

Plan Merging & Plan Reuse As Satisfiability *

Amol Dattatraya Mali

Dept. of Elect. Engg. & Computer Science, P.O.Box 784
University of Wisconsin, Milwaukee, WI 53201, USA
mali@miller.cs.uwm.edu

July 2, 2003

Abstract

Planning as satisfiability has hitherto focused only on purely generative planning. There is an evidence in traditional refinement planning that planning incrementally by reusing or merging plans can be more efficient than planning from scratch (sometimes reuse is not more efficient, but becomes necessary if the cost of abandoning the reusable plan is too high, when users are charged for the planning solution provided). We adapt the satisfiability paradigm to these scenarios by providing a framework where reusable or mergeable plans can be either contiguous or partially ordered and their actions can be removed and new actions can be added. We report the asymptotic sizes of the propositional encodings for several cases of plan reuse and plan merging. Our empirical evaluation shows that the satisfiability paradigm can scale up to handle plan reuse and plan merging.

*I thank Subbarao Kambhampati and the anonymous referees of the ECP-99 for useful comments on the previous draft of this paper. This paper appears in the proceedings of European Conference on Planning (ECP), Durham, UK, Sept. 1999, pp. 84-96.

1 Introduction

Impressive results have been obtained by casting planning problems as propositional satisfiability in [Kautz & Selman 96]. In planning as satisfiability, a propositional encoding is generated by fixing the number of plan steps (say K). If the number of steps in the solution is more than K , the value of K is increased and an encoding is regenerated. An encoding contains all action sequences of length K and solving it can be viewed as extracting a plan from it. To ensure that each model of an encoding is a valid plan, several constraints about the satisfaction of the pre-conditions of actions and resolution of conflicts between the actions are included in the encoding.

This paradigm has become highly popular, as can be seen from the work that followed [Kautz & Selman 96] such as [Ernst et al 97] [Giunchiglia et al 98] [Kautz et al 96]. However this work is limited to purely generative style of planning where each problem is solved from scratch without reusing or merging plans. On the other hand, there is a considerable evidence and argument in the previous literature that reusing plans can provide improvements in the plan synthesis time [Hanks & Weld 95] [Kambhampati & Hendler 92] [Veloso 94]. Many times, plans have to be reused even if there is no efficiency gain, because the cost of abandoning them is too high (because the users are charged for the planning solution provided). [Britanik & Marefat 95] and [Foulser et al 92] show how plans can be merged by resolving interactions between them and how this kind of plan synthesis can be sometimes faster than planning from scratch.

To examine how these arguments apply to planning as satisfiability, we set up several propositional encodings that contain the constraints from the reusable or mergeable plans. Since there are different forms in which plans can be stored (for further use, to solve a new problem) and there are different ways of encoding them in propositional logic (discussed in [Kautz et al 96]), a variety of ways of casting plan reuse and plan merging as satisfiability exist. These ways are of interest to us because of the different sizes of the encodings that they yield. Our encodings for plan merging and plan reuse are synthesized by adapting the encodings of [Kautz et al 96] to handle these scenarios and we assume some familiarity of the readers with their encodings. We identify the constraints that dominate the asymptotic sizes of the encodings for reuse and merging. Treating the number of actions and fluents as domain constants, we use the power of the variable (number of

steps in an encoding) to compare the encoding sizes.

Our work makes the following contributions.

- An automated synthesis of propositional encodings for several cases of plan merging and plan reuse and a report of their asymptotic sizes.
- We show that the size of the state-based encodings which are shown to be the smallest (have the fewest number of clauses, sum of the clause lengths and number of variables) in the generative planning scenario [Mali & Kambhampati 99], approaches the size of the causal encodings, in the plan merging scenario, when the order preservation restriction (defined in section 3) is enforced.
- We show that the causal encodings for solving planning problems by merging or reusing causal plans are smaller and also faster to solve, than the causal encodings for solving the same problems in a generative style.
- We show that the state-based encodings for merging or reusing contiguous plans are generally neither smaller nor faster to solve, than the state-based encodings that do not reuse or merge plans.
- We show that the causal encodings for merging causal plans can be smaller (and also faster to solve) than the state-based encodings for merging contiguous plans.

In section 2, we explain the notation used for representing the constraints in the plans. In section 3, we describe the semantics of plan merging and show how different cases of merging can be encoded as satisfiability. In section 4, we revisit these cases, but for reusing plans. In section 5, we report empirical results on problems in several benchmark domains¹ and discuss the insights obtained from them. We present conclusions in section 6.

2 Notation

p_i denotes a plan step. o_j denotes a ground action. U is the set of ground pre-condition and effect propositions u_j in the domain. $u_j(t)$ denotes that

¹Available at <http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/mcdermott.html>

u_j is true at time t . ϕ denotes the null action (no-op) which has no pre-conditions or effects. O is the set of ground actions in the domain, it also includes ϕ . ϕ occurs at a time j only when no other non null action occurs at j . That is, ϕ is mutually exclusive with every other non null action. $(p_i = o_j)$ denotes the step→action mapping and o_j is called the binding of p_i . p_i then inherits the pre-conditions and effects of o_j . $p_i \xrightarrow{f} p_j$ denotes a causal link where p_i (contributor) adds the condition f and p_j (consumer) needs it and p_i precedes p_j . $o_i(j)$ denotes that the action o_i occurs at time j (when this happens, the pre-conditions of o_i are true at time j and its effects are true at time $(j + 1)$). $p_i \prec p_j$ denotes that p_i precedes p_j . k_i denotes the number of steps in i th plan. m denotes the number of plans being merged. o_{ij} denotes j th action ($j \in [1, k_i]$) from i th plan, $i \in [1, m]$. K denotes the sum of the number of steps from the m plans being merged or the number of steps in a reusable plan. If there are K time steps in a state-based encoding, they range from 0 to $(K - 1)$. K' denotes the number of new steps added during plan reuse. A causal link $p_i \xrightarrow{f} p_j$ is said to be threatened if there is a step p_q that deletes f , such that $p_i \prec p_q, p_q \prec p_j$.

In sections 3 and 4, we do not list all the constraints that the encodings contain (since many are similar to the constraints in the encodings of [Kautz et al 96]), but we rather focus on the constraints that either capture the representation of the mergeable and reusable plans or dominate the asymptotic size of an encoding. When plans are to be reused or merged, it is necessary to convert the constraints of these plans into clauses that can be included in the encodings. We call this process “the translation/ adaptation of constraints in the plans”. We assume that the plans that are reused or merged may be stored as either a contiguous sequence of actions or as a causal structure (a set of causal links and precedence constraints). It should be noted that contiguous plans are not necessarily found by state space planners and causal plans are not necessarily found by partial order planners. We refer to an encoding for the non-incremental planning scenario (where plans are generated from scratch, without any merging or reuse) as a “naive” encoding.

3 Plan Merging

Plan merging consists of uniting separately generated plans into one global plan, obeying the constraints due to interactions within and between indi-

vidual plans [Britanik & Marefat 95]. Our notion of merging is close to the one in [Kambhampati et al 96]. As per their definition of *mergeability*, a plan P' found by merging the plans P_1 and P_2 contains all the constraints from P_1 and P_2 . We allow the actions from the mergeable plans to be removed while the merging process takes places, however, if the actions are not removed, then the orderings between them (if the plans are contiguous) and both the orderings and causal links between them (if the plans are causal) are preserved in the final plan (which is an outcome of the merging process) (we refer to this criterion as the *order preservation restriction*).

We consider four cases of plan merging obtained by varying the following two factors - **1.** The plans that are merged may be either contiguous or partially ordered (partial order plans are also referred to as *causal plans*). **2.** The encoding itself may be based on state space planning or partial order planning.

We assume that the specification of the merging problem contains initial state and goal state of the problem to be solved and the plans to be merged. For all four cases of merging, we make the following assumptions -

1. No new actions are added, that is, an action o_j will not belong to the final plan if it does not appear in any of the m plans that are merged. **2.** A contiguous plan to be merged is a contiguous action sequence $[o_{i_1} o_{i_2} \dots]$. **3.** i th causal plan to be merged has k_i step \rightarrow action bindings, c_i causal links and r_i precedence relations on the steps. **4.** Actions from the mergeable plans can be removed just by mapping the corresponding steps to the null action.

3.1 Merging Contiguous Plans Using the State-based Encoding

[Kautz et al 96] use the term state-based encoding to refer to an encoding based on state space planning, with the action variables eliminated, since the occurrence of an action is equivalent to the truth of its pre-conditions and effects. We do not insist on the elimination of action variables. Since the steps in this encoding are contiguous and actions from the plans may need to be interleaved, we make K copies of each plan. We concatenate the individual plans in a particular order to create a sequence of K steps and then we concatenate this sequence with itself $(K - 1)$ times to create a

sequence of K^2 steps. Then we convert this sequence into the structure in Figure 1 where each action o_i in the sequence is replaced by the disjunction $(o_i \vee \phi)$ to allow the removal of o_i while synthesizing the final plan. For example, if we want to merge the two contiguous plans $[o_1o_2]$ and $[o_3o_4]$, then the step \rightarrow action mapping structure shown in Figure 1 is used. It can be verified that this structure represents all possible orderings of the actions in these two plans. As we show later, making K copies of each plan allows us to preserve the orders of the actions in the original plans, in the final plan.

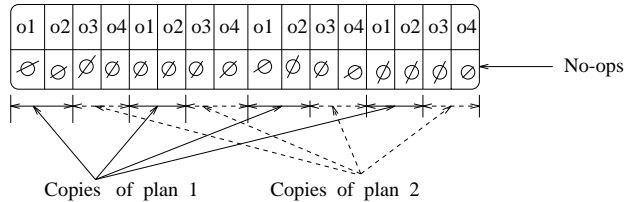


Figure 1: Merging two contiguous plans by making 4 copies of each. $K = 4$. Thus 16 steps are used.

Consider the state-based encoding with explanatory frame axioms from [Kautz et al 96] to handle the merging process. Since the truth of each of the $|U|$ propositions is stated at each time step (to compute the world state), there are $O(K^2 * |U|)$ variables in the encoding. Since explanatory frame axioms explaining the change in the truth of each of the propositions from U are stated for each of the K^2 time steps, there are $O(K^2 * |U|)$ clauses in the encoding. (An explanatory frame axiom states that if a proposition f is true at time t and false at time $(t + 1)$, some action having the effect of deleting f must have occurred at t). Note that if a K step encoding is generated to solve the planning problem from scratch, without merging the plans, it will have $O(K * (|O| + |U|))$ variables and $O(K * (|U| + |O|))$ clauses (because one has to consider the possibility of any action from the domain occurring at any time step (which requires $K * |O|$ variables) and because an action occurring at time t implies the truth of its pre-conditions at t and the truth of its effects at $(t + 1)$ which requires $O(K * |O|)$ clauses and $O(K * |U|)$ clauses are required to state the explanatory frame axioms).

Though the encoding has K^2 steps, we are looking for a plan of only K steps. To reduce the potential non-minimality in the plans, we specify extra constraints. If a particular occurrence of an action from a plan is used (that

is, it is not removed by choosing ϕ), then all other $(K - 1)$ copies of this occurrence of this action must be removed and those steps must be mapped to ϕ , to reduce non-minimality in the plans (e.g. repeatedly loading and unloading the same package at the same location). This is stated in the following clauses.

$$\bigwedge_{q=1}^K \bigwedge_{i=1}^m \bigwedge_{j=1}^{k_i} (o_{ij}((q-1)*K + \sum_{p=1}^{i-1} k_p + j - 1) \Rightarrow (\bigwedge_{q_1=1, q_1 \neq q}^K \neg o_{ij}((q_1 - 1)*K + \sum_{p=1}^{i-1} k_p + j - 1)))$$

This specification requires $O(K^3)$ clauses. We also need clauses to respect the order preservation restriction. Since the i th copy of any plan precedes its j th copy (when $i < j$), if actions o_a and o_b are present in the plan such that o_a precedes o_b , then if o_b from i th copy of the plan appears in the final plan, the copies of o_a in all the remaining j th copies of the plan, $j \in [i+1, K]$ must be discarded and ϕ should occur at those time steps. If this does not happen, the ordering relations between the actions from the original plan will be violated. The following clauses enforce the order preservation restriction. The number of such clauses required will be $O(K^3 * \alpha)$, $\alpha = \max(\{k_i \mid i \in [1, m]\})$.

$$\bigwedge_{q=1}^K \bigwedge_{i=1}^m \bigwedge_{j=1}^{k_i} (o_{ij}((q-1)*K + \sum_{p=1}^{i-1} k_p + j - 1) \Rightarrow (\bigwedge_{q_1=q+1}^K \bigwedge_{j_1=1}^{j-1} \neg o_{ij_1}((q_1 - 1)*K + \sum_{p=1}^{i-1} k_p + j_1 - 1)))$$

Thus the number of variables and clauses in the encoding are $O(K^2 * |U|)$ and $O(K^2 * |U| + K^3 * \alpha)$ respectively.

3.2 Causal Plans Merged Using the Causal Encoding

We explain below how each constraint in a causal plan is represented in the encoding, to handle the merging scenario. The step \rightarrow action binding $(p_i = o_j)$ is modified to $((p_i = o_j) \vee (p_i = \phi))$. We need the mutual exclusion constraint $\neg((p_i = o_j) \wedge (p_i = \phi))$ as well. Let us consider the causal links. A causal link $p_i \xrightarrow{f} p_j$ is modified to the constraints $((p_i \xrightarrow{f} p_j) \Rightarrow ((p_i = o_s) \wedge (p_j = o_q) \wedge (p_i \prec p_j)))$ and $((p_i = o_s) \wedge (p_j = o_q) \Rightarrow (p_i \xrightarrow{f} p_j))$ (for order preservation restriction) where o_s, o_q are the actions to which p_i, p_j are mapped in the plan being merged. This means that if the causal link is used, the step \rightarrow action binding of the contributing and consuming steps must be used as well. Also, if the contributor and consumer steps are not mapped to ϕ , then the causal links must be used as well. Some of the r_i precedence relations in i th plan might have been introduced to resolve the threats.

In such a case we assume that the plan contains a record indicating that the ordering was introduced to resolve the threat. Such an ordering is also represented. For example, let the ordering $p_1 \prec p_2$, $p_1 = o_1$ be introduced to resolve the threat to the link $p_2 \xrightarrow{f} p_3$ posed by p_1 . Then $p_1 \prec p_2$ is translated into the constraint $((p_2 \xrightarrow{f} p_3) \wedge (p_1 = o_1)) \Rightarrow (p_1 \prec p_2)$. Note that since new causal links may be established between steps from different plans and new threats can arise, $O(K^3 * |U|)$ (all possible causal links) variables and $O(K^3 * |U|)$ threat resolution clauses are needed (to resolve the potential threats to each of the $O(K^2 * |U|)$ potential causal links by each of the $O(K)$ steps). These also determine the size of the encoding. Note that the actual encoding will be much smaller since the knowledge of the step \rightarrow action binding can be propagated to generate fewer causal links and fewer threat resolution axioms. In this merging-based encoding, the size of the domain of the binding of each step is 2 (because it is either retained or removed and not mapped to a new action) rather than $|O|$, which is the case for the naive causal encoding. Thus the causal encoding for merging causal plans will be significantly smaller than the naive causal encoding. This is also confirmed by the sizes in our experiments reported in Fig. 6.

Though we do not discuss the cases of merging causal plans using the state-based encoding and merging contiguous plans using the causal encoding, we report their asymptotic sizes in Figure 3.

4 Plan Reuse

We assume that the specification of the reuse problem contains the plan to be reused and the initial and goal state of the problem to be solved, along with the ground actions in the domain. We focus on generating $(K + K')$ step encodings in which the reusable plan is represented, along with the K' new steps. (A related issue is that of ensuring that some portion (even the maximum applicable) of the reusable plan is reused. In section 4.3, we discuss some strategies to ensure that the reusable plan is maximally recycled.) Similar to plan merging, we consider 4 cases of plan reuse. Our assumptions for reuse are stated below -

1. Actions in the reusable plan are either used in the final plan or removed by replacing them with ϕ . **2.** If the reusable plan is causal, it has c causal links and r precedence constraints. **3.** A new step may be mapped to any of

the $|O|$ actions in the domain.

If a contiguous plan is to be reused, we represent only the actions from the plan in an encoding. We do not represent the orderings of the actions from the plan because the actions in a contiguous plan are not ordered with the least commitment, that is, they may be ordered even if there is no reason to do so. On the other hand, if a causal plan is to be reused, we represent the actions, orderings and causal links from it into an encoding.

4.1 Reusing a Causal Plan Using the Causal Encoding

We discuss here how the constraints in a reusable causal plan are represented in an encoding. Each step \rightarrow action binding $(p_i = o_j)$ in the plan is replaced by the constraint

$$((p_i = o_j) \vee (p_i = \phi)) \wedge \neg((p_i = o_j) \wedge (p_i = \phi))$$

The size of the domain of binding of each step p_i from the reusable plan is 2 ($\{o_j, \phi\}$ here) rather than $|O|$ and this reduction can be propagated to reduce the size of the encoding, as confirmed by the results in Fig. 5. A causal link $p_i \xrightarrow{f} p_j$ is represented by the constraint $((p_i \xrightarrow{f} p_j) \Rightarrow ((p_i = o_s) \wedge (p_j = o_q) \wedge (p_i \prec p_j)))$ where o_s, o_q are the actions to which p_i, p_j are mapped (in the reusable plan). If the ordering $p_1 \prec p_2$, $p_1 = o_1$ was introduced to resolve the threat to the link $p_2 \xrightarrow{f} p_3$ posed by p_1 , then $p_1 \prec p_2$ is translated into the constraint $((p_2 \xrightarrow{f} p_3) \wedge (p_1 = o_1)) \Rightarrow (p_1 \prec p_2)$. Assuming that most of the causal links between the K steps from the reusable plan will be present in it, the number of new variables in the encoding will be dominated by the new causal links that will be established between new K' steps and the new K' steps and the old K steps. Hence the number of variables in the encoding will be $O((K'^2 + K' * K) * |U| + c)$. Assuming that the reusable plan is free of threats (since such threat resolving information is present in the reusable plan itself and is translated into appropriate constraints that are added to the encoding), since the steps from the reusable plan may threaten new causal links and the new steps may threaten both the new causal links as well as the causal links from the reusable plan, $O((K'^3 + K'^2 * K) * |U| + K' * c)$ clauses will be required to resolve the threats.

4.2 Reusing a Contiguous Plan Using the State-based Encoding

We discuss two schemes here to achieve this.

4.2.1 Scheme 1

Since the actions from the reusable plan may have to be removed or reordered to allow the incorporation of new actions at arbitrary places, we make K copies of the reusable plan and reserve $(K + 1)$ blocks, each of K' steps for accomodating new actions (as shown in Figure 2). This means that to synthesize a $(K + K')$ step plan by reusing a K step plan, we have $(K^2 + (K + 1) * K')$ steps in the encoding. Due to the representation of state at each time step, the possibility of occurrence of any of the $| O |$ actions at the $(K + 1) * K'$ steps reserved for new actions and the explanatory frame axioms, the encoding has $O((K^2 + (K + 1) * K') * | U | + (K + 1) * K' * | O |)$ variables and clauses each. To ensure that the extra $(K^2 + K * (K' - 1))$ steps are mapped to no-ops, $O(K^3 + (K + 1)^2 * K')$ additional clauses are needed.

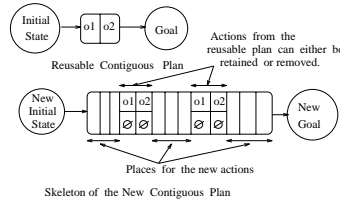


Figure 2: Scheme 1 - Reusing a contiguous plan by making multiple copies of it and reserving multiple blocks of steps for the new actions, $K = 2, K' = 3$.

4.2.2 Scheme 2

In this scheme, we specify that each action from the reusable plan will occur at some time step or will never occur. As in the naive state-based encoding, this encoding will have $O((K + K') * (| O | + | U |))$ variables and $O((K + K') * (| U | + | O |))$ clauses. Note that if an action o_i occurs in the reusable

plan s_i times, then since K' new steps are added, this action should certainly not occur more than $(K' + s_i)$ times in the final plan. The specification

$$((\bigvee_{j=0}^{K+K'-1} o_i(j)) \vee \phi'_i)$$

does not guarantee this (ϕ'_i is introduced to be able to make the clause true if o_i is not required). In this scheme, we can restrict the maximum number of occurrences of o_i in the final plan to $(K' + s_i)$, but only by adding $O((K + K')^{(K'+s_i+1)})$ new clauses which state that no choice of $(K' + s_i + 1)$ out of $(K + K')$ steps should have o_i occurring at them. Thus though this scheme yields an encoding that has fewer variables and fewer clauses than the encoding in scheme 1, it requires more clauses to guarantee a control over the number of occurrences of an action.

4.3 Ensuring the Reuse of a Plan

We discuss here various ways of constraining or solving the encodings from sections 4.1 and 4.2 for ensuring that a plan will be maximally reused.

1. Weighted Satisfiability - One can assign weights to the clauses that represent the constraints from the reusable plan and maximize the sum of the weights of the satisfied clauses. MAX-SAT solvers can be used to find models that contain as much part of the reusable plan that can be recycled as possible. **2. Iterative Solving** - One can create a K step encoding such that it contains only the constraints from the reusable plan and try to solve it. If a solution can be found by removing actions from the reusable plan or if the reusable plan also solves the new problem, one can terminate. If there is no model of this encoding, one can increase the number of steps in the encoding by 1 and iteratively solve the resulting encodings. This ensures that the plan is maximally reused.

[Nebel & Koehler 95] prove that deriving the maximally reusable sub-plan is not easier than planning. Also, as they point out, maximal reuse makes sense only in the replanning scenario, where costs are charged for not executing the steps already planned. The conventional reuse strategies like those in [Kambhampati & Hendler 92] are heuristic and do not guarantee maximal recycling of the reusable plan. In our empirical evaluation, the plans were maximally recycled because the values of K' chosen were same as the minimum number of new actions required to be included in the reusable plans.

<i>Plan Type</i>	<i>Type of Encoding</i>	
	<i>State-based</i>	<i>Causal</i>
<i>Contiguous</i>	$O(K^2 * U)$ vars	$O(K^2 * U)$ vars
	$O(K^2 * U + K^3 * \alpha)$ clauses	$O(K^3 * U)$ clauses
<i>Causal</i>	$O(C * K^2)$ vars	$O(K^2 * U)$ vars
	$O(C * K^3)$ clauses	$O(K^3 * U)$ clauses

Figure 3: Sizes of the encodings for plan merging. C denotes the sum of the number of causal links in the individual plans.

5 Discussion

A comparison of the asymptotic sizes of various encodings for the plan merging and plan reuse cases is shown in Figures 3, 4. Note that the main attraction of the state-based encodings, in the non-incremental planning is their lowest size [Ernst et al 97] and the causal encodings are not used because they are hard to solve [Mali & Kambhampati 99]. We have shown that in the plan merging scenario, in the presence of the order preservation restriction, the size of the state-based encoding approaches the size of the causal encoding.

Our empirical results are shown in Figures 5 and 6². The number of steps in the encodings were same as the number of actions in the plans. The encodings were generated and solved on a Sun Ultra with 128 M RAM. As argued in the sections 3 and 4, the causal encodings for the reuse or merging of causal plans are far smaller than the naive causal encodings, since the domains of bindings of the steps from the reusable or mergeable plans are much smaller than $|O|$ and this shrinkage can be propagated to trim the portions of the encodings dependent on these steps. The causal encodings for causal plan merging and causal plan reuse were far smaller and also faster to

²The encodings were solved with the “satz” solver at SATLIB

<i>Plan Type</i>	<i>Type of Encoding</i>	
	<i>State-based</i>	<i>Causal</i>
<i>Contiguous</i>	$O((K + K') * (O + U))$ vars	$O((K + K')^2 * U)$ vars
	$O((K + K') * (U + O))$ clauses	$O((K + K')^3 * U)$ clauses
<i>Causal</i>	$O(c * (K + K')^2)$ vars	$O((K'^2 + K * K') * U + c)$ vars
	$O(c * (K + K')^3)$ clauses	$O((K'^3 + K'^2 * K) * U + K' * c)$ clauses

Figure 4: The sizes of the encodings for plan reuse. The size of the encoding for the reuse of a contiguous plan using the state-based encoding assumes that scheme 2 in section 4.2 is used. c denotes the number of causal links in the reusable plan.

solve than the naive causal encodings. Though the state-based encodings are the smallest in the generative planning, the state-based encodings for merging contiguous plans were sometimes harder to solve than the causal encodings for merging the causal plans (as shown in Figure 6), due to the blowup in their size (because of multiple copies of the plans) needed to respect the order preservation restriction and because they could not be significantly simplified like the causal encodings, by propagating the knowledge of the step \rightarrow action mapping from the mergeable plans. We also found that neither the causal encodings for merging contiguous plans nor the state-based encodings for merging causal plans were easier to solve, possibly due to the introduction of intermediate variables, the multiple copies of the plans and the choice of encodings that are based on different type of planning than the representation of the mergeable plans, which made the explicit variable dependency in the plans less visible to the systematic solver.

We found that plan reuse as satisfiability is not necessarily faster than generative planning as satisfiability, especially when a contiguous plan is reused using the state-based encoding. This is however not a surprise, since the sizes of these encodings are almost same. We also found that the causal encodings for reusing causal plans are harder to solve than the state-based encodings for reusing contiguous plans, because despite the reduction in the sizes of the causal encodings achieved by constraint propagation, $O((K +$

$K')^2$) variables like $p_i \prec p_j$ are still needed to represent the all possible \prec relations between the steps.

Our work ignores the costs of goal decomposition and generation of sub-plans that fulfill subgoals (in plan merging) and the plan retrieval cost in plan reuse. In future, we intend to augment the declarative approach of satisfiability with procedural control, to efficiently handle the problems of goal decomposition, subplan generation and plan retrieval.

6 Conclusion

We developed a framework for casting the plan reuse and plan merging problems as satisfiability. We reported the complexities of converting the reuse and merging problems into encodings in propositional logic. This analysis and the empirical evaluation lead to several new insights. We showed that the causal encoding of [Kautz et al 96], which has been shown to be hard to solve in generative planning [Mali & Kambhampati 99], can be used to solve several problems (from benchmark domains) of plan merging and plan reuse, due to the reduction in its size achieved by the propagation of the constraints from the plans, though it was still not always the fastest to solve. We also showed that the size of the state-based encoding with explanatory frame axioms [Kautz et al 96] which has been empirically shown to be the smallest and also generally the fastest to solve in the generative planning scenario [Ernst et al 97][Giunchiglia et al 98][Mali & Kambhampati 99], approaches the size of the naive causal encoding, due to its adaptation required to handle the order preservation restriction of plan merging. We also showed that though the satisfiability paradigm is scalable to reusing and merging plans, plan reuse is not always faster than generative planning (however as pointed out in [Nebel & Koehler 95], plan reuse may still be indispensable, if the users are charged for the planning solution provided).

References

- [**Britanik & Marefat 95**] J. Britanik and M. Marefat, Hierarchical plan merging with application to process planning, Procs. of the International Joint Conference on Artificial Intelligence (IJCAI), Vol. 2, 1995, 1677-1683.
[**Ernst et al 97**] Michael Ernst, Todd Millstein and Daniel Weld, Automatic

Domain, Plan Actions	CPSB	S	CPC	CU
	V, C, T	V, C, T	V, C, T	V, C, T
Tsp, 14 $K = 7, K' = 7$ $k = 14$	638	631	1562	7785
	1744	1639	10003	88873
	0.06	0.07	3.59	2.48
Tsp, 29 $K = 20, K' = 9$ $k = 29$	2631	2611	5098	58320
	7474	6874	55766	1509973
	0.37	0.3	1.42	*
Ferry, 79 $K = 52, K' = 27$ $k = 79$	8410	8358	33164	*
	65231	61071	1198060	*
	1.3	1.27	*	*
Logistics, 35 $K = 19, K' = 16$ $k = 35$	3240	3132	9118	73637
	10060	9063	138402	2343912
	0.94	0.61	-	*
Blocks, 12 $K = 9, K' = 3$ $k = 12$	543	534	675	6123
	1788	1563	3156	59715
	0.12	0.11	0.34	1.6

Figure 5: **Empirical results for plan reuse and generative planning.** The state-based encoding in case **CPSB** was generated using scheme 2 in section 4.2. K, K' denote the number of steps in the reusable plan and those newly added to it respectively. In this figure as well as figure 6, V, C, T denote the number of variables and clauses in and the times needed to solve the encodings respectively. Times are in CPU seconds. k denotes the number of steps in the naive encodings. **CPSB** denotes the contiguous plan, state-based encoding case, **S** denote the naive state-based encoding, **CPC** denotes the causal plans, causal encoding case and **CU** denotes the naive causal encoding. A “-” indicates that the encoding was not solved within 10 minutes of CPU time. A “*” denotes that the encoding was too large to store. Tsp is the traveling salesperson domain.

Domain, Plan Actions	CPSB	S	CPC	CU
	V, C, T	V, C, T	V, C, T	V, C, T
Tsp, 14 $K = k = 14$ $m = 5$	5909	631	379	7785
	16017	1639	2616	88873
	0.29	0.07	0.43	2.48
Tsp, 29 $K = k = 29$ $m = 5$	50519	2611	1219	58320
	161560	6874	23022	1509973
	-	0.3	10.05	*
Ferry, 19 $K = k = 19$ $m = 4$	6877	588	682	8535
	27744	2436	7231	138172
	0.5	4.05	57.75	-
Logistics, 35 $K = k = 35$ $m = 5$	68740	3132	1726	73637
	256370	9063	44326	2343912
	-	0.61	22.69	*
Blocks, 12 $K = k = 12$ $m = 4$	4494	534	324	6123
	11775	1563	1737	59715
	7.81	0.11	0.22	1.6

Figure 6: Empirical results for plan merging and generative planning. m denotes the number of plans merged and K denotes the sum of the number of steps in plans merged.

- SAT compilation of planning problems, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.
- [**Foulser et al 92**] David E. Foulser, Ming Li and Qiang Yang, Theory and algorithms for plan merging, *Artificial Intelligence* 57, 1992, 143-181.
- [**Giunchiglia et al 98**] Enrico Giunchiglia, Alessandro Massarotto and Roberto Sebastiani, Act and the rest will follow: Exploiting determinism in planning as satisfiability, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1998.
- [**Hanks & Weld 95**] Steve Hanks and Daniel S. Weld, A domain-independent algorithm for plan adaptation, *Journal of Artificial Intelligence Research* 2, 1995, 319-360.
- [**Kambhampati & Hendler 92**] Subbarao Kambhampati and James A. Hendler, A validation-structure-based theory of plan modification and reuse, *Artificial Intelligence* 55, 1992, 193-258.
- [**Kambhampati et al 96**] S. Kambhampati, L. Ihrig and B. Srivastava, A candidate set-based analysis of subgoal interactions in conjunctive goal planning, Proceedings of the International conference on Artificial Intelligence Planning Systems (AIPS), 1996.
- [**Kautz et al 96**] Henry Kautz, David McAllester and Bart Selman, Encoding plans in propositional logic, Proc. of Knowledge Representation and Reasoning Conference, (KRR), 1996.
- [**Kautz & Selman 96**] Henry Kautz and Bart Selman, Pushing the envelope: Planning, Propositional logic and Stochastic search, Proc. of the National Conference on Artificial Intelligence (AAAI), 1996.
- [**Mali & Kambhampati 99**] Amol D. Mali and Subbarao Kambhampati, On the utility of causal encodings, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1999.
- [**Nebel & Koehler 95**] Bernhard Nebel and Jana Koehler, Plan reuse versus plan generation: a theoretical and empirical analysis, *Artificial Intelligence* 76, 1995, 427-454.
- [**Veloso 94**] Manuela M. Veloso, Planning and learning by analogical reasoning, *Lecture notes in artificial intelligence* 886, Springer-Verlag, 1994.