illustrated with an example in the logistics transportation domain (Veloso, 1994a). A typical problem in this domain is, starting from a certain configuration of objects, locations, and transportation means, to place the objects at different locations. There are different sorts of locations and means of transportation. In addition, the means of transportation have certain operational restrictions. For example, a truck can only be moved between places located within the same city.

Figure 1 (a) shows an example of a possible configuration in this domain describing an initial state. The final state, shown in figure 1 (b), consists of one goal: the object *obj1* must be placed at the post office *post2*. Figure 2 shows a possible solution (i.e., a plan). Continuous boxes represent plan steps and continuous lines represent the order of execution among the plan steps. Dashed boxes represent goals (in figure 2, only one -$sameCity(post1, post2)$- is shown, which corresponds to a precondition of the plan step $move(truck1, post1, post2)$.). Notice that the *truck2* was not used for achieving

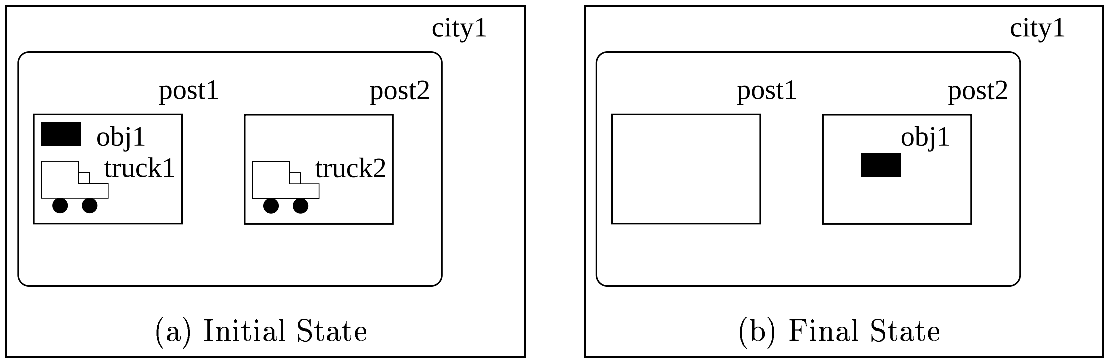

Figure 1: A problem in the logistics transportation domain

the goal. Thus, *truck2* is not relevant to the goal with respect to this solution.

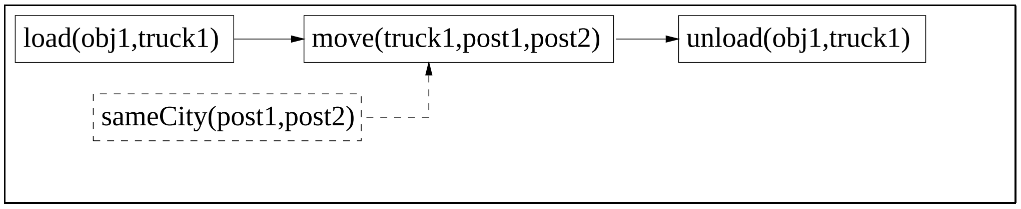

Figure 2: A solution in the logistics transportation domain

2.1 The Foot-printed Similarity Metric

As defined, the foot-printed similarity metric counts the number of relevant features in the case that match features in the new problem. If the percentage of relevant features that are included in the new problem is greater than a certain threshold, the case is retrieved independent of *which features were matched*. However, not all features have the same importance to the solution. To illustrate this affirmation consider the initial state illustrated in figure 3. Suppose that the final state is to place *obj3* in *post4* and that the problem shown in figure 1 and the solution shown in figure 2 conform a case. The case partially match the new problem with the substitution: $\{post1 \rightarrow post3, post2 \rightarrow post4, obj1 \rightarrow obj3, truck1 \rightarrow truck3\}$. However, the solution of the case can not be reused in the new problem because *post3* and *post4* are in different cities, so *truck3* can not be moved from *post3* to *post4*. Thus, the retrieval of the case is considered to have failed in this situation.

2.2 The Weighted Foot-printed Similarity Metric

Incrementing the value of the threshold fixing the percentage of relevant features that must be matched may significantly reduce the number of retrieval failures. However, a high value of the threshold over restricts the situations for which the cases can be retrieved.

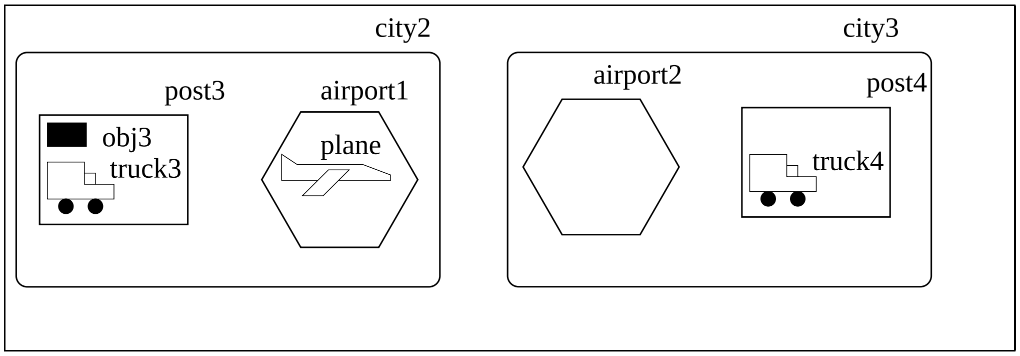

Figure 3: Initial state of a new problem

Ranking the relevant features to establish an order of evaluation improves the adaptability of the cases retrieved (Veloso, 1994a). However, certain features should not only be matched first, but they should be given *more relevance* when deciding if the case is to be retrieved. To illustrate this affirmation, consider the same case as before and a new problem with a similar initial state to the one shown in figure 1. The only difference is that *truck1* is in a third post office, *post3*, placed in the same city as the other two. In this situation 50% of the relevant features in the case match features in the new problem. Thus, unless the threshold has a very low value, the case is not retrieved. However, setting the threshold in a lower value increases the number of failed retrievals. Interestingly, retrieving the case may be adequate depending on the adaptation strategy. For example, the new problem can be solved by adding $move(truck1, post3, post1)$ as first step to the solution shown in figure 2.

The weighted foot-printed similarity metric overcomes these limitations by counting feature weights. Before continuing, some notation must be introduced: a problem P consists of a pair (I^P, G^P), where I^P is the set of features conforming the initial state and G^P is the set of goals conforming the final state of the problem. Cases consist of pairs $((I^C, G^C), Sol)$, where (I^C, G^C) is a problem and *sol* a solution (i.e., a plan). We will suppose for the sake of simplicity that I^C only contains relevant features and that all the goals in G^C interact[1]. We define $sim_{fp}^{wg}(C, P)$, the *weighted foot-printed similarity metric* between C and P, as follows: if G^C matches a subset of G^P with a substitution θ (i.e., $G^C\theta \subset G^P$), then:

$$sim_{fp}^{wg}(C, P) = |G^C| + \sum_{i \in I^C \bigcap_\theta I^P} \omega_{i,C} \qquad (Eq.1)$$

Where $I^C \bigcap_\theta I^P$ denotes the set of all features in I^C matching a feature in I^P with the substitution θ. The factor $\omega_{i,C}$ denotes the weight of the relevant feature i in C. If G^C does not match a subset of G^P, then $sim_{fp}^{wg}(C, P)$ is defined to be zero.

Let P be a new problem, a case, C, is retrieved if: (1) G^C matches a subset of G^P with a substitution θ, and (2) if the weighted proportion of relevant features in I^C matching

[1]The concept of interacting goals (Veloso, 1994a) may be defined as follows: two goals interact if they have at least one relevant feature in common or if there is a third goal they both interact with.

a feature in I^P is greater than a certain threshold, thr. Condition (2) may be written as follows:

$$(\sum_{i \in I^C \bigcap_\theta I^P} \omega_{i,C})/(\sum_{i \in I^C} \omega_{i,C}) \geq thr \qquad (Eq.2)$$

2.3 Weighting Model

For weighting features, a feedback model based on incremental optimizers is used (Salzberg, 1991; Wettschereck and Aha, 1995). Each case, C, contains two counters: k_C and f_C. The first one indicates the number of times a case was adequately retrieved, and the second one the number of times in which not. The weight, $w_{i,C}$, of each relevant feature i is updated as follows:

$$\omega_{i,C} = \omega_{i,C} + \triangle_{k_C,f_C} \quad (with\ 0 \leq \triangle_{k_C,f_C} \leq \beta \times n_C) \qquad (Eq.3)$$

if not matching the feature i caused a failure. On the other hand, if the feature i was not matched, and the retrieval was adequate, then $w_{i,C}$ is updated as follows:

$$\omega_{i,C} = \omega_{i,C} - \triangle_{k_C,f_C} \quad (with\ 0 \leq \triangle_{k_C,f_C} \leq \beta \times n_C) \qquad (Eq.4)$$

If the value of $\omega_{i,C}$ is smaller than 1 then $\omega_{i,C}$ is assigned the value 1 and the weight of the other features in the case is incremented proportionally. The number of features in the initial state of C is denoted by n_C. The change of a feature weight is bounded by a factor, $\beta \times n_C$, directly proportional to the number of features in the case. The incremental factor $\triangle_{k_C,f_C}$ depends on the values of k_C and f_C in the following way: the larger the ratio of k_C to f_C is, the smaller the value of $\triangle_{k_C,f_C}$ is. Thus, as the number of adequate retrieval episodes increases, the effect of a retrieval episode on the feature weights decreases. In contrast, the smaller the ratio of k_C to f_C is, the closer $\triangle_{k_C,f_C}$ to $\beta \times n_C$ is. If the weights of the relevant features were normalized so that $\omega_{i_1,C} + \omega_{i_2,C} + ... + \omega_{i_{n_C},C} = 1$, then the factor $\sum_{j \neq k} \omega_{i_j,C} / \sum_i \omega_{i,C}$ expresses the *reliability* of making an adequate retrieval, when the feature i_k is the only one not to be matched in the new problem. Thus, the feature weights measure the relative relevance of the features in the case.

3 Analysis of Case-Based Problem Solving Episodes in CAPLAN/CBC

The problem solving cycle in our case-based planner, CAPLAN/CBC (Muñoz-Avila et al., 1995; Muñoz-Avila and Hüllen, 1995), consists of four steps: first, cases meeting the retrieval condition stated in the last section are retrieved. Then, the cases are adapted into the new problem. At the third step, an analysis of the adaptation effort is performed and certain features are identified. Finally, the weights of those features are updated.

3.1 Adaptation Strategy in CAPLAN/CBC

CAPLAN/CBC is built on a first-principles planner called CAPLAN (Weberskirch, 1995; Paulokat and Wess, 1994). CAPLAN is a plan-space planner (McAllester and Rosenblitt, 1991). Thus, first-principles planning proceeds by achieving goals and solving conflicts. A goal can be achieved in two ways:

- by *establishing* it with the initial state of the problem, that is, by matching the goal with a feature in the initial state. For example, in figure 2, the goal $sameCity(post1, post2)$ has been established with the initial state shown in figure 1 (a).
- by *establishing* it with a plan step, that is, by matching the goal with the effects of the plan step. For example, in figure 2, the plan step $unload(obj1, truck1)$ is used to achieve the goal: "the object *obj1* must be placed at the post office *post2*" (this goal is not shown in figure 2.).

The adaptation strategy followed in CAPLAN/CBC is known as eager replay (Ihrig and Kambhampati, 1994). Eager replay is done in two phases: in the first phase, each plan step contained in the retrieved cases is replayed in the new situation when replaying the step introduces no inconsistency to the new solution. Once this phase is finished, a partial solution, also known as the "skeletal plan" (Ihrig and Kambhampati, 1994), is obtained. Skeletal plans may contain unsolved goals. A goal remains unsolved because either a corresponding goal does not exist in the cases or a corresponding goal exists, but the way it is achieved in the case can not be replayed in the new problem. At the second phase, the skeletal plan is completed by first-principles planning.

3.2 Evaluating the Adaptation Effort

To evaluate the adaptation effort, three numbers are calculated: n_{case}, the number of plan steps in the case, n_{Sk}, the number of plan steps in the skeletal plan, and, n_{SkFin}, the number of plan steps in the skeletal plan that remain after the completion process is finished. The algorithm evaluating the adaptation effort is shown in figure 4. Its input consists of the three numbers, n_{case}, n_{Sk}, and n_{SkFin}, the case, C, and the skeletal plan, Sk. The outcome of this algorithm is the recomputation of the weights of certain relevant features in the case C.

If the percentage of plan steps replayed is smaller than a certain threshold, $thr1$, or the percentage of plan steps in the skeletal plan that were rejected after completion is smaller than another threshold, $thr2$, the retrieval is said to have failed. Otherwise, the retrieval is said to be adequate. In both situations, the weights of the set of features, $Fail$, returned by the function $FilterFeatures(C, Sk)$ are recomputed (this function will be presented later.).

If the retrieval of the case failed, the weights of features in $Feat$ are increased according to equation 3 (step 2). The argumentation for increasing the weights of those features is that their absence was important in this situation because either only few plan steps

```
evaluateAdaptation($C, Sk, n_{case}, n_{Sk}, n_{SkFin}$)
1. Fail ← FilterFeatures(C,Sk)
2. If $(n_{Sk}/n_{case} \leq thr1)$ or $(n_{SkFin}/n_{Sk} \leq thr2)$
   For-each $i \in Fail$
     $\omega_{i,C} = \omega_{i,C} + \triangle_{k_C,f_C}$
3. Else
   For-each $i \in Fail$
     $\omega_{i,C} = \omega_{i,C} - \triangle_{k_C,f_C}$
```

Figure 4: Algorithm evaluating the adaptation effort

were replayed or too many plan steps in the skeletal plan were rejected during the completion process.

If the retrieval of a case is adequate, the weights of the features in $Fail$ are decreased according to equation 4 (step 3). The reason for decreasing the weights of those features is that their absence was not important in the retrieval episode because enough plan steps were replayed, and the first-principles planner was able to complete the solution without rejecting too many plan steps of the skeletal plan.

Consider the case formed by the problem and solution shown in figures 1 and 2 respectively, and the new problem shown in figure 3. Each of the three plan steps can be replayed in the new situation (i.e., $n_{case} = n_{Sk} = 3$). As a result, the skeletal plan shown in figure 5 is obtained. In this plan, the subgoal $sameCity(post3, post4)$ remains unsolved (represented with a question mark in figure 5). Because $sameCity(post3, post4)$ can not be achieved, the first-principles planner rejects the solution step $move(truck3, post3, post4)$, and pursues other alternatives (i.e., $n_{SkFin} \leq 2$). Thus, a retrieval failure takes place, if, for example, $thr2$ is equal to 2/3.

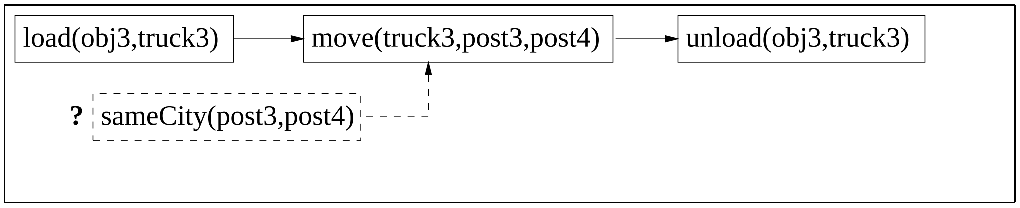

Figure 5: An skeletal plan

3.3 Filtering Non Matched, Relevant Features

To identify the relevant features whose weights must be recomputed, CAPLAN/CBC examines the contribution of the feature to the solution of the case by taking as basis the skeletal plan. For example, the feature $sameCity(post3, post4)$ of the skeletal plan shown in figure 5 was the only one not matched in the new problem, and, as explained before, the retrieval failed. Thus, the weight of this feature must be increased.

In more complex situations non matched, relevant features may not be included in the skeletal plan. Consider, for example, the feature g_3 in the abstract situation described in figure 6. A question arises, namely, whether the weight of g_3 must be recomputed or not. The filtering function, $FilterFeatures(C, Sk)$, returns a set, $Fail$, of non matched, relevant features whose weight must be recomputed. The set $Fail$ meet the following conditions:

1. **Fail explains the failed establishments in the skeletal plan.** In this context, explaining a failed establishment means that if any feature in $Fail$ would have been also present in the new problem, at least one additional goal, g, in the skeletal plan would have been achieved. This condition is trivial if g is established with the initial state in the case. Examples of such goals are $sameCity(post3, post4)$ that occurs in the skeletal plan shown in figure 5, and g_1 that occurs in the skeletal plan shown in figure 6. However, if g is established with a plan step in the case, a careful analysis must be done to meet this condition. For example, in figure 6, it is assumed that g_2 remains unsolved because a failed establishment with the plan step s_2 occurs.

2. **Fail constitutes the minimal set explaining the failed establishments.** That is, if any feature in $Fail$ is removed, at least one failure of a goal in the skeletal plan can not be explained in the sense described at point 1.

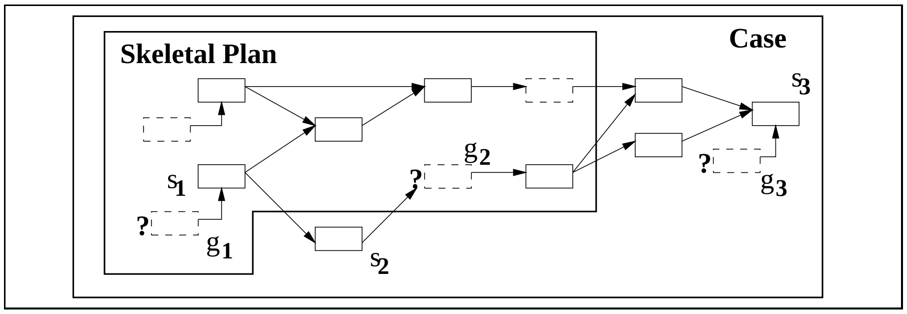

Figure 6: Abstract configurations of an skeletal plan and a case

The function $FilterFeatures(C, Sk)$ is shown in figure 7. Its input are the retrieved case, C, and the skeletal plan, Sk, and returns the set $Feat$. This function uses two global variables: Arg and G. The set, Arg, contains all arguments of features in $Feat$ and is computed dynamically. Initially, Arg is empty because $Feat$ is empty (steps 1 and 2). The idea of the function $FilterFeatures(C, Sk)$ is to examine each unsolved goal in the skeletal plan. To accomplish this, G contains the unsolved goals in Sk that have not been examined. Initially, G is assigned all unsolved goals in Sk that have a corresponding goal in C (step 3). Thus, the goals in the skeletal plan that remain unsolved because they have no corresponding goal in the case are not examined.

Recall that a goal can be achieved by establishing it either with the initial state or with a plan step. Thus, there are only two possible reasons why the way a goal was achieved in the case can not be replayed in the new problem:

- The establishment with the initial state can not be performed because there is no feature in the initial state of the new problem that matches the goal. Examples of failed establishment with the initial state are $sameCity(post3, post4)$ in figure 5 and g_1 in figure 6. This failure is considered in the first control block of the algorithm (step 4): each goal, g, whose plan step in C is established with the initial state (i.e., $isEstI(g, C)$ is true), is stored in $Feat$ (step e.1) and its arguments are collected in Arg (step e.2).

- The establishment with a plan step can not be performed because an inconsistency in the variable bindings will be introduced. An inconsistency can be expressed as follows (Kambhampati et al., 1995): there are two variables x_1 and x_2 bound with the constants a and b respectively, and the constraint $x_1 = x_2$ is stated in the plan step. For example, in figure 6, it is assumed that s_2 can not be replayed because a failed establishment with a plan step occured. Thus, the goal g_2 remains unsolved in the skeletal plan. This failure is considered in the second control block of the algorithm (step 5). If the inconsistent arguments of the plan step are included already in Arg, nothing is done (step p2). Otherwise, two steps are performed: first, each argument, c_i, that was not included in Arg already is added to Arg (steps p3.1, e2). Second, for each argument c_i, the feature found by calling the function $searchFeat(g, c_i, C)$ is add to $Feat$ (steps p3.2, e1). This function returns a feature in the case that has c_i as argument and whose distance[2] to g is minimal.

```
FilterFeatures($C, Sk$)
1. $Feat \leftarrow \{\}$
2. $Arg \leftarrow \{\}$
3. $G \leftarrow unsolvedGoalsMatched(Sk, C)$.
4. while (exists $g \in G$ with $isEstI(g, C)$)
   4.1 processEst(g)
   4.2 G ← G − $\{g\}$
5. while ($G \neq \{\}$)
   5.1 processInconsistency(g,C,Sk)
   5.2 G ← G − $\{g\}$
6. return Feat
```

```
processEst(g)
e1. Feat ← Feat $\bigcup \{g\}$
e2. $Arg \leftarrow Arg \bigcup argIn(g)$
```

```
processInconsistency(g,C,Sk)
p1. $I \leftarrow argInconsistentStep(g, C)$
p2. If $I \subset Arg$ then skip
p3. while ($I - Arg$) isNotEmpty
   p3.1 Let $c_i \in I - Arg$
   p3.2 g' ← $searchFeat(g, c_i, C)$
   p3.3 processEst(g')
```

Figure 7: Algorithm for filtering features

4 Empirical Results

The experiments were made in two different domains: the logistics transportation domain and the domain of process planning (Muñoz-Avila and Weberskirch, 1996). During problem solving, the former is characterized by the high number of occurrences

[2]The number of arcs of the shortest path connecting two goals in the plan.

of goal blocks caused by goal loops[3] whereas the latter by the high number of interactions between goals. Thus, both domains provide a wide spectrum of problems for our experiments in which we validate the following hypothesis:

Over a period of multiple problem solving episodes, weighting relevant features in a case increases the reliability that the retrieval of the cases is adequate by converging to best weights.

4.1 The experiments

This hypothesis was validated with an experiment performed with two collections of problems (one for each domain). The collections were formed in the following way: a problem, that we called the pivot problem, was manually stated. Then, some features in the initial state were randomly fixed. A new goal and new features that do not occur in the pivot problem were also given. Taking as basis the pivot problem, new problems were formed by varying the fixed features, or/and by adding the new goal and the new features. The problem collection met also the following conditions: (1) every problem in the collection can be solved within 300 seconds in the logistics transportation domain and 350 seconds in the domain of process planning. (2) The pivot case meets the foot-printed retrieval condition for every problem with $thr = 75\%$ for the transportation domain and $thr = 80\%$ for the domain of process planning (thr as in equation 2.). The reason for the parameter thr to be higher the domain of process planning is the high level of interactions taking place in this domain, which obligates a more accurate retrieval. (3) The number of times that fixed features where changed in the collection is the same. For example, in the transportation domain, if a truck and a post-office are fixed, then, in the initial state, the number of problems in which the truck is changed of location is the same as the number of times the post office is changed of city. (4) If n denotes the number of fixed features, then problems were ordered in a way that within a sequence of problems, $Problem_{mn+1}, ..., Problem_{mn+m}$, the number of changes of every fixed feature is the same. For this reason, the number of problems in the collection is a multiple of the number of selected features.

The ideal experiment to validate our hypothesis is to form all possible combinations of every collection of problems and show that, in average, our hypothesis holds. Because this implies a combinatorial explosion, we stated conditions (3) and (4), in pursue of equally distributing the effect of every change in the fixed features and of capturing the average situation. Condition (3) ensures that no feature takes advantage of the others by changing them the same number of times. As explained before, the ratio of k_C to f_C determines the incremental rate of the weigths. Thus, if not matching a fixed feature causes an inadequate retrieval, then the change of weights will be greater when problems changing that feature are allocated at the beginning of the collection. In contrast, if those problems are allocated at the end of the collection, the changes in weights are reduced. For this reason, condition (4) ensures that the final weight of each fixed feature is closer to the average by distributing the problems changing the feature equally throughout the collection.

[3]Informally, a goal loop occurs when to achieve a goal, the planner pursues to achieve another goal that unifies the first one.

In each domain three collections of problems were formed. In the transportation domain 6 features were fixed in every collection, and 8 in the domain of process planning. The size of each collection was 5 times the number of fixed features. Before each run, a solution to the pivot problem was found. The pivot problem and its found solution are taken as a pivot case. At each run, each problem in the collection was given and a complete problem-solving episode was performed (see section 3).

4.2 The results

Tables 1 and 2 summarize the results of this experiment for the domain of process planning and for the logistics transportation domain respectively. Tables 1 (a) and 2 (a) show the results when weights were recomputed. The first row of the tables 1 (a) and 2 (a) present the percentage of times the pivot case was retrieved. The second row of the same tables present the percentage of times that retrieving the pivot case caused a retrieval failure. The third row presents the percentage of times in which retrieving the pivot case would have been adequate, but it was not retrieved. Each column, m, presents the results for the sequence of problems, $Problem_{mn+1}, ..., Problem_{mn+n}$.

Items	**1**	**2**	**3**	**4**	**5**
% Retr.	80	70	60	50	50
% Failed	37	29	17	0	0
% Nret.	0	0	0	0	0

Items	**80%**	**85%**	**90%**	**95%**
% Retr.	100	80	30	30
% Failed	50	25	25	0
% Nret.	0	0	29	29

Table 1: Measures of effectiveness of retrieval in the domain of process planning (a) by weighting features, and (b) by varying the fixed threshold.

Tables 1 (b) and 2 (b) summarize the results of changing the fixed percentage, *thr*, without considering the feature weights. Each column summarizes the results when the corresponding percentage was fixed, and all the problems in the collections were given.

Items	**1**	**2**	**3**	**4**	**5**
% Retr.	100	86	71	71	71
% Failed	43	33	17	17	17
% Nret.	0	0	0	0	0

Items	**75%**	**80%**	**85%**	**90%**
% Retr.	100	86	86	0
% Nadeq.	43	33	33	0
% Nret.	0	0	0	100

Table 2: Measures of effectiveness of retrieval in the logistics transportation domain (a) by weighting features, and (b) by varying the fixed threshold.

These experiments show that feature weighting increases the probability that the retrieval is adequate as the number of problem solving episodes increases. This can be seen by comparing the fifth column of table 1 (a) (resp. 2 (a)) with the first column of table 1 (b) (resp. 2 (b)). That is, comparing the effectiveness of retrieval by recomputing weights as several problem solving episodes took place and the effectiveness of retrieval without recomputing weights respectively. Further, these experiments show that an increase in the effectiveness of the retrieval can not be gained by changing the fixed percentage *thr* of initial features that must be matched (see tables 1 (b) and 2 (b)). Finally, by comparing the fourth and fifth columns of table 1 (a) (resp. 2 (a)), we conclude that the feature weights converge.

5 Discussion

The feature weights may be interpreted as a hypothesis about the relative importance of the features in a case. This hypothesis is validated or rejected in problem solving episodes where one or more relevant features are absent. The effect on the adaptation effort caused by the absence of a relevant feature in a new problem depends on (1) the contribution of the feature to the solution of the case, (2) the characteristics of the domain, (3) the adaptation strategy, and (4) the features that are present in the new problem but not in the case. The first three factors are considered implicitly by the algorithm *evaluateAdaptation*. The last factor could be important in situations in which the retrieval fails, or the retrieval is adequate *exclusively* because of the additional features in the new problem. Thus, in these situations, the feature weights should not be recomputed. Goal regression (Kambhampati et al., 1995) may be used to detect those situations and may speed-up the convergence of the feature weights. However, in the experiments discussed before, goal regression was not necessary because of the uniform distribution of the problems, the domain characteristics, and the adaptation strategy being used.

6 Conclusion

In this paper, we presented the weighting foot-printed similarity metric and an algorithm to analyze and explain the performance of the cases during case-based problem solving episodes. We integrated them in the feature weighting model and performed experiments in two representative domains. Based on these experiments, we conclude that this integration improves the adaptability of the retrieved cases by converging to best weights.

References

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundation issues, methodological variations and system approaches. *AI-Communications*, 7(1):pp 39–59.

Fox, S. and Leake, D. (1995). Using introspective reasoning to refine indexing. In *(Veloso and Aamodt, 1995)*.

Ihrig, L. and Kambhampati, S. (1994). Derivational replay for partial-order planning. In *Proceedings of AAAI-94*, pages 116–125.

Ihrig, L. and Kambhampati, S. (1995). Automatic storage and indexing of plan derivations based on replay failures. In *Proceedings of IJCAI-95*.

Kambhampati, S., Katukam, S., and Qu, Y. (1995). Failure driven dynamic search control for partial order planners: An explanation-based approach. *Artificial Intelligence.* (submitted, ASU-CSE-TR-95-010).

Leake, D. B., Kinley, A., and Wilson, D. (1995). Learning to improve case adaptation by introspective reasoning and cbr. In *(Veloso and Aamodt, 1995).*

Marir, F. and Watson, I. (1995). Representing and indexing building refurbishment cases for multiple retrieval of adaptable pieces of cases. In *(Veloso and Aamodt, 1995).*

McAllester, D. and Rosenblitt, D. (1991). Systematic nonlinear planning. In *Proceedings of AAAI-91*, pages 634–639.

Minton, S. (1988). *Learning Search Control Knowledge: An Explanation-Based Approach.* Kluwer Academic Publishers, Boston.

Muñoz-Avila, H. and Hüllen, J. (1995). Retrieving relevant cases by using goal dependencies. In *(Veloso and Aamodt, 1995).*

Muñoz-Avila, H., Paulokat, J., and Wess, S. (1995). Controlling non-linear hierarchical planning by case replay. In Keane, M., Halton, J., and Manago, M., editors, *Advances in Case-Based Reasoning. Selected Papers of the 2nd European Workshop (EWCBR-94)*, number 984 in Lecture Notes in Artificial Intelligence. Springer.

Muñoz-Avila, H. and Weberskirch, F. (1996). Planning for manufacturing workpieces by storing, indexing and replaying planning decisions. In *Third International Conference on AI Planning Systems (AIPS-96).* AAAI-Press.

Paulokat, J. and Wess, S. (1994). Planning for machining workpieces with a partial-order nonlinear planner. In Gil, Y. and Veloso, M., editors, *AAAI-Working Notes 'Planning and Learning: On To Real Applications'*, New Orleans.

Richter, M., Wess, S., Althoff, K., and Maurer, F., editors (1994). *First European Workshop on Case-base Reasoning (EWCBR-93).* Number 837 in Lecture Notes in Artificial Intelligence. Springer Verlag.

Salzberg, S. L. (1991). A nearest hyperrectangle learning method. *Machine Learning*, 1.

Smyth, B. and Keane, M. (1994). Retrieving adaptable cases. In *(Richter et al., 1994).*

Smyth, B. and Keane, M. (1995). Experiments on adaptation-guided retrieval in case-based design. In *(Veloso and Aamodt, 1995).*

Veloso, M. (1992). *Learning by Analogical Reasoning in General Problem Solving.* PhD thesis, Carnegie Mellon University.

Veloso, M. (1994a). *Planning and learning by analogical reasoning.* Number 886 in Lecture Notes in Artificial Intelligence. Springer Verlag.

Veloso, M. (1994b). Prodigy/analogy: Analogical reasoning in general problem solving. In *(Richter et al., 1994).*

Veloso, M. and Aamodt, A., editors (1995). *Case-Based Reasoning Research and Development, Proceedings of the 1st International Conference (ICCBR-95).* Number 1010 in Lecture Notes in Artificial Intelligence. Springer Verlag.

Weberskirch, F. (1995). Combining SNLP-like planning and dependency-maintenance. Technical Report LSA-95-10E, Centre for Learning Systems and Applications, University of Kaiserslautern, Germany.

Wettschereck, D. and Aha, D. W. (1995). Weighting features. In *(Veloso and Aamodt, 1995)*.