Appears in: Proceedings of the first International Conference on Case-Based Reasoning, Case-Based Reasoning Reseach and Development, LNAI 1010, Springer Verlag, 1995

Retrieving Cases in Structured Domains by Using Goal Dependencies*

Héctor Muñoz-Avila and Jochem Huellen

Centre for Learning Systems and Applications (LSA)
University of Kaiserslautern, Dept. of Computer Science
P.O. Box 3049, D-67653 Kaiserslautern, Germany
E-mail: {munioz|huellen}@informatik.uni-kl.de

Abstract. Structured domains are characterized by the fact that there is an intrinsic dependency between certain key elements in the domain. Considering these dependencies leads to better performance of the planning systems, and it is an important factor for determining the relevance of the cases stored in a case-base. However, testing for cases that meet these dependencies, decreases the performance of case-based planning, as other criterions need also to be consider for determining this relevance. We present a domain-independent architecture that explicitly represents these dependencies so that retrieving relevant cases is ensured without negatively affecting the performance of the case-based planning process.

1 Introduction

Reusing previous problem solving experience has proven to speed-up planning systems. Problem solving experience can be stored in generalized (Minton, 1988) or abstracted (Bergmann & Wilke, 1995) form, or it can be stored as cases (Veloso, 1994; Ihrig & Kambhampati, 1994; Yang & Lu, 1994). One of the methods better studied and used for adapting cases is derivational analogy (Carbonell, 1983; Veloso & Carbonell, 1993; Cunningham & Slattery, 1994) that basically consists of replaying the planning-decisions taken in selected cases, while solving a new problem. As other adaptation approaches (Smyth & Keane, 1994), the effectivity of derivational analogy depends on the adaptability of the cases selected. As a result, the derivational analogy replay method has been integrated with retrieval procedures in general problem solving systems (Veloso, 1994).

CAPLAN/CBC (Muñoz-Avila, Paulokat, & Wess, 1994) is a case-based planning system that is build on top of a partial-order nonlinear planner (Paulokat & Wess, 1994) and uses derivational analogy for replaying past solutions. To overcome the problem of retrieving adaptable cases, the first version of CAPLAN/CBC used classifications manually defined by domain experts as an index for the case-library. However, such classifications are not avaliable in every

* This research was partially sponsored by the Deutsche Forschungsgemeinschaft (DFG), Sonderforschungsbereich (SFB) 314: "Künstliche Intelligenz - Wissensbasierte Systeme", Project X9 (1991 - 1995).

domain. Further, domain-dependent classifications may be made, based on technological issues different than adaptability of the solutions. As a result, it was necessary to perform additional tests during the retrieval phase to ensure the relevancy of the cases retrieved, increasing thereby the average execution time.

There are domain where structural dependencies between certain key elements in the domain are defined. These dependencies are based on intrinsic processing restrictions and they form part of the problem descriptions. They also reduce the range of possible solutions for a problem as they establish relations between elements that must be meet by any solution. Thus, they must be taken into account when retrieving cases during a case-based solving process.

In this paper we define the notion of dependencies between elements of structured domains. Based on this notion a domain-independent architecture for the case-base is presented that extends the one developed in PRODIGY/ANALOGY (Veloso, 1994). By comparing it against the one in PRODIGY/ANALOGY, we show that this architecture ensures the retrieval of relevant cases for structured domains in better time ranges.

Subsequent sections present an example of a structured domain. Then, we state the requirements for a retrieval procedure and give a survey the architecture of the case-base in PRODIGY/ANALOGY. After that, section 4 defines the concept of dependencies. Then, we present the architecture of the case-base in CAPLAN/CBC (section 5) and introduce the retrieval procedure based on this architecture (section 6). In section 7, the results of an experiment comparing the retrieval times by using both architectures is discussed. Finally, a conclusion about our work is made.

2 Domain of process planning

An example of a domain with structural dependencies between goals is the domain of process planning for manufacturing mechanical workpieces (Paulokat & Wess, 1994; Yang & Lu, 1994). A planning problem in this domain is to find a sequence of processing operations in order to machine a workpiece, by considering the available resources (i.e. tools, machines) and technological constraints relative to the use of these resources. The process begins by clamping a piece of raw material on a lathe machine that rotates it at a very high speed. Then different tools are used to remove layers of raw material. Depending on the structure of the workpiece, several clamping operations may be needed to process the workpiece completely.

Fig. 1 shows an example of a workpiece. The grid area corresponds to the portion of raw material that need to be removed. Based on the geometry of the workpiece the grid area is decomposed in several processing areas, some of which are indicated explicitely in Fig. 1, such as *hor* (a horizontal outline) and $ucut1$ (an undercut). The initial state of the planning problem is a collection of propositions regarding the resources available and propositions describing the relations between the processing areas. $rotary{-}cutting{-}tool(rct2)$ is an example of the first type of propositions, whereas $lies{-}below(ucut1, asc1)$ and

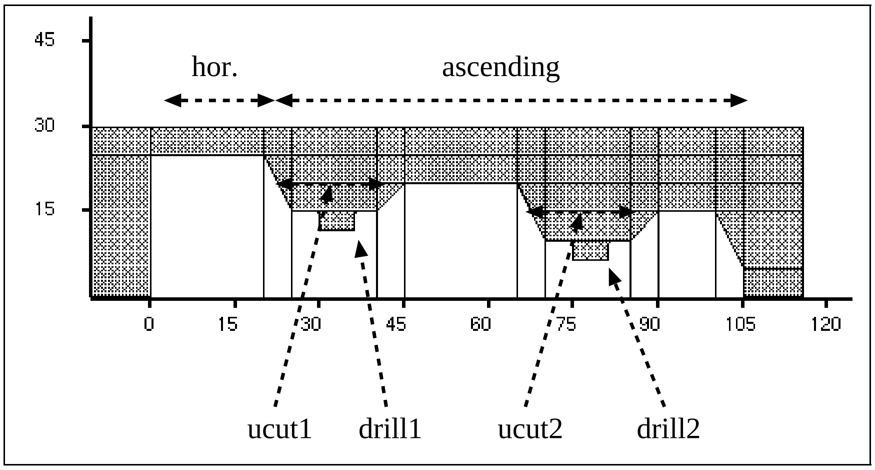

Fig. 1. Half display of a rotational symteric workpiece.

neighbour(*hor*, *asc*1) are examples of the second type. The goals of a planning problem are the processing areas encapsulated by the predicate *machined* (i.e. *machined*(*hor*), *machined*(*ucut*1)). Fig. 2 shows a part of a plan[2] for machining the workpiece shown in Fig. 1. There are two possible reasons for a plan-step s_1 to be ordered before another plan-step s_2: first, if applying s_2 makes impossible to apply s_1. For example *Clamp*(*hor*) is ordered after *machine*−*outline*(*hor*), because when the workpiece is clamped on *hor*, then it is impossible to machine *hor* as this area becomes unaccessible for the cutting tools. Second, when s_2 needs the effects of s_1 in order to be applied. For example, for machining the outline *hor*, the workpiece needs to be clamped on the outline *asc*1.

Notice that in Fig. 2 the processing areas are machined before the processing areas lain below them. For example, the undercut *ucut*1 lies below the ascending area *asc*1. Then in order to achieve the goal *machined*(*ucut*1) it is necessary to achieve the goal *machined*(*asc*1) first. This kind of dependencies between certain goals can be determined independent from any consideration of the tools or clamping operations needed to achieve them. So they are established before the planning process begins. Thus they can be used as an order-constraint for achieving the goals during the planning process or as an additional constraint that must be meet by cases in order to be retrieved during a case-base planning process. This observation is one of the motivations for the architecture of the case-base presented in this paper.

[2] In this plan other processing areas and the steps for mounting cutting-tools are omitted.

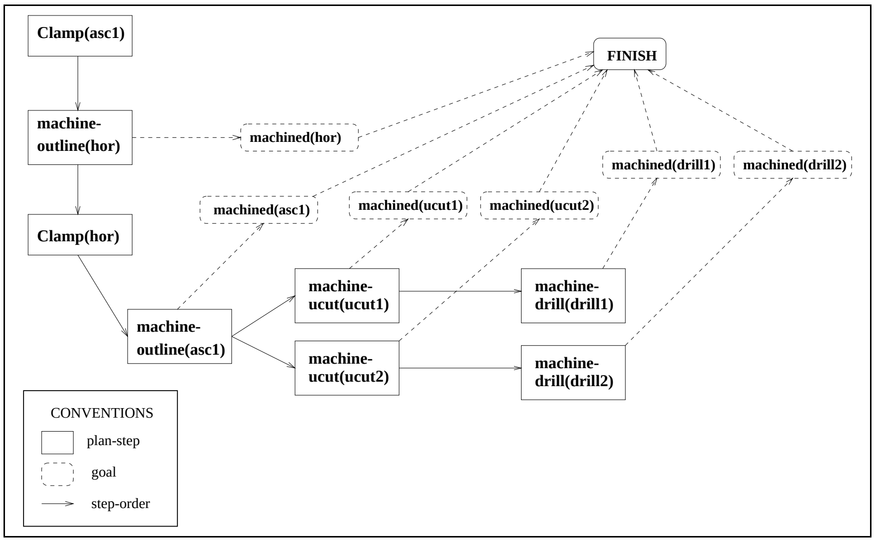

Fig. 2. Plan for machining a workpiece.

3 Domain Independent Case Retrieval

There are two basic restrictions that any retrieval procedure must meet: first, the cases retrieved must be relevant with respect to the planning problem (Veloso, 1994; Cunningham & Slattery, 1994). Second, the time to do this must be short enough, that the overall time for the case-based planner to solve a problem is shorter than the time that it takes to plan from scratch. We will refer to this restriction as the *time restriction*. In PRODIGY/ANALOGY an organization of the case-base has been proposed which enables the retrieval procedure to meet both restrictions. The basic idea is that after finding a solution plan that solves a problem, the problem description and the solution plan are stored together as a new case in the case-base. For indexing the case, an analysis of the problem description with respect to its solution plan is made. Based on this analysis a relation between goals is stated, which is defined as follows.

Definition 1 Interacting goals (Veloso, 1994). Given a solution plan that solves a problem and two goals g_1 and g_2 that appear in the problem. The goals $g1$ and $g2$ *interact* with respect to the solution plan, if $g1$ and $g2$ are achieved in the same connected component of the solution plan.

Consider the solution plan shown in Fig. 2. All the goals achieved in this plan interact with each other, as there is only one connected component in it with respect to the step-order. This is characteristic for the domain of process planning, although in other domains such as the logistic transportation domain and blocks world, plans may contain more than one connected component.

Giving a goal achieved in the plan, it is possible to perform goal regression (Veloso, 1994; Mitchell, Keller, & Kedar-Cabelli, 1986; Janetzko, Wess, & Melis, 1993) through the plan-steps in order to identify which propositions in the initial state had been used to achieved that goal, as stated by the following definition:

Definition 2 Foot-print of goals (Veloso, 1994). Given a solution plan that solves a problem. Then the foot-print of a goal g is constituted by the propositions in the initial state of the problem that contributed to achieve g. The foot-print of a set of goals is the union of the foot-prints for each individual goal in the set.

3.1 Organization of the Case-base

The case-base in PRODIGY/ANALOGY is organized as a three-level structure (Fig. 3). In the first level there are three access tables (not shown in Fig. 3). The second level is constituted by an initial-state discrimination network. Finally, the third level is constituted by the library of cases.

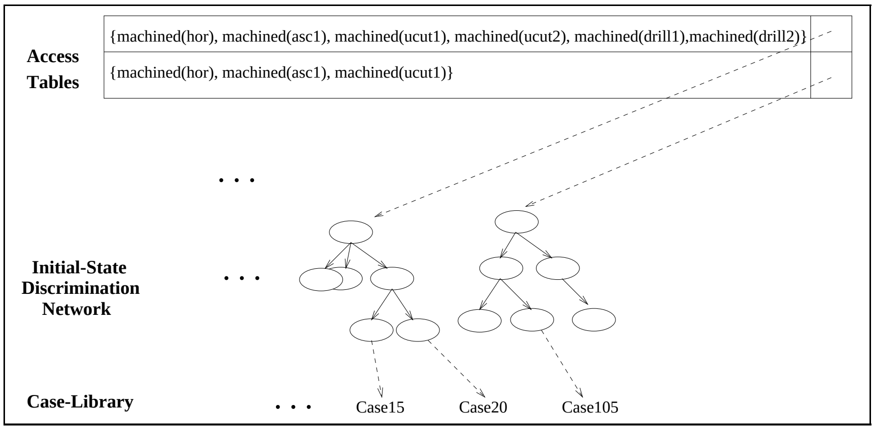

Fig. 3. Architecture of the case-base in PRODIGY/ANALOGY

The three access tables define entry points in the initial-state discrimination network, and their construction is based on the sets of interacting goals of each case stored in the case-base. Basically, these tables define a hash function that identifies for each set of goals, S, a tree in the initial-state discrimination network.

The initial-state discrimination network is constituded by a collection of trees. These trees serve to discriminate cases that have the same set of interacting goals.

The nodes of these trees contain sets of propositions and their leaves contain pointers to cases in the case-library. They are constructed in a way that by following any path from a leave to the root and collecting the set of propositions at each node, results in the foot-print for the set of interacting goals of the case pointed by the leave.[3]

3.2 Domain Independent Retrieval

The purpose of the retrieval phase is to find a collection of cases, so that their sets of interacting goals cover the new problem description. That is:

- For each goal in the new problem, there is one corresponding goal in one of the sets of interacting goals that matches it.
- The foot-print of each set of interacting goals matches the initial state of the new problem with a predefined accuracy. The accuracy indicates which percentage of the set of predicates need to be matched.

The strategy followed by PRODIGY/ANALOGY is to try to cover the new problem with as few cases as possible. So, at the first step, a case will be searched for that covers the whole problem description and more precisely that covers all the goals of the new problem. If this fails, PRODIGY/ANALOGY will try to find two cases: one that covers all the goals but one, and the other that covers the remaining goal. If this does not work, then the decomposition process will continue. At last PRODIGY/ANALOGY will try to cover each goal independ from the others.

Retrieving relevant cases and meeting the time restriction may be difficult. The matching process between sets of goals is one of the more expensive steps and it takes place each time the retrieval procedure is pursuing to cover a set of goals. The reason for this is the combinatorial factor involved when binding variables. However PRODIGY/ANALOGY has been tested massively in domains such as the logistical transportation domain, with good performances on the average. We argue that this performance may be threaten, when considering other classes of domains, where structural dependencies between goals need to be considered, and that by explicitely considering this dependencies between goals, the threat is reduced.

4 Dependencies between Interacting Goals

In some domains, it is necessary to consider the dependencies between goals that are included in the problem description. An example is the domain of process planning for mechanical workpieces, where the dependencies represent an order between steps for achieving certain processing areas. Thus, they establish

[3] Actually, they are not trees but graphs. The change was made for the sake of simplicity. There are other properties that characterize these trees (graphs), but they are not relevant for our discussion.

a partial order between the goals. So a problem description can be defined as follows:

Definition 3 Extended problem description. A problem description is constituted by an initial state, a set of goals to be achieved, and a partial order between the goals.

Comparing such a problem description against sets of interacting goals and their foot-prints will result in the selection of unadaptable cases, provided that a higher accuracy is not predefined. The reason for this is that these dependencies are hidden in the initial state of the problem and of the cases, so only high accurate matches between the initial states will ensure the consideration of all dependencies. But performing matches with high accuracy usually violates the time restriction of the retrieval procedure.

Before continuing, it is necesary to establish some conventions: a partial ordered plan $\mathcal{P}$ may be viewed as a pair $(\mathcal{S}, \rightarrow)$, where $\mathcal{S}$ represent the set of plan-steps of $\mathcal{P}$, and $\rightarrow$ the partial order of achievement between the plan-steps. For example, $Clamp(hor) \rightarrow machine{-}outline(asc1)$ is a plan-step order in the plan shown in Fig. 2. Given a goal g and a plan-step s_g, it is said that s_g achieves g if g is an effect of s_g. In the same way g is achieved by $\mathcal{P}$ if g is achieved by a plan-step in $\mathcal{P}$. So for example, the goal $machined(ucut2)$ is achieved by the plan-step $machine{-}ucut(ucut2)$. Given two plan-steps s_1 and s_2, the notation $s_1 \rightarrow^* s_2$, denotes: (1) $s_1 \rightarrow s_2$, or (2) there is a plan-step s, such that $s_1 \rightarrow s$ and $s \rightarrow^* s_2$ holds. For example $Clamp(hor) \rightarrow^* machine{-}ucut(ucut1)$. The notion of dependency between goals can be defined as follows.

Definition 4 Dependencies between goals. Let $\mathcal{P}$ be a plan and f and g two goals achieved by $\mathcal{P}$. Then f depends on g with respect to $\mathcal{P}$ if there are two plan-steps s_f, s_g in $\mathcal{P}$, such that s_f achieves f, s_g achieves g and $s_g \rightarrow^* s_f$.

For the plan presented in Fig. 1, the two machining-steps of the two undercuts $ucut1$ and $ucut2$ depend on the machining-step of the ascending outline $asc1$. It is easy to see that the dependency relation is strictly contained in the interaction relation: $machined(ucut1)$ and $machined(ucut2)$ interact, as they both are in the same connected component (Fig. 2). However, none of them depend on the other one. We used this refinement to construct an intermediate level between the access tables and the initial state network in order to represent explicitely dependencies between goals.

5 Indexing and retrieving cases in CAPlan/CbC

Our approach consists of using the dependencies between goals in order to further structure the case-base. In this way irrelevant cases are rapidly discarded, enabling the retrieval procedure of CAPlan/CbC to concentrate on a small portion of the case-base.

5.1 Architecture of the Case-Library

The case-base in CAPLAN/CBC is a four-level structure (fig. 4). There are three major differences with respect to the one presented in PRODIGY/ANALOGY (section 3.1): first, rather than explicitely representing the set of interacting goals in the access tables, chains of goals are used to conform a goal discrimination network. Usually an arc between two nodes in the chains of goals represents the dependency order, as for example the arc labeled with A in fig. 4. However, some arcs may represent an order that extends the dependency order, as for example the arc labeled B in fig. 4. The reason for this is that some information is lost due to the representation of the dependency order (which is a partial order) on chains of goals that are totally ordered. Even with this loss of information, the case-base is further structured, which results in better retrieval times as will be shown in section 6. Thus, the accuracy of the match can be improved, resulting in the retrieval of relevant cases.

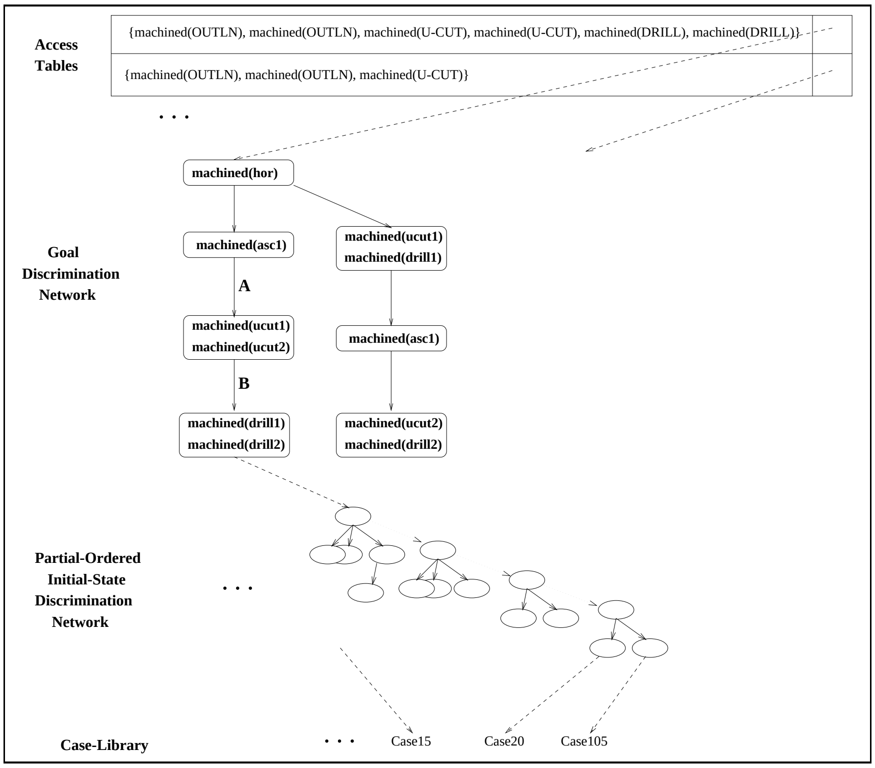

Fig. 4. Case-library in CAPLAN/CBC

The second difference is that the access tables contain only the class representation[4] of sets of interacting goals. The third difference is that the initial state discrimination network has been partially ordered through chains of threes. These chains serve to discriminate between cases that have the same representation in the chain of goals. Each node in the goal discrimination network points to a tree in the partially ordered discrimination network (not shown in figure 4). This initial-state tree contains the foot-print of the set of predicates included in the node. Only the leaves of the last initial-state tree in each chain point to cases in the case-library. Every link of each initial-state tree is labeled with one or more names of cases (not shown also in figure 4). These cases correspond to the ones pointed by the nodes in the last tree of the chain. Collecting the sets at each node that is pointed by a link labeled with the name of a case, results in the foot-print for the interacting goals of the case.

This organization is not well suited for domains without structural dependencies between their goals. For example, in the logistic transportation domain (Veloso, 1994), it is easy to state problems, so that for any combination of the goals, it is possible to obtain a plan that achieves the goals in the order stated through the combination. Thus the goal discrimination network tends to be very large for the same set of interacting goals. Further, no dependencies between goals can be predefined (we are not considering efficiency issues here), so there is no advantage of having the dependencies explicit in the case-base.

5.2 Retrieval of relevant cases in CAPLAN/CBC

CAPLAN/CBC follows the same strategy as PRODIGY/ANALOGY in order to retrieve relevant cases for the new problem: cover the goals with as few cases as possible and match the initial state with a given accuracy.

For finding a case that covers a partially ordered set of goals $(G_{prob}, \rightarrow_{prob})$, CAPLAN/CBC uses the access tables to identify a class representation of a set of interacting goals that is equal to G_{prob}. If none is found then there is no case that covers $(G_{prob}, \rightarrow_{prob})$ and the procedure terminates. Otherwise, a tree, $goal{-}tree$, in the goal discrimination network is accessed that corresponds to the class representation of G_{prob}. As we saw, $goal{-}tree$ is a tree representation of goal chains, where arcs between nodes represent the dependency order or an extension of it. So, a search on $goal{-}tree$ is made, looking for a chain that is consistent with the dependencies of the new problem, $\rightarrow_{prob}$. If no such chain is found, the procedure returns with a failure. Otherwise, a path from the root to a leave in $goal{-}tree$ has been identified that matches G_{prob} and that is consistent with $\rightarrow_{prob}$. Each node in the path contains a pointer to a tree in the partially-ordered initial state network. For the initial-state tree pointed by the root of $goal{-}tree$ the procedure identified the set of cases, $Cases$, that matches the

[4] The class representation (Veloso, 1994) of a set of predicates is obtained by replacing the arguments of each predicate with their types. For example, the class representation of $\{machined(ucut1)\}$ is $\{machined(U-CUT)\}$. Comparing class representations is not expensive, as the arguments of the predicates are constants.

initial state with the given accuracy. This is possible because all arcs in the initial-states trees are labeled with names of cases. This process is repeated recursively for all nodes of $goal{-}tree$ in the path, excluding cases in $Cases$ that do not meet the accuracy predefined. If at some point $Cases$ is empty, then CAPlan/CbC backtracks and continues the search in the goal tree. Otherwise, the leave of the path will be reached and one of the cases contained in $Cases$ is returned.

6 Empirical Results

We implemented the architectures of the case-base presented in Prodigy/Analogy (Veloso, 1994) and the one proposed here, integrating them in our case-based planning system CAPlan/CbC, which is implemented in Smalltalk-80. As a result, the basic operations such as matching between propositions, and the implementations of the basic data types such as set of propositions are the same for both architectures, so none of them takes advantage of the other based on implementation details. In both case-bases the same cases from the workpieces domain where added. In the experiment 60 new problem descriptions corresponding to complete descriptions of workpieces were given. We meassured the time that it took in each architecture to retrieve a case with an accuracy of 75%. The reason for retrieving just one case is that plans in this domain contain only one connected component, so no decomposition of the problem description based on interacting goals is possible. The results are shown in figure 5. The first 10 problems contained 7 goals, the next 10 problems contained 8 goals, and continuing with the same proportion, the last 10 problems contained 12 goals. Notice that for the last 20 problems, the increase in running time by using the architecture of Prodigy/Analogy is higher than the one by using the architecture of CAPlan/CbC. Thus, as the number of goals increases the architecture of CAPlan/CbC leads to better retrieval times. There are two reasons for this: first, in Prodigy/Analogy a matching between the sets of goals takes place, whereras in CAPlan/CbC the match is between partially ordered collections of goals. Second, the initial state network in CAPlan/CbC is constructed in a way, that the sets of predicates in the nodes tend to be smaller than in Prodigy/Analogy. More precisely, the initial-state trees in CAPlan/CbC store only foot-prints of small subsets of interacting goals, whereas in Prodigy/Analogy these trees store foot-prints of all the interacting goals. As a result, matches in CAPlan/CbC involve small sets of propositions only.

7 Conclusion

We have shown that the architecture of the case-library implemented in CAPlan/CbC ensures the adaptability of the cases retrieved for structured domains. In this class of domains it is necessary to consider structural dependencies

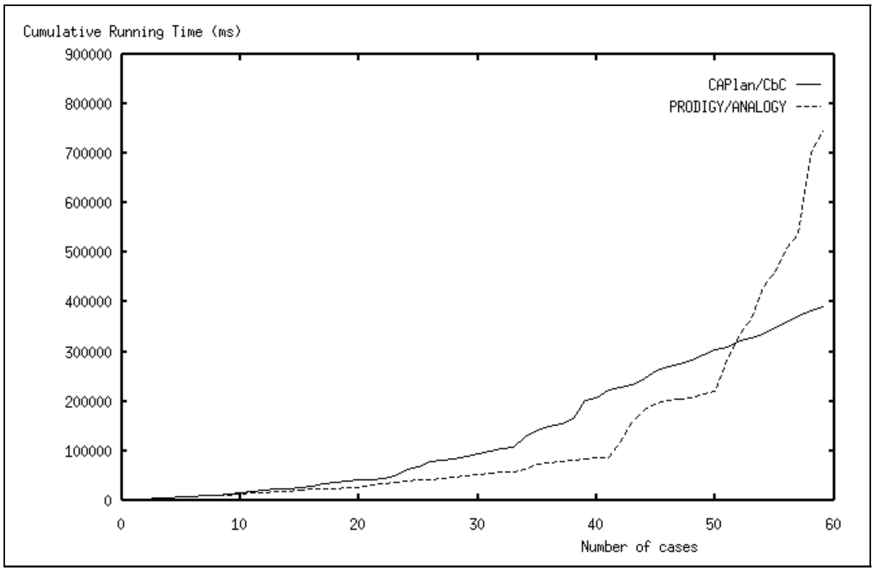

Fig. 5. Cumulative running times of the retrieval procedure by using the architectures of CAPLAN/CBC and of PRODIGY/ANALOGY

in the problem description in order to retrieve relevant cases during a case-base planning process. The architecture presented here is an extension of the one proposed in PRODIGY/ANALOGY, in that the concept of interacting goals has been refined to consider the dependencies between interacting goals. Based on this refinement the foot-print has also been partitionated. This partition is explicitely represented in the case-base and leads to an increased performance of the retrieval process.

Acknowledgements

The authors want to thank Michael M. Richter, Jürgen Paulokat and Frank Weberskirch for their contributions as well as Jürgen Paulokat, Ralph Bergmann and the reviewers for helpful comments on earlier versions of this paper.

References

Bergmann, R. (1995). Building and refining abstract planning cases by change of representation language. To appear in *Journal of AI Research.*

Carbonell, J. (1983). Derivational analogy in problem solving and knowledge acquisition. In *Proceedings of the 2nd International Workshop on Machine Learning.* University of Illinois, Monticello, Illinois.

Cunningham, P., & Slattery, S. (1994). Knowledge engineering requirements in derivational analogy. In *(Richter, Wess, Althoff, & Maurer, 1994)*.

Ihrig, L. H., & Kambhampati, S. (1994). Derivational replay for partial-order planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94*. The AAAI Press/The MIT Press.

Janetzko, D., Wess, S., & Melis, E. (1993). Goal-Driven Simmilarity Assessment. In Ohlbach, H.-J. (Ed.), *GWAI-92: Advances in Artificial Intelligence*, Springer Verlag, pp. 283–298.

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

Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, *1*, 47–80.

Muñoz-Avila, H., Paulokat, J., & Wess, S. (1994). Controlling a nonlinear hierachical planner using case-based reasoning. In Keane, M., Halton, J. P., & Manago, M. (Eds.), *Proceedings Second European Workshop on Case-Based Reasoning, EWCBR-94*.

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

Richter, M., Wess, S., Althoff, K., & Maurer, F. (Eds.). (1994). *First European Workshop on Case-Based Reasoning (EWCBR-93)*. No. 837 in Lecture Notes in Artificial Intelligence. Springer Verlag.

Smyth, B., & Keane, M. T. (1994). Retrieving adaptable cases: the role of adaptation knowledge in case retrieval. In *(Richter et al., 1994)*.

Veloso, M., & Carbonell, J. (1993). Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. *Machine Learning*, *10*.

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

Yang, H., & Lu, W. F. (1994). Case adaptation in a case-based process planning system. In Hammond, K. (Ed.), *Proceedings of The Second International Conference on Artificial Intelligence Planning Systems, AIPS-94*. The AAAI Press.