SGPlan: Subgoal Partitioning and Resolution in Planning[*]

Yixin Chen, Chih-Wei Hsu, and Benjamin W. Wah
Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois, Urbana-Champaign
Urbana, IL 61801, USA
{*chen,chsu,wah*}*@manip.crhc.uiuc.edu*

Abstract

We have developed SGPlan, a planner that competes in the Fourth International Planning Competition. SGPlan partitions a large planning problem into subproblems, each with its own subgoal, and resolves inconsistent solutions of subgoals using our extended saddle-point condition. Subgoal partitioning is effective because each partitioned subproblem involves a substantially smaller search space than that of the original problem. We have developed methods for the detection of reasonable orders among subgoals, an intermediate goal-agenda analysis to hierarchically decompose each subproblem, a search-space-reduction algorithm to eliminate irrelevant actions in subproblems, and a strategy to call the best planner to solve each bottom-level subproblem. Currently, SGPlan supports PDDL2.1 and derived predicates, and algorithms for supporting time initiated facts and ADL are under development.

OVERALL ARCHITECTURE

By formulating a subproblem in such a way that each has one goal state, SGPlan partitions a planning problem into subproblems, orders the subproblems according to a sequential resolution of its subgoals, and finds a feasible plan for each goal fact. Using the extended saddle-point condition and constrained search, new constraints are enforced to ensure that facts and assignments in a later subgoal are consistent with those of earlier subgoals.

Figure 1 shows the architecture of our planner. In the global level, we select a suitable order for the planner to solve the partitioned subgoals, introduce artificial global constraints to enforce that the solution of one subgoal solved later does not invalidate that of an earlier subgoal, and resolve violated global constraints using the theory of extended saddle points. In the local level, we perform a hierarchical decomposition of first-level subgoals, prune irrelevant facts and actions before calling a basic planner, and choose a suitable basic planner for solving the second-level subproblem.

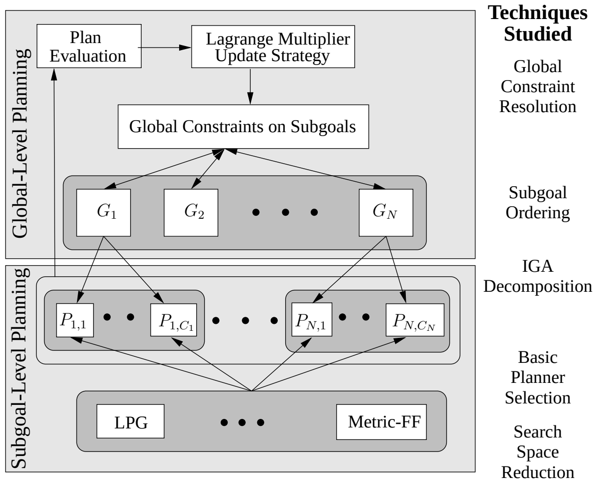

Figure 1: The architecture of SGPlan.

Figure 2 presents the pseudo code of our planner. Based on the subgoals identified, we partition the problem into N subproblems $G_1, \cdots, G_N$, one for each subgoal, and order the subproblems appropriately. For G_i, we perform an intermediate-goal-agenda (IGA) analysis to decompose it into C_i smaller subproblems $P_{i,1}, \cdots, P_{i,C_i}$. For each second-level subproblem, we perform subspace-reduction analysis to reduce its search space and choose a suitable planner (called *basic planner*) to solve it. Finally, we evaluate the composed plan and update the Lagrange multipliers.

Our approach is different from incremental planning (Koehler & Hoffmann 2000) that uses a goal agenda. In incremental planning, a planner maintains a set of target facts, adds goal states incrementally into the target set, and extends the solution by using the new target set. This means that a goal state will always be satisfied once it is satisfied. However, it may be more expensive to solve subsequent problems, since the search space increases as more goal states are added. Moreover, it is difficult to tell which goals should be satisfied before others. In contrast, SGPlan always involves only

[*]Research supported by the National Aeronautics and Space Administration Grant NCC 2-1230 the National Science Foundation Grant ITR 03-12084.

```
1.  procedure SGPlan
2.      compute the partial orders among subgoals;
3.      generate an initial ordered list of subgoals;
4.      set $iter \longleftarrow 0$;
5.      repeat
6.          for each goal fact in the subgoal list
7.              find the intermediate goal facts;
8.              generate an IGA agenda;
9.              for each entry in the IGA agenda
10.                 call search space reduction procedure and
                        eliminate irrelevant actions;
11.                 call basic planner to solve the subproblem;
12.             end_for
13.         end_for
14.         if (plan $z$ found is feasible)
15.             evaluate the solution plan;
16.             decrease some Lagrange multipliers;
17.         else increase Lagrange multipliers $\gamma$ on unsatisfied
                global constraints;
18.         $iter \leftarrow iter + 1$;
19.         if ($iter$ % $\tau$ == 0) dynamically re-order the subgoals;
20.     until no change on $z$ and $\gamma$ in an iteration;
21. end_procedure
```

Figure 2: The pseudo code of SGPlan.

one goal fact in a subproblem. Therefore, the search space of the subproblems is not increasing, and irrelevant actions in each subproblem can be pruned.

GLOBAL-LEVEL PLANNING

Subgoal Ordering and Global Constraints

When dependent subgoals are evaluated sequentially, it is possible that a subgoal evaluated later may invalidate the results of a subgoal evaluated earlier, and the earlier subgoal has to be re-evaluated. Although such conflicts may be unavoidable, appropriately ordered subgoals can significantly reduce the occurrences of such conflicts. Intuitively, difficult subgoals should be resolved before easier ones.

It is non-trivial to find an optimal order that minimizes the conflicts among subgoals. In fact, it may be more computationally expensive to find the best order than solving the problem itself. In SGPlan, we have developed three heuristics for partial ordering of subgoals that can be computed efficiently (Step 2 of SGPlan).

The first level is called *reasonable ordering* proposed in (Koehler & Hoffmann 2000). Suppose goal fact A is ordered before B in the subgoal list, but after we get a plan that achieves A, we cannot achieve B without invalidating A first. Then the search for achieving A first is wasted, and it is more efficient to achieve B before A. We use an algorithm in FF2.2 (Koehler & Hoffmann 2000) to find such reasonable orders.

For goal pairs not ordered by reasonable ordering, we apply a second level of ordering called *irrelevance ordering*. Based on backward relevance analysis (discussed in the next section), we compute the number of irrelevant actions of each goal fact, and order A before B if A has less irrelevant actions. The idea is to resolve more difficult subgoals, with less irrelevant actions.

For goal pairs not ordered by the first two levels, we apply the third level of ordering called *precondition ordering*. Specifically, for A and B with the same number of irrelevant actions that cannot be ordered by reasonable ordering, we order A before B if $n_p(A) > n_p(B)$. Here, $n_p(A)$ is the minimum number of preconditions of those supporting actions:

$$n_p(A) = \min_{a \in S(A)} n_{pre}(a), \tag{1}$$

where $S(A)$ is the set of all actions that support goal fact A, and n_{pre} is the number of preconditions of action a. Again, the idea is that more difficult goals, with larger n_p, should be resolved first.

For pairs of subgoals that are not involved in any of the three levels or ordering, we randomly order them. At the beginning of a search, we randomly generate a total ordering of the goal facts that satisfy the three levels of partial orders (Step 3) and periodically generate new total orders during the search (Step 19).

To identify conflicts among solutions of subgoals, we define a global constraint so that the solution plan of a subgoal will not invalidate the goal fact of another subgoal. Each global constraint in SGPlan is a binary constraint that indicates whether conflicts exist or not.

Resolution of Global Constraints

The planning problems studied in SGPlan are defined in mixed space with nonlinear objective and constraints that may be procedural and not in closed form. SGPlan implements a search to find extended saddle points in the Lagrangian space of a problem (Chen & Wah 2003; Wah & Chen 2003). The extended saddle-point condition (ESPC) states that solution points in mixed space that are local optima of the objective and that satisfy all the constraints must satisfy ESPC. The condition is defined on a Lagrangian function that consists of the sum of the objective and the constraints weighted by Lagrange multipliers, where an extended saddle point is a point that is a local minimum of Lagrangian function with respect to the original variable space and a local maximum of the function with respect to the Lagrange-multiplier space.

An important property of ESPC is that the condition is true for all Lagrange multipliers larger than a minimum threshold. Hence, finding points that satisfy ESPC can be implemented iteratively, with an inner loop that looks for local minimum of the Lagrangian function, and an outer loop that looks for any Lagrange multipliers larger than the critical threshold. The property also allows a search looking for extended saddle points to be partitioned into multiple searches, each looking for a local extended saddle point for a partitioned problem (Steps 6-12 of Figure 2), and an outer loop that resolves the global constraints across the subproblems (Step 17).

A direct implementation of ESPC in a search algorithm may get stuck in an infeasible region when the objective is too small or when the Lagrange multipliers and/or constraint violations are too large. To address this issue, SGPlan performs periodic decreases of Lagrange multipliers in the Lagrangian space in the outer loop, in addition to ascents (Step 16).

SUBGOAL-LEVEL PLANNING

Subgoal-Level Decomposition

Sometimes the subproblems after first-level partitioning by subgoals are still too large to be solved quickly. An obvious approach to reduce this complexity is to further partition the subproblem into smaller ones.

Given subgoal G after first-level partitioning, we propose to identify some "hidden" intermediate second-level subgoals (or facts) that must be true in any plan that achieves G from a given initial state (Steps 7 and 8). These facts allow us to construct an intermediate goal agenda (IGA), which is an ordered list of agenda entries, each containing a set of intermediate facts.

From a fixed initial state $\mathcal{S}$, we define the following relationship between two facts A and B. A is an intermediate goal before B, denoted as $A \preceq_{IGA} B$, if the planning graph starting from S cannot achieve B without achieving A first. We construct the planning graph similar to that in Graphplan, with the following two changes: a) we do not compute any mutual exclusion relations; b) we forbid the insertion of A into the planning graph at any level (thereby also forbidding the insertion of any actions having A as a precondition). If B is not in the planning graph after the construction of the graph, then we have $A \preceq_{IGA} B$.

Based on the intermediate facts, we detect the $\preceq_{IGA}$ orders among them and construct a directed graph showing their partial orders. We then identify an agenda of sets of facts that must be true in any plan of G.

SGPlan determines dynamically whether partitioning should be further carried out, depending on whether a subgoal G is easy enough to be resolved quickly using the IGA agenda. If subgoal G is to be partitioned, SGPlan further uses symmetry-group detection to see if a path can be constructed from the current facts to the subgoal: $f_0 \rightarrow f_1 \rightarrow \cdots \rightarrow G$, where $f_0, f_1, \cdots$ are all in the same symmetry group as that of G. It then partitions the problem of achieving G from f_0 into N subproblems: $f_0 \rightarrow f_1, f_1 \rightarrow f_2, \ldots, f_{N-1} \rightarrow G$.

Our approach is different from existing approaches for finding intermediate facts (Koehler & Hoffmann 2000) that expand a search space from the goal state and find some indispensable pre-conditioning facts. Since the initial state is not specified, there is no way to tell to what depth the backward expansion should stop. In contrast, our method considers both the initial and the goal states in determining whether an intermediate fact is critical and always stops in finite levels of expansions. In addition, we detect the partial orders among these facts and form an agenda to avoid unachievable intermediate states, which could occur in previous methods.

Search-Space Reduction

After partitioning a subproblem into easier second-level subproblems, we can often eliminate many irrelevant actions in their search space before solving them. Such a reduction is generally not applicable to planning problems that are not partitioned because in most cases all actions in their search space are relevant.

We have designed a polynomial-time *backward relevance analysis* to exclude some irrelevant actions before applying any planner to solve a subproblem (Step 10). Given a subproblem to be solved, we maintain an *open list* of unsupported facts, a *close list* of relevant facts, and a *relevance list* of relevant actions. In the beginning, the open list contains only the subgoal facts of the subproblem, and the relevance list is empty. In each iteration, for each fact in the open list, we find all the actions supporting that fact and not already in the relevance list. We then add these actions to the relevance list, and add the action preconditions that are not in the close list to the open list. We move a fact from the open list to the close list when it is processed. The analysis ends when the open list is empty. At that point, the relevance list will contain all possible relevant actions, while excluding those irrelevant actions.

Since partitioned subproblems usually have similar structures, we learn suitable rules for subproblem solving during a search. After a number of trial-and-error, SGPlan records some suitable heuristics and parameters that lead to the successful resolution of subgoals and use them in solving other subproblems.

Basic-Planner Selection

Our current implementation of SGPlan uses a modified Metric-FF planner for basic planning and only invokes LPG when the modified planner fails. We have developed new algorithms and modified heuristic functions in the enhanced Metric-FF to fully support derived predicates, temporal planning, and time initiated facts (still under development).

References

Chen, Y. X., and Wah, B. W. 2003. Automated planning and scheduling using calculus of variations in discrete space. In *Proc. Int'l Conf. on Automated Planning and Scheduling*, 2–11.

Koehler, J., and Hoffmann, J. 2000. On reasonable and forced goal ordering and their use in an agenda-driven planning algorithm. *J. of AI Research* 12:339–386.

Wah, B. W., and Chen, Y. X. 2003. Partitioning of temporal planning problems in mixed space using the theory of extended saddle points. In *Proc. IEEE Int'l Conf. on Tools with Artificial Intelligence*, 266–273.