In: S. Steel & R. Alami (Eds.): Recent Advances in AI Planning. Proceedings of ECP-97, Springer, 1997

A Case Study on Mergeability of Cases with a Partial-Order Planner

Héctor Muñoz-Avila & Frank Weberskirch

University of Kaiserslautern, Dept. of Computer Science
P.O. Box 3049, D-67653 Kaiserslautern, Germany
E-mail: {munioz|weberski}@informatik.uni-kl.de

Abstract. Retrieving multiple cases is supposed to be an adequate retrieval strategy for guiding partial-order planners because of the recognized flexibility of these planners to interleave steps in the plans. Cases are combined by merging them. In this paper, we will examine two different kinds of merging cases in the context of partial-order planning. We will see that merging cases can be very difficult if the cases are merged eagerly. On the other hand, if cases are merged by avoiding redundant steps, the guidance of the additional cases tends to decrease with the number of covered goals and retrieved cases in domains having a certain kind of interactions. Thus, to retrieve a single case covering many of the goals of the problem or to retrieve fewer cases covering many of the goals is at least equally effective as to retrieve several cases covering all goals in these domains.

1 Introduction

In case-based reasoning (CBR) systems, the performance of the adaptation phase depends on the adequacy of the retrieval criteria [Koehler, 1994] [Francis and Ram, 1995b]. For example, if CBR is used to guide the search process of a domain-independent planner, inadequate retrieval criteria may result in a lower performance of the problem solving process compared to first-principles planning in state-space [Veloso, 1994] and partial-order planning [Hanks and Weld, 1995] [Francis and Ram, 1995a] [Ihrig and Kambhampati, 1996].

Retrieving several cases covering different parts of the problem is a recurrent idea in case-based planning (CBP) [Veloso, 1994] [Francis and Ram, 1995a] [Ihrig and Kambhampati, 1996]. Cases are combined by *merging* them. A general definition of merging has been presented in [Kambhampati et al., 1996]. The basic idea is to replay decisions of the retrieved cases independent of the decisions replayed from the other cases. These independently replayed cases are completed into a solution by the first-principles planner. We will see that this form of merging, which we called *eager merging*, is not too useful for a partial-order planner like SNLP [McAllester and Rosenblitt, 1991]. We will analyze the reasons for this and provide experiments supporting our analysis.

A more elaborated merging method is to consider the replayed decisions to avoid adding steps that already have been introduced [Veloso, 1994,Ihrig and Kambhampati, 1996]. In this way redundancy in a plan is reduced. This method, which we called *non-redundant merging*, has been shown to be effective in guiding the planning process. However, we will see that in domains having a certain kind of interactions, the guidance tends to decrease with problem size. Thus, to retrieve a single case covering much of the goals of the problem or to retrieve fewer cases covering much of the goals is at least equally effective as to retrieve several cases covering all goals in these domains.

The paper is organized as follows: first, we briefly recall the SNLP paradigm and present a domain used throughout the paper. Section 3 discusses eager merging in the context of partial-order planning. The next section presents non-redundant merging and a first observation about this method. Section 5 defines a certain form of interactions and analyzes its effects on non-redundant merging. Then, empirical results are presented. Finally, related work is discussed and conclusions are made.

2 Partial-Order Planning and an Example Domain

In this work we will concentrate on a CBP scenario where we have a partial-order planner, SNLP, as the base level planner. Formally, a partially ordered plan [McAllester and Rosenblitt, 1991] is a 4-tuple (S, L, CL, B), where S is the set of plan steps, L is a set of *links* between steps in S, CL is the subset of L containing the causal links, and, B is a set of *constraints* on the variables bindings. L contain causal links and ordering links and induces a partial order, $<_L$, for executing steps in S. A causal link, $s_1 \rightarrow p@s_2$, captures causal dependencies between steps by indicating the step s_1 achieving a precondition p of another step s_2. Ordering links are added to solve threats. A *threat* to a causal link $s_1 \rightarrow p@s_2$ occurs if there is a third plan step s_3 that has as effect p or $\neg p$ and that is parallel to $s_1 \rightarrow p@s_2$, that is, neither $s_3 <_L s_1$ nor $s_2 <_L s_3$ holds. If the effect of s_3 is p the threat is said to be *positive* otherwise it is said to be *negative*. Positive threats must be solved to avoid redundancy in plans in that the same goal is achieved by two parallel steps. Negative threats must be solved to ensure the consistency of the plan.

Planning with SNLP [McAllester and Rosenblitt, 1991] proceeds by *establishing open conditions* and *resolving conflicts*. To establish an open precondition, a step in S is selected or a new step is added to S such that one of its effects matches the open precondition. For each establishment of a precondition a causal link is added. To solve a threat to a causal link $s_1 \rightarrow p@s_2$ caused by a step s_3, an ordering link is added such that $s_3 <_L s_1$ or $s_2 <_L s_3$.[1].

[1] Originally, SNLP proposed another variant to solve threats called *separation*. With separation, binding constraints on the variables are added that make it impossible to match the effect of s_3 with the precondition $p@s_2$. However, [Smith and Peot, 1993] show that ignoring separable threats does not affect correctness or completeness of

Example Domain. A domain frequently referred to in the planning literature is the domain of process planning (see [Hayes, 1987], [Kambhampati et al., 1991], [Gil, 1991], [Karinthi et al., 1992], [Britanik and Marefat, 1995], [Nau et al., 1995], [Muñoz-Avila and Weberskirch, 1996a]), particularly the manufacturing process of mechanical workpieces. To manufacture a workpiece, several so-called machining operations are performed to transform a given piece of raw material into design specifications. Typically, a piece of raw material is clamped on a machine and a cutting tool is used to remove layers of raw material. The process continues by changing the cutting tool or the clamping position to machine areas of the workpiece. A basic restriction is that at any time of the machining process the workpiece is *clamped* from at most one position and a limited number of cutting tools can be *held*. In the specification used in [Muñoz-Avila and Weberskirch, 1996b], an order, $\prec$, for achieving the machining goals can be predefined based on the geometry of the workpiece [Muñoz-Avila and Weberskirch, 1996a]. Thus, problem descriptions consist not only of the initial and final state, (I, F), but of the order $\prec$. The 3-tuple $(I, F, \prec)$ is called an *extended problem description*. The order $\prec$ can be seen as an additional restriction that any solution must meet.

3 Eager Merging of Cases

Eager replay has been shown to be an adequate reuse strategy for partial-order planners such as SNLP [Ihrig and Kambhampati, 1994]. Reuse with *eager replay* is done in two phases. First, the decisions taken in the retrieved case are replayed in the new situation as long as they don't produce inconsistencies. Once this phase is finished, a so-called *skeletal plan* is obtained that may contain open preconditions and threats. Then, the skeletal plan is completed by first-principles planning.

When several cases are retrieved, the question of how to merge them arises. A first approach is to replay each retrieved case and let the planner do the merging. We called this *eager merging* as it corresponds to the straightforward use of eager replay for multiple cases. To examine the limitations of eager merging we recall the definition of mergeability of plans [Kambhampati et al., 1996]:

Definition 1 (Mergeability of Plans). Given a plan P_1 for achieving a goal g_1. P_1 is **mergeable** with respect to a plan P_2 for achieving a goal g_2 if there is a plan P extending P_1 and P_2 achieving both g_1 and g_2.

In addition, the plans are said to be **simple mergeable** if every step in P is present in either P_1 or P_2 and the number of steps in P is equal to the number of steps in P_1 and P_2 (that is, only ordering links are added).

The definition of mergeability does not exclude that backtracking takes place in finding P. The point is that no backtracking should take place in the plan

SNLP. Additionally, the search space without separation is at most the same as with separation.

refinement steps that added plan steps, links, or constraints in P_1 or in P_2. In CBP, a maximal gain is expected if a case base is given that is constituted of mergeable plans. If the cases can be replayed totally in the the context of the new problem, the mergeability condition ensures that the planner will not need to revise decisions taken in the cases during the completion phase. It may not always be possible to replay the cases completely, however if the cases are small in terms of the number of goals solved (e.g., one-goal cases), this situation is likely to occur. The strategy of storing small cases has been explored before to decrease the size of the case base [Ihrig and Kambhampati, 1996].

Mergeability turns out to be a strong requirement. A first indication of how strong this requirement is can be derived from the following definition:

Definition 2 (Parellelizability of Goals). Given two goals g_1 and g_2. If any two plans P_1 and P_2 achieving g_1 and g_2 are mergeable, then g_1 and g_2 are said to be **parallelizable**.

[Kambhampati et al., 1996] show that if a set of goals is parallelizable, then the goals are trivially serializable[2] and that the opposite direction does not hold. That is parellelizability is a stronger requirement than trivial serializability. We found that in the context of partial-order planning mergeability is indeed a very restrictive condition:

Proposition 3. *Suppose that first-principles planning is done with SNLP. Then, a plan P_1 for achieving a goal g_1 is mergeable with respect to a plan P_2 for achieving a goal g_2 if and only if P_1 and P_2 are simple mergeable.*

Proof. Planning in SNLP proceeds by solving threats and establishing open (unachieved) preconditions. New steps are added to a plan, only to achieve open preconditions. Because P_1 and P_2 are complete plans (they achieve g_1 and g_2), only threats may occur (interactions between P_1 and P_2). Thus, P_1 and P_2 can be extended to a plan achieving g_1 and g_2 if and only if the threats can be solved. That is, if only ordering links need to be added to extend these plans.■

This simple result shows that mergeability is a very strong requirement for partial-order planning with SNLP. For example, as explained before, in the domain of process planning there are always interactions between two machining plans because the use of the cutting tools needs to be rationalized. In the specification used [Muñoz-Avila and Weberskirch, 1996b], not only ordering links must be added between the holding steps to solve the threats, but new steps must be introduced as well for performing the operation that unmounts the tool from the holding machine, *MakeToolHolderFree*. The same situation occurs if the specification of [Gil, 1991] is used (the unmount operation is called *Release-from-holding-device* there). As a result, proposition 1 shows that machining plans are

[2] Given two goals g_1 and g_2, if any plan achieving g_1 can be extended to a plan achieving g_1 and g_2, then g_1, g_2 is said to be a *serialization order*. If g_1, g_2 and g_2, g_1 are serialization orders then g_1 and g_2 are said to be *trivially serializable* [Korf, 1987,Barrett and Weld, 1994,Kambhampati et al., 1996].

not mergeable for SNLP. In the logistics transportation domain [Veloso, 1994], it is easy to find situations, where new steps must be added to extend the cases.

4 Non-Redundant Merging of Cases

A disadvantage of eager merging is that the solution obtained may be very large because several steps are repeated unnecessarily. Therefore, another form of merging, which we denote as *non-redundant merging*, has been proposed [Veloso, 1994,Ihrig and Kambhampati, 1996]. During the replay phase, opportunities to establish preconditions are considered in the following way: before replaying a step to establish an open precondition, the system checks if the precondition can be established with a step in the current subplan (i.e., the subplan already obtained from the cases replayed before). If this is possible, the step is not replayed and the precondition is left open. During the completion process, the first-principles planner prefers to establish the conditions left open by the replay process by using the available steps. New steps are added only if no completion of the plan by using the steps available is possible. If more than two cases are involved, the process is done stepwise: the first two are merged and extended to obtain a complete subplan (i.e., a subplan containing no open preconditions and no threats). The third case is then merged with the current subplan and so on. The rationale behind non-redundant merging is to take advantage of opportunities to establish the open preconditions using steps that were not available because the cases were solved separately.

Example. The way the interactions between the goals affect the non-redundant merging process is illustrated with the example in figure 1 (*cs*, *hs* and *ms* denote clamping, holding and machining steps respectively): it shows two subplans for *machined(ctr)* and *machined(ucut)* when the non-redundant form of merging was used and the subplan for *machined(ctr)* was generated before the subplan for *machined(ucut)* (threats are depicted with double arrows). A step *cs(left)* was not replayed (depicted with a dashed box) because another step *cs(left)* in the other subplan establishes the precondition *clamped(left)*. As a result, the part of the case achieving the preconditions of *cs(left)* is not replayed, too.

Based on the non-redundant form of merging plans, *non-redundant mergeability of plans*, the notion of *simple non-redundant mergeability of plans* and *non-redundant parellelizability of goals* can be defined. We define non-redundant mergeability of plans for illustration purposes:

Definition 4 (Non-Redundant Mergeability). Given two plans P_1 and P_2, then $P_2 \setminus P_1$ denotes the plan that remains after pruning plan steps from P_2 achieving goals that can be established by using steps in P_1. Plan steps that were added to achieve a precondition of a pruned step, are pruned as well.

Given plan P_1, P_2 for achieving the goal g_1, g_2. Plan P_1 is *non-redundant mergeable* with respect to P_2 if there is a plan P extending P_1 and $P_2 \setminus P_1$ achieving both g_1 and g_2.

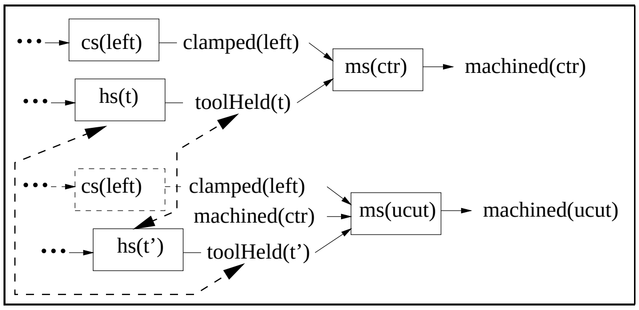

Fig. 1. Interactions between two subplans.

Clearly, the mergeability of plans implies their non-redundant mergeability and the parellelizability of goals implies their non-redundant parellelizability. Non-redundant merging is more flexible than eager merging. For example, in the specification of the domain of process planning used, if the machining goals are achieved in an order consistent with $\prec$, the plans are non-redundantly mergeable even though they are never mergeable. The two subplans depicted in Figure 1 are non-redundant mergeable: the precondition *clamped(left)* can be established with the existing step *cs(left)* and an unmount step can be added an ordered between *hs(t)* and *hs(t')*. Machining goals are also $\prec$-constrained trivially serializable [Muñoz-Avila and Weberskirch, 1996b].[3] However, examples can be found of goals that are trivially serializable and not non-redundantly parellelizable. We will now show that non-redundant parallelizability is also an stronger requirement than trivial serializability.

Proposition 5. *If a set of goals g_1, ..., g_n is non-redundant parallelizable, then it is trivially serializable.*

Proof. We will prove this result for two goals. Let P_1 be a plan achieving g_1. We will show that P_1 can be extended to a plan achieving g_1 and g_2. Let P_2 be any plan achieving g_2 and SEQ_2 the sequence of refinement steps to obtain P_2. Let $SEQ_{P_2 \setminus P_1}$ be the subsequence of refinement steps in SEQ_2 achieving $P_2 \setminus P_1$. Because, g_1 and g_2 are non-redundantly parallelizable, there must be a plan P_3 extending P_1 and $P_2 \setminus P_1$ that achieves g_1 and g_2. Let SEQ_3 be the sequence of refinement steps to obtain P_3 by extending P_1 and $P_2 \setminus P_1$. Then, P_1 can be extended to a plan achieving g_1 and g_2 by following the sequences of refinement steps $SEQ_{P_2 \setminus P_1}$ and then SEQ_3. This shows that g_1, g_2 is a serialization order. In the same way it can be shown that g_2, g_1 is also a serializiation order. Thus, g_1, g_2 are trivially serializable. ■

[3] $\prec$-constrained trivially serializable is a restricted form of trivial serializability for extended problem descriptions, $(I, F, \prec)$, in which only orders consistent with $\prec$ are taken into account (and not all the permutations of the goals).

5 Positive Interactions and Non-redundant Merging

When following the non redundant merging method, it can be observed that once the replay phase is finished, no *positive* threats between the current subplan (i.e., the subplan obtained after merging the previous cases) and the replayed case occur. The reason for this is that positive threats indicate that the same effect has been obtained in the subplan and the replayed case, which is the kind of redundancy that non redundant merging avoids. In the example in Figure 1, which only involves two goals, one of the subgoals of the machining step achieving the first goal is left open and as a result a significant portion of the case is not replayed. The kind of interactions occuring can be formalized as follows:

Definition 6 (Conflict between goals). A set of goals G is said to be *in conflict*, if for every two subplans P_i and P_j achieving g_i and g_j in G, there is a step in P_i that threatens a causal link in P_j and vice versa.

We also say that P_i and P_j interact. If there are positive (negative) threats, the interactions are said to be positive (negative).

In the domain of process planning, machining goals are *always* in conflict. Negative interactions between subplans indicate violations to the restrictions on the clamping operation (i.e., the workpiece can only be clamped from one position at a time). In the specification used in [Muñoz-Avila and Weberskirch, 1996b], only one tool can be held at a time. Thus, there are always at least two interactions between machining subplans, one caused by the clamping steps and the other one caused by the holding steps. Given that in this specification any workpiece can be clamped by using at most 6 different clamping operations, there are always positive interactions when at least 7 cases are merged. We will see in the experiments that as the number of positive interactions increases, the guidance provided by the additional cases tends to be small as more cases are merged and more goals are covered.

Notice that in the logistics transportation domain goals might not be in conflict. For example, a subplan in which a truck picks an object in one location, moves to other location and drops the object does not interact with a subplan performing the same operations in a different city.

6 Evaluating the Merging Strategies

We performed experiments to evaluate the different merging strategies in partially ordered plans based on SNLP.

Domains: We performed experiments with the domain of process planning, an artificial domain[4] and the logistics transportation domain.

[4] The artificial domain is an extension of the ART-1D-RD [Kambhampati, 1993], in that two new operators were added that rationalize the use of the resources hf and he instead of adding and deleting them directly in the actions A_i. This domain is similar to the domain of process planning in that goals are in conflict.

Problems: We constructed a sequence of single-goal problems (15 for the domain of process planning, 12 for the transportation domain and 8 for the artificial domain). To observe the way the positive interactions affect the merging process, sequences of $n+1$-goal problems were constructed by adding a goal randomly selected from the sequence of single-goal problems to each problem of the sequence of n-goal problems. The constructed problems were revised to avoid repetitions. In this way, we ensured that when several cases were retrieved, all the goals in the problem are covered. The construction of the case base reflects an ideal situation because in practice it is unlikely that all the goals of the problem are covered. However, this situation is appropiate to compare the merging methods because a maximal gain is expected from CBP.

Retrieval and adaptation: The similarity assessment used was the foot-printed similarity metric and the case base was constructed in the way of Prodigy/Analogy [Veloso, 1994]. For each problem, the retrieval strategy followed was to retrieve one-goal cases such that all goals of the problem are covered. The retrieved cases were merged in eager-merging and non-redundant-merging modes. For the domain of process planning the goals were merged in an order consistent with $\prec$.

The data obtained with pure SNLP is intended as a reference to compare the merging algorithms and not to show the advantages of CBP over first-principles planning. [Veloso, 1994,Ihrig and Kambhampati, 1996] already observed benefits of CBP over first-principles planning. As explained before, the construction of the case base was biased towards obtaining maximal gains with CBP.

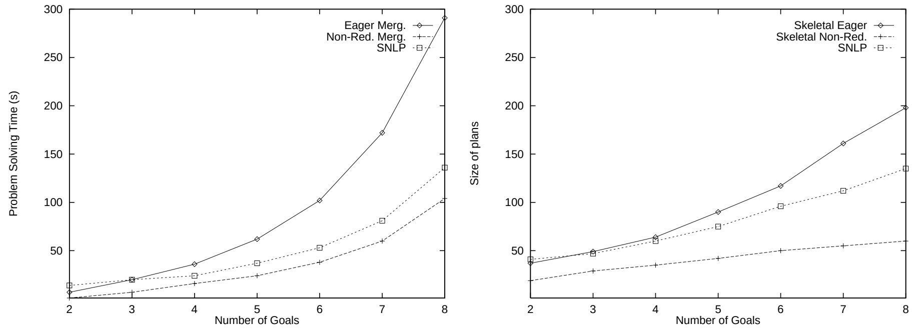

Fig. 2. (a) Problem solving time and (b) plan size for the domain of process planning

Results. Figures 2, 3 and 4 compare data obtained with the domain of process planning, the artificial domain and the logistics transportation domain. Part (a) of these figures shows the problem solving time by using eager and non-redundant merging and SNLP. Parts (b) shows the size (i.e., $S + L$) of the skeletal plans

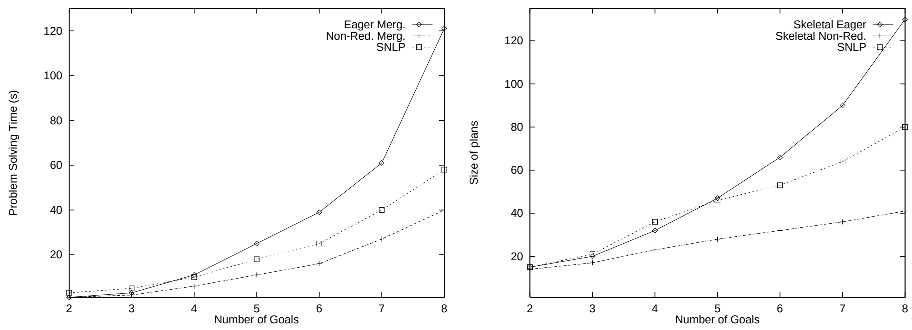

Fig. 3. (a) Problem solving time and (b) size for the artificial domain

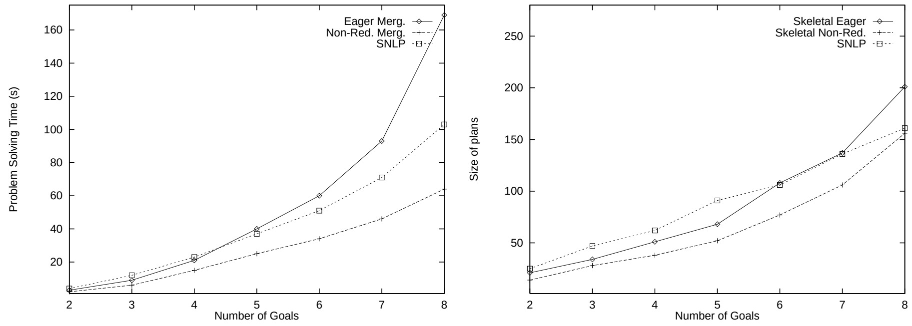

Fig. 4. (a) Problem solving time and (b) skeletal plan size for the logistics transportation domain

obtained with eager and non-redundant merging and the size of the solution plans obtained by SNLP.

Common to the three domains is that the worst performance is obtained with the eager merging mode (it is outperformed by SNLP). Eager merging seems to be an inadequate choice when the base level planner is SNLP corroborating in this way Proposition 1. In the domain of process planning and the artificial domain this result reflects the fact that subplans are always nonmergeable. In the transportation domain some of the skeletal subplans generated were mergeable. Thus, the performance with the eager merging mode is slightly better compared to the other domains. With the non-redundant mode, the difference between the size of the skeletal plan and the solution plan is a measure for the effort needed in the completion phase. For example, the effort for completition in the transportation domain (see Figure 4 (a)) is less than in the other two domains because the skeletal plans are comparatively larger (see Figure 4 (b)).

Figure 5 compares (a) the number of positive interactions and (b) the total number of interactions occuring in these domains. Because no positive interactions can occur after non redundant merging, the positive interactions are mea-

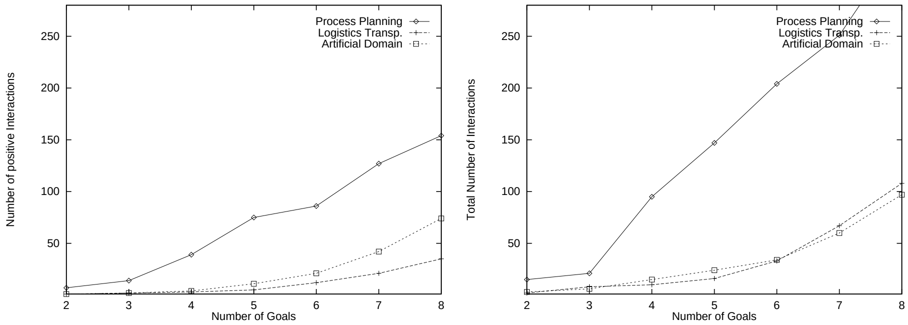

Fig. 5. (a) Number of positive interactions and (b) total number of interactions

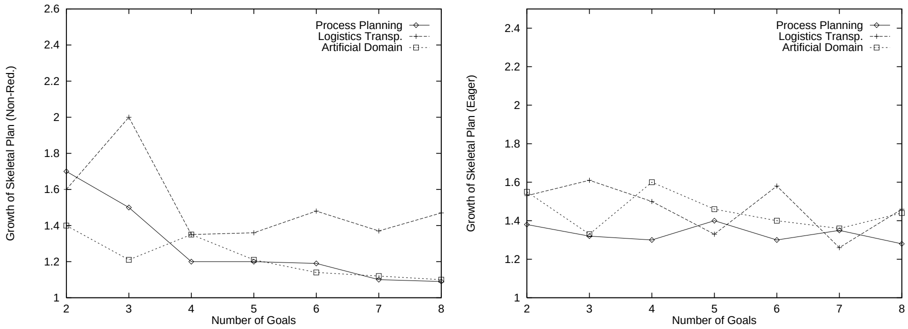

Fig. 6. Growth of skeletal plan size with the (a) non-redundant and (b) eager merging

sured by observing the eager-merging mode. The difference between the skeletal plan in eager-merging mode with the skeletal plan in non-redundant-merging mode reflect the percentage of the cases that was not replayed because of the positive interactions. Particularly, in the domain of process planning and the artificial domain, it can be observed that for solving the 8-goal problems, a significant part of the cases was not replayed (more than 70%). This is the result of the positive interactions, which in both situations correspond to approximately 50% of the interactions (see Figure 5).

Related to this issue, notice that as the number of goals increases, the guidance provided by the additional cases retrieved with the non-redundant merging mode decreases in the domain of process planning and the artificial domain. Figure 6 (a) compares the relative growth of the plans in the three domains for non-redundant merging whereas part (b) compares this growth for eager merging. The relative growth for n goals was measured by dividing the average size of the plans for n goals by the average size of the plans for n-1 goals. It can be observed that for the process planning domain and the artificial domain the relative growth of the skeletal plan generated in non-redundant mode decreases with the number of goals (see Figure 6). Notice that for both domains there is a significant increase in the number of positive interactions (see Figure 5). This supports our claim that in domains were goals are in conflict, the guidance provided by the additional cases retrieved tends to decrease with the number of goals. These results suggest that in these domains retrieving a single case covering as much of the goals as possible or fewer cases is an equivalent strategy because merging additional cases is worthless after several goals have been solved. Notice, that the growth of the skeletal plans obtained with eager merging does not decrease with the number of goals.

In contrast to the other two domains, in the logistics transportation domain the guidance provided by the additional cases with non redundant merging does not decrease as more goals have been solved and correspondly the number of positive interactions does not increase in a significant way. This shows that in domains were goals are not in conflict, retrieving several cases and using non-redundant merging is indeed an adequate choice.

7 Related Work

The idea of avoiding redundancy during replay of multiple cases was first proposed in Prodigy/Analogy [Veloso, 1994]. Prodigy/Analogy uses a mixed-initiative strategy to switch the search control between case-based and first-principles. The basis for this strategy is the fact that Prodigy searches in the space of states instead of the space of plans as in our work. Selecting the kind of base-level planner depends on the characteristics of the particular domain [Kambhampati et al., 1996,Barrett and Weld, 1994,Veloso and Blythe, 1994]. For the specification of the domain of process planning being used here, there is theoretical evidence that a partial-order planner such as SNLP is a better choice [Muñoz-Avila and Weberskirch, 1996b].

derSNLP+EBL [Ihrig and Kambhampati, 1996] is a cased-based planner searching in the space of plans. derSNLP+EBL applies the following strategy: in principle, several single-goal cases are retrieved, each covering a goal of the problem. Non redundant merging is used to combine them. If decisions in the subplans achieving the goals need to be revised to obtain a solution, this solution is stored as a new multi-goal case. If in future retrieval episodes the same situation is encountered, the multi-goal case is retrieved instead of the single-goal cases. Based on the results of this paper, we affirm that in domains where goals are in conflict this method results in an improvement of the performance of the planning process as the result of the multi-goal cases learned but not of the merging method itself. As the number of goals increases, the process of merging cases serves mainly to construct multi-goal cases.

Although not in the context of partial-order planning, previous work has shown that merging subplans into a solution is NP-complete [Karinthi et al., 1992,Yang et al., 1992]. The same work, however, shows that there are instances of the problem that can be solved in polinomial time. An algorithm for merging is presented containing several operations, one of which involves merging the same step occuring in different plans into a single step. This operation is comparable to prefering existing establishing oportunities of the non redundant merging method.

8 Conclusion

We explored eager and non redundant merging in the context of CBP with a partial-order planner. We have seen that eager merging has a very limited applicability because of the strong requirements that it makes on the cases so that they can be combined. In particular for SNLP we saw that mergeability is equivalent to simple mergeability. Non redundant merging is more flexible but we have seen that the guidance provided decreases by the additional cases retrieved as the number of goals being solved increases in domains where goals are in conflict and subplans have positive interactions. The empirical results show that the guidance provided by the additional cases tends to be small as more cases are merged and more goals are covered. Thus retrieving a single cases covering much of the goals or fewer cases is an effective retrieval strategy.

References

[Barrett and Weld, 1994] Barrett, A. and Weld, D. (1994). Partial-order planning: Evaluating possible efficiency gains. *Artificial Intelligence*, 67(1):71–112.

[Britanik and Marefat, 1995] Britanik, J. and Marefat, M. (1995). Hierarchical plan merging with application to process planning. In *Proceedings of IJCAI-95*, pages 1677–1683.

[Francis and Ram, 1995a] Francis, A. G. J. and Ram, A. (1995a). A domain-independent algorithm for multi-plan adaptation and merging in least-commitment planning. In Aha, D. and Rahm, A., editors, *AAAI Fall Symposium: Adaptation of Knowledge Reuse*, Menlo Park, CA. AAAI Press.

[Francis and Ram, 1995b] Francis, A. J. and Ram, A. (1995b). A comparative utility analysis of case-based reasoning and control-rule learning systems. In *Proceedings ECML-95*, number 912 in Lecture Notes in Artificial Intelligence.

[Gil, 1991] Gil, Y. (1991). A specification of manufacturing processes for planning. Technical Report CMU-CS-91-179, School of Computer Science, Carnegie Mellon University, Pittsburg.

[Hanks and Weld, 1995] Hanks, S. and Weld, D. (1995). A domain-independent algorithm for plan adaptation. *Journal of Artificial Intelligece Research*, 2.

[Hayes, 1987] Hayes, C. (1987). Using goal interactions to guide planning. In *Proceedings of AAAI-87*, pages 224–228.

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

[Ihrig and Kambhampati, 1996] Ihrig, L. and Kambhampati, S. (1996). Design and implementation of a replay framework based on a partial order planner. In Weld, D., editor, *Proceedings of AAAI-96*. IOS Press.

[Kambhampati, 1993] Kambhampati, S. (1993). On the utility of systematicity: Understanding tradeoffs between redundancy and commitment in partial-order planning. In *Proceedings of IJCAI-93*, pages 116–125.

[Kambhampati et al., 1991] Kambhampati, S., Cutkosky, M., Tenenbaum, M., and Lee, S. (1991). Combining specialized reasoners and general purpose planners: A case study. In *Proceedings of AAAI-91*, pages 199–205.

[Kambhampati et al., 1996] Kambhampati, S., Ihrig, L., and Srivastava, B. (1996). A candidate set based analysis of subgoal interactions in conjunctive goal planning. In *Proceedings of the 3rd International Conference on AI Planning Systems (AIPS-96)*, pages 125–133.

[Karinthi et al., 1992] Karinthi, R., Nau, D., and Yang, Q. (1992). Handling feature interactions in process-planning. *Applied Artificial Intelligence*, 6:389–415.

[Koehler, 1994] Koehler, J. (1994). Flexible plan reuse in a formal framework. In *Current Trends in AI Planning*, pages 171–184. IOS Press, Amsterdam, Washington, Tokio.

[Korf, 1987] Korf, R. (1987). Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88.

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

[Muñoz-Avila and Weberskirch, 1996a] Muñoz-Avila, H. and Weberskirch, F. (1996a). Planning for manufacturing workpieces by storing, indexing and replaying planning decisions. In *Proceedings of the 3nd International Conference on AI Planning Systems (AIPS-96)*. AAAI-Press.

[Muñoz-Avila and Weberskirch, 1996b] Muñoz-Avila, H. and Weberskirch, F. (1996b). A specification of the domain of process planning: Properties, problems and solutions. Technical Report LSA-96-10E, Centre for Learning Systems and Applications, University of Kaiserslautern, Germany.

[Nau et al., 1995] Nau, D., Gupta, S., and Regli, W. (1995). AI planning versus manufacturing-operation planning: A case study. In *Proceedings of IJCAI-95*.

[Smith and Peot, 1993] Smith, D. and Peot, M. (1993). Postponing threats in partial-order planning. In *Proceedings of AAAI-93*, pages 500–506.

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

[Veloso and Blythe, 1994] Veloso, M. and Blythe, J. (1994). Linkability: Examining causal link commitments in partial-order planning. In *Proceedings of the 2nd International Conference on AI Planning Systems (AIPS-94)*, pages 13–19.

[Yang et al., 1992] Yang, Q., Nau, D., and Hendler, J. (1992). Merging separately generated plans with restricted interactions. *Computational Intelligence*, 8(2):648–676.