# AN EFFICIENT CASE-BASED PLANNING SYSTEM: PRELIMINARY RESULTS

FLAVIO TONIDANDEL[1], MÁRCIO RILLO[1,2]

[1] *Universidade de São Paulo - Escola Politécnica*
*Av. Luciano Gualberto 158, trav3, São Paulo, SP, Brasil, 05508-900*
[2] *Faculdade de Engenharia Industrial*
*Av. Humberto de A. Castelo Branco, 3972, São Bernardo do Campo, SP, Brasil, 09850-901*
*E-mails:* flavio@lac.usp.br, rillo@lsi.usp.br

**Abstract** --Heuristic search planners, as FF and HSP systems, are new planning approaches which performance has been increasing over the last years. One way to outperform the heuristic search planners is using the case-based planning approach, but new similarity metrics must be investigated. This is because the old similarity rules used in previous case-based planning systems are not accurate enough and, consequently, the systems will require more time to adapt a retrieved case than the heuristic search planners require to find a solution. This paper examines the use of a new and more accurate similarity rule, called ADG, in a case-based planning system, and confronts its results with the results of the FF heuristic search planner.

**Key words** – Artificial Intelligence Planning ; Case-Based Reasoning ; Similarity Metric .

## 1 Introduction

In Artificial Intelligence, the planning area is a research field that is more related to intelligent automation field than others. Consequently, the interest of industries in the planning area has been increasing over the last years,. One factor that collaborates for this increasing has been the results of the performance of the planning systems based on heuristic search that have been aplying with some degree of success to problems in automation and factory production (Hoffman; Nebel, 2001)(Bonet; Geffner, 1999).

The researches in planning area have been focusing on how to improve the efficiency of the heuristic search planners. One way is by using previous executed plans to find a solution for a new problem. This approach is named Case-based Planning (CBP), which stores previous plan in a memory as cases.

Although the theoretical complexity of CBP systems and planning systems are *NP-Hard* (Nebel; Koehler, 1995), some previous researches have shown that the CBP systems outperform the results of generative planning systems (Koehler, 1996) (Muñhoz-Avila, Hüllen, 1996)(Veloso, 1994). However, the performance of a CBP system depends on the accuracy of the similarity metric, which selects the most appropriate stored case to help in the new problem solution. The most used similarity metrics is not designed to take into account the effort of the case adaptation. Consequently, CBP systems with one of these similarity metrics will rarely outperform the new planning approach based on heuristic search.

The purpose of this paper is to test a new similarity rule, called ADG, in a CBP system in order to surpass in performance the fastest planning system in the AIPS'2000: the FF system (Hoffman;Nebel, 2001).

This paper is a further step of (Tonidandel, Rillo, 2001) towards a new CBP system, which uses the ADG similarity rule (Tonidandel, Rillo, 2001) to outperform the heuristic search planners. The following section presents an overview of planning and CBP systems. Section 3 presents the Transaction Logic used to formalize the planning components. Section 4 shows the ADG similarity rule. The first results of the CBP system with the use of ADG is analyzed and compared with generative planning system in section 5. Finally, the paper is concluded in section 6.

## 2 Case-Based Planning Systems

Most planning systems are generative planners. They find a solution 'from scratch', i.e., they find a plan that transforms the initial state into a desirable final state. This kind of planner always starts the planning process from the beginning, even when the same problem was solved previously.

One way to avoid that the planner repeats the planning process that was executed in some previous planning is using case-based reasoning techniques (Kolodner, 1993). The case-based reasoning (CBR) was first used in planning by the system CHEF (Hammond, 1989), which received the denomination of a case-based planning (CBP) system.

A CBP system is distinguished by using a memory of previous executed plans. Each previous plan is usually stored with specific features of the situation that the plan was applied. These features are used to indicate how much similar is the 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 less similar one.

After the choice of the most similar case, a CBP system starts the adaptation process. In most of CBP systems, this process is composed by 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.

In this cycle, the retrieval phase is mainly responsible for the efficiency of the system. It must search in a space of cases in order to choose a one case that allows the system to solve a problem easily.

Designing 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.

Some adaptation-based similarity rules and techniques were proposed in the past, such as RCR in DIAL system (Leake;Kinley, 1997) and AGR (*Adaptation-Guided retrieval*) in DéjàVu system (Smyth;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 can not be directed applied in domain independent case-based planning.

On the other hand, domain independent rules of similarity are usually designed as a difference between states. This difference is a confrontation between *(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 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. A state-space CBP system is designed to search in a space of states. An alternative approach to planning with states is the plan-space planning systems (Muñhoz-Avila;Hüllen, 1996). They search in a space of plans and they have no goals, but only tasks to be achieved. Since tasks are semantically different of goals, the similarity metric designed for plan-space CBP systems is also different from the similarity rules designed for state-space CBP systems.

An interesting similarity rule in plan-space approach is presented in the CAPLAN/CBC system (Muñhoz-Avila;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.

The ADG (*Action-Distance Guided*) similarity (Tonidandel; Rillo, 2001) differs from previous approaches. First, because it is designed for general state-space CBP systems that are independent of the domain. Second, it takes into account the adaptation effort based on the distance between states. It does not use any additional domain knowledge neither need to learn any knowledge to perform an accurate estimate. None of previous approach has all these features.

The purpose of this paper is, therefore, to perform comparative tests of the performance of a CBP system using the ADG similarity and the results of the FF planner, which was the fastest planner in the AIPS'00 competition (Bacchus, 2000).

## 3 The Transaction Logic in Planning

The Transaction Logic (TR) (Boner; 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., $\alpha \otimes \beta$, which means "first execute $\alpha$, and then execute $\beta$".

It uses the following notation to describe a transaction: $P, D_0,...,D_n \models \phi$, 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; $D_0,...,D_n$ is a sequence of databases that represents an updating made by $\phi$. This updating is performing by *ins(_)* an *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 $D_k$.

With all these formal features, TR is a suitable logic to describe actions and plans for planning systems (Tonidandel, Rillo, 1998)(Tonidandel, Rillo, 2000)(Santos; 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.

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 $\boldsymbol{L}$ 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.*

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

**Definition 2** (Strips-like Actions): *Considering $\boldsymbol{A} \subseteq \boldsymbol{L}$ as a set of action definitions, each $\alpha$, $\alpha \in \boldsymbol{A}$, 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 all facts that will not be true after the action execution.*
- *add($\alpha$) is the add list of the action $\alpha$. It represents all facts that will be true after the action execution.*

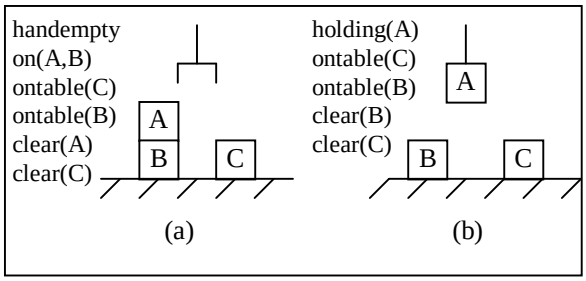



Figure 1. Two examples of state definition in block-world domain.

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 α just can be executed from a state *D* if pre(α) ⊆ *D*. For example, consider the following action in the well-known blocks-world domain:

*unstack (x,y)* ← *qry(handempty(x))* ⊗ *qry(clear(x))* ⊗ *qry(on(x,y))* ⊗ *del(clear(x))* ⊗ *del(handempty)* ⊗ *del(on(x,y))* ⊗ *ins(holding(x))* ⊗ *ins(clear(y)).*

which means to get the block *x* from the top of block *y*. Consider now a state $D_0$ as the state described in Fig. 1a and the state *Df* as the state described in Fig. 1b. Thus, *unstack(A,B)* can be satisfied in TR as: *P*, $D_0$...*Df* |= *unstack(A,B).* Where *P* is the set of actions *A* and <$D_0$...*Df* > is the state path from $D_0$ to *Df* by the application of *del* and *ins* predicates contained in formula *unstack(A,B).*

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

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

**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* η *is a TR rule:*

$$\eta \leftarrow \mathbf{Wi} \otimes \alpha_1 \otimes ... \otimes \alpha_n \otimes \mathbf{Wf},$$

*where:*

- η *is a TR rule that represents a stored case.*
- $\alpha_i \in \mathbf{A}$; $1 \leq i \leq n$, *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*

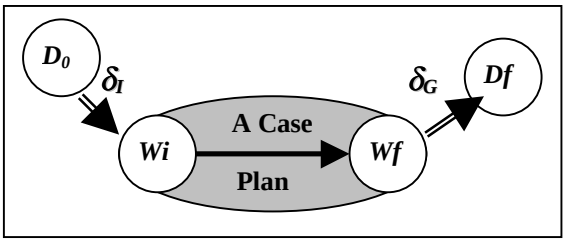



Figure 2. The values involved in ADG similarity metric. The *Initial Similarity Value* ($\delta_I$) is the distance between the Initial State $D_0$ and the *Wi* of a case. The Goal Similarity Value ($\delta_G$) is the distance between the *Wf* of a case and the desirable final state *Df*.

Intuitively, *Wi* is a set of those facts that are deleted by the plan. It is equal to the result of the foot-printing method used by PRODIGY/ANALOGY system (Veloso, 1994) and it represents the pre-condition of the plan. The process of foot-printing the initial state is described in (Veloso, 1994).

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. There is no correspondence to other cases features in any case-based planning system.

## 4 The ADG similarity

The ADG similarity was purposed by Tonidandel and Rillo (2001) to supply the lack of researches in accurate similarity rule for case-based planning systems. The ADG similarity is more accurate than the similarity rules usually applied in CBP systems.

It is obtained by calculating two estimates of the distance between states. Both estimates are based in 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 *Wi*. The second value is called *goal similarity value* ($\delta_G$) and it is a distance estimate between the desirable final state (*Df*) of the goal (def. 4) and the final state features (*Wf*) of the case (Fig. 2).

The process of estimating the distance between two states is obtained by the calculation of the heuristic used by the FF planner. In fact, the ADG similarity uses the same heuristic, which is used by FF planner to guide the search, to determine the initial and final similarity values.

The first step for determining the heuristic value is to create a graph for a relaxed planning problem. Hoffman (2001) explains 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 Hoffman (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

```
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 α level=i-1; g ∈ add(α);
    h:=h+1;
    for all d level ≠ 0 ∈ pre(α), d not True
       at time i-1 do
      G level(d) := G level(d) ∪ {d};
    endfor
    for all d ∈ add(α) do
      mark d as True at time i-1 and i;
    endfor
  endfor
endfor
return h;
```

(a)

```
relaxed_final_length(Wi,Wf,Df)

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

(b)

Figure 3. The ADG similarity algorithms: (a) The algorithm that computes the relaxed solution from a relaxed fixpoint, where G is the target-state. It is extracted from (Hoffman, 2001); (b) The algorithm that extracts the distance between *Wf* and *Df* by considering the marks from relaxed_initial_length(*Wi*).

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 (Hoffman; Nebel, 2001), they are:

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

**Definition 7:** *level(α) := min {i | α ∈ $O_i$, where $O_i$ is the $i^{th}$ 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 (Hoffman, 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 *Wi*. First, each fact in *Wi* 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 *Wi* is the number of action selected to satisfy the goals in each layer. The algorithm to compute the relaxed solution is shown in Fig. 3a, 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*: $\delta_I = h$.

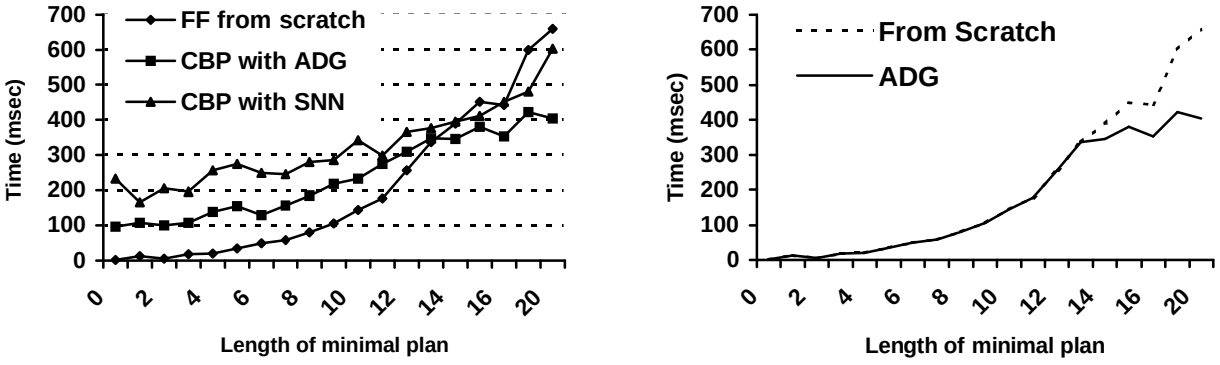
In order to calculate the second value $\delta_G$, it is necessary to force the solution trace from $D_0$ to consider the actions in the case. To do this suitably, it is necessary to maintain the values of each mark of each fact after the performing of the *Initial Similarity Value* calculation. It means that the function ***relaxed_final_length(Wi,Wf,Df)*** (Fig. 3b) will be called using the marks changed by the ***relaxed_initial_length(Wi).***

In addition, all marks of all facts in *Wi* must be set as false and the marks of each fact in *Wf* must be set as true in their correspondent layer. It is also necessary to initialize *Wi* as a goal, because the trace must be calculated through the actions of the case (Tonidandel, Rillo, 2001).

As highlighted by Tonidandel and Rillo (2001), 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 actions between $D_0$ and *Wi*.

The result of the function ***relaxed_final_length (Wi,Wf,Df)*** of the algorithm presented in Fig. 3b 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: $\boldsymbol{ADG = \delta_I + \delta_G}$.




(a) (b)

Figure. 4. It shows the results of the experiment. (a) The processing time of the FF planner when plans from scratch, the CBP planner with ADG and the CBP with SNN (b) The results of a CBP planner that mixes the use of previous cases with ADG similarity and the use of FF planner to plan from scratch.

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 *Df*, that can be calculated with ***relaxed_initial_length(Df).*** If this distance is less than the ADG value of any stored case, a generative planner 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 *Df*, the use of a retrieved case is preferable.

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 modifications 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 *Df*.

To complete a case, first the generative planner finds, from scratch, a plan *p1* that goes from the initial state to a new state where the *Wi* of the case is satisfied. Then, all facts of the *Wi* are deleted from this new state, and all facts of the *Wf* are inserted. The resulted state is a new initial state for the generative planner that, finally, finds a plan *p2* to satisfy the desirable final state *Df*. The solution resulted is a concatenation of *p1*, case actions and *p2*. This process of completing is considered in the experiment below.

## 5 Experiments

Some tests are performed to show the improvement provided by the use of the ADG similarity. In the tests, the Blocks World domain with 5 blocks is used. It is a simple domain, but it is one of the most difficult domains for planning systems. Blocks World definition and problems can be found in (Bacchus, 2000).

In order to perform tests, a case-base with 100 cases in blocks world domain is generated by a case-base seeding program. The tests also use the FF system as the generative planner that will find a solution from scratch.

The FF uses a heuristic to guide a hill-climbing search engine. The heuristic is the estimate of the distance between the current state and the goal state (*Df*). It is calculated at each state in the searching.

Since the heuristic estimation is accurate, the search engine is correctly directed by it and the solution plan can be easily found. Some other heuristics as goal ordering, states pruning techniques, and others are also used in FF (Hoffman; Nebel, 2001).

In the experiment, the FF planner is used in the adaptation phase of the CBP system to complete a case and to find a solution. For the CBP system, two tests are performed. One uses the ADG similarity, and another uses the SNN (*Standard Nearest Neighborhood)* (Kolodner, 1993) representing a general similarity rule used in most CBP systems.

The SNN calculates first the normalized difference between the initial state and the *Wi* of a case. Then, it calculates the normalized difference between the desirable final state (*Df*) and *Wf*.

The similarity determination process takes less than 0.01 seconds to search and find a similar case in a memory with 100 cases for both calculations: ADG and SNN. However, most of cases chosen by the ADG are more accurate than the cases chosen by the SNN. It permits that the CBP with the ADG similarity takes less time to adapt the chosen cases.

All tests are performed in the same computer and operational system. For each test, an initial state ($D_0$) and a final state (*Df*) are randomly generated. The FF planner is applied to find, from scratch, a plan that goes from $D_0$ to *Df*. Its processing time is then annotated. After, the CBP system is applied to search a case that can find a solution more easily in the case-base, following both ADG and SNN values. The retrieved case is then completed using the FF planner. The time is annotated for both.

The Fig. 4a presents the results of the FF planner, the CBP with ADG, and the CBP with SNN, after performing 1000 tests. The curve of each method is the medium value obtained for a solution of problems with different length. The axis *x* represents the minimal number of actions necessary to solve a problem.

When the solution of a problem requires only few actions, the FF planner performs better. However, when the solution becomes more complex and more actions are necessary, the CBP system with ADG is better. It is possible to note that the use of the CBP with SNN does not perform as good as the CBP with ADG. It confirms that old similarity rules that are usually applied in CBP systems are not appropriate to improve the performance of a heuristic search planner.

On the other hand, the use of the ADG similarity permits that the CBP system performs better in difficult situations. Fig. 4b shows that if the CBP system with ADG works together with FF planner, it is possible to create a system that have, in the worst case, the same performance as FF planner.

The results show, therefore, that a CBP system can improve the performance of the heuristic search planners with the use of the ADG similarity metric.

## 6 Conclusion

This paper is the first result toward a complete and efficient case-based planning system that can also outperform the fastest planning system of the AIPS'00 competition (Bacchus, 2000). The main part developed and presented in this paper is the use of the ADG similarity rule (Tonidandel; Rillo, 2001) which is more accurate than the similarity metric used by some case-based planning systems.

The results presented in section 5 also show that a CBP system that uses the ADG similarity a CBP system can guarantee better performance than the FF planner in hardest situations.

However, other tests with other planning domains must be analyzed, and other methods must be incorporated in the CBP system in order to guarantee an improvement of the performance and the results.

## Acknowledgments

This work is supported by FAPESP under contract number 98/15835-9.

## References

Bacchus, F. (2000). AIPS-2000 Planning Competition Results. Available in: http://www.cs.toronto.edu/aips2000

Bonet, B; Geffner, H. (1999). Planning as Heuristic Search: New Results. *Proceedings of European Conference on Planning – ECP'99*. Durham, UK.

Bonner, A.J., Kifer, M.: (1995) Transaction logic programming. *Technical Report, CSRI-323*, Department of Computer Science, University of Toronto.

Hammond, K. (1989). *Case-Based Planning:Viewing Planning as a Memory Task*. Academic Press.

Hoffman, J.; Nebel, B. (2001). The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research – JAIR*, vol. 14, pp 253-302.

Kolodner, J. (1993). *Case-Based Reasoning* . Morgan Kauffmann.

Koehler, J. (1996). Planning from Second Principles. *Artificial Intelligence*, Elsevier Science, no. 87, pp. 148-187.

Leake, D., Kinley, A., Wilson, D. (1997). Case-Based Similarity Assessment: Estimating Adaptability from Experience. *Proceedings of 14th National Conference on Artificial Intelligence – AAAI'97*. AAAI Press.

Muñoz-Avila, H.,Hüllen, J. (1996). Feature Weighting by Explaining Case-Based Planning Episodes. In: Smith, I., Faltings, B. (Eds) *Proceedings of 3rd European Workshop on Case-Based Reasoning (EWCBR'96)*. Lecture Notes in Artificial Intelligence, Springer-Verlag, vol 1168. pp. 280-294.

Nebel,B. ; Koehler, J. (1995). Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence Special Issue on Planning and Scheduling*, no. 76, pp.427-454.

Santos, M., Rillo, M. (1997). Approaching the *Plans are Programs* Paradigm using Transaction Logic. In: Steel, S., Alami,R (Eds) *Proceedings of 4th European Conference on Planning – ECP'97*. Lecture Notes in Artificial Intelligence, Springer-Verlag, vol. 1348, pp. 377-389.

Smyth, B., Keane, M. (1998). Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning. *Journal of Artificial Intelligence*, vol. 102(2), pp. 249-293.

Tonidandel, F, Rillo, M. (1998). Case-Based Planning in Transaction Logic Framework. *Proceedings of Workshop on Intelligent Manufacturing Systems (IMS'98)*. Gramado, Brasil, Elsevier Science. pp. 281-286.

Tonidandel, F., Rillo, M. (2000). 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, Springer-Verlag, vol 1952, pp. 23-32.

Tonidandel, F.; Rillo, M. (2001). An Accurate Adaptation-Guided Similarity Metric for Case-Based Planning In: Aha, D., Watson, I. (Eds.) *Proceedings of 4th International Conference on Case-Based Reasoning (ICCBR-2001)*. Lecture Notes in Artificial Intelligence, Springer-Verlag, vol 2080, pp. 531-545.

Veloso, M. (1994). *Planning and Learning by Analogical Reasoning*. Lecture Notes in Artificial Intelligence, vol 886. Springer-Verlag.