New Features in SGPlan for Handling Preferences and Constraints in PDDL3.0*

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
{*chsu,wah*}*@manip.crhc.uiuc.edu*

Ruoyun Huang and Yixin Chen
Department of Computer Science and Engineering
Washington University in St Louis
St Louis, MO 63130, USA
rh11@cec.wustl.edu
chen@cse.wustl.edu

Abstract

In this paper, we describe our enhancements incorporated in SGPlan (hereafter called SGPLan_5) for supporting the new features of the PDDL3.0 language used in the Fifth International Planning Competition (IPC5). Based on the architecture of SGPlan that competed in the Fourth IPC (hereafter called SGPLan_4), SGPLan_5 partitions a large planning problem into subproblems, each with its own subgoal, and resolves those inconsistent solutions using our extended saddle-point condition. Subgoal partitioning is effective for solving large planning problems because each partitioned subproblem involves a substantially smaller search space than that of the original problem. In SGPLan_5, we generalize subgoal partitioning so that the goal state of a subproblem is no longer one goal fact as in SGPLan_4, but can be any fact with loosely coupled constraints with other subproblems. We have further developed methods for representing a planning problem in a multi-valued form in order to accommodate the new features in PDDL3.0, and for carrying out partitioning in the transformed space. The multi-valued representation leads to more effective heuristics for resolving goal preferences and trajectory and temporal constraints.

DESIGN GOALS

SGPLan_5 has participated in the suboptimal track of the deterministic part of the Fifth International Planning Competition (IPC5). It was designed to fully support the PDDL3.0 language (Gerevini & Long 2005) specifications.

PDDL3.0, the modeling language used in IPC5, extends the previous PDDL2.2 (Edelkamp & Hoffmann 2004) specifications by introducing several new features: a) simple preferences over only action preconditions or goals, b) qualitative preferences that are logical preferences over trajectory constraints, c) complex constraints that are trajectory constraints with metric time and possibly numeric fluents, and d) complex preferences that are preferences over trajectory constraints with metric time and possibly numeric fluents.

SGPLan_5 uses a *multi-valued domain formulation (MDF)* based on the SAS+ formalism. MDF has been used by other planning systems, such as Fast Downward (Helmert 2004) and IP planner (van den Briel, Vossen, & Kambhampati 2005). We have implemented our own preprocessing engine for translating a PDDL3.0 problem into MDF. There are several reasons for using MDF.

a) MDF provides a more compact representation than a binary-valued representation and leads to a more effective partitioning of the constraints.
b) Using transition graphs, MDF facilitates the analysis of the causal dependencies among variables. The analysis leads to much more accurate heuristic guidance values than those of the Metric-FF heuristic used in SGPLan_4.
c) Using the new heuristic function, high-quality approximate plans can be found for resolving temporal constraints in PDDL3.0 problems efficiently.

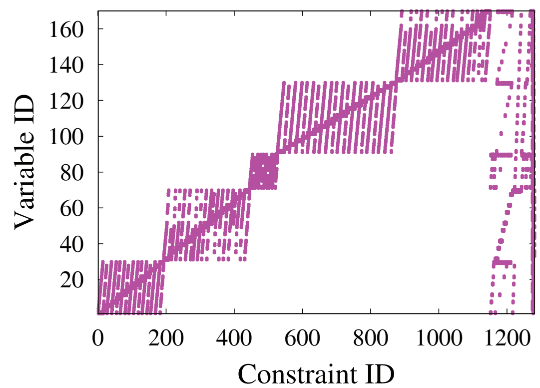

Figure 1: The locality of the constraint-variable structure in the fifth instance of the *TPP-SimplePreferences* domain.

CONSTRAINT LOCALITY

Constraint partitioning in SGPLan_5 is based on the locality of constraints observed in IPC5 planning domains. Figure 1 illustrates the constraint-variable structure in the fifth instance of the *TPP-SimplePreferences* domain. Each variable represents an action and its schedule in the plan, and a constraint can be a mutual exclusion, an inconsistent state

*Research supported by the National Science Foundation Grant IIS 03-12084.

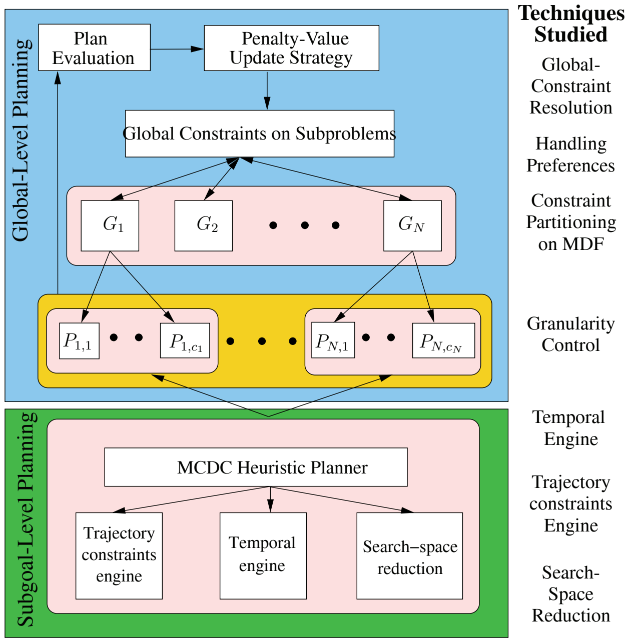

Figure 2: Architecture of SGPLan$_5$.

variable assignment, or a trajectory constraint. It is obvious that a majority of the constraints can be localized after identifying those (global) constraints with variables that have strong causal dependencies to many other variables.

We have developed SGPLan$_5$ to exploit constraint locality in all IPC5 domains. However, constraints in PDDL3.0, both hard and soft, can be over the intermediate states of a plan, in addition to those hard constraints on the final states as in PDDL2.2. Hence, we must extend SGPLan$_4$ (Chen, Wah, & Hsu 2006) that only aims to satisfy a conjunctive list of the conditions on the final states. In SGPLan$_5$, we have extended our partitioning approach based on MDF. The multi-valued domain analysis allows us to eliminate a number of mutual exclusions as well as inconsistencies among the soft constraints before the constraints are partitioned.

Note that, although constraint locality is common in all IPC5 benchmarks, the difficulty of resolving the constraints varies across domains. For instance, we have found that all the subproblems in the *OpenStacks* domain are trivial to solve, but the major challenge is to enforce the consistency of its shared variables.

ARCHITECTURE OF SGPLan$_5$

By formulating a subproblem in such a way that each has one goal state, SGPLan$_5$ partitions a planning problem into subproblems and finds a feasible plan for each subgoal (Figure 2). In the global level, it partitions the problem by its multi-valued state variables and resolves its violated global constraints using the theory of extended saddle points (Wah & Chen 2006). In the local level, it calls a basic planner for solving each partitioned subproblem, using the violated global constraints and the global preferences as biases.

Global-Level Search

Partitioning Strategy. We have observed that many constraints have a strong locality if we can identify those constraints that involve many state variables. From the causal graph, we can extract those (low-level) state variables that influence many other state variables. Dependencies due to these state variables would cause active mutual exclusions across subproblems no matter how the constraints are partitioned. On the other hand, there are (high-level) state variables whose state transitions require a set of other (low-level) variables. Since constraint locality is associated with high-level state variables, we can formulate constraints that involve state variables across partitions as global constraints. Also, we have chosen an optimal grain size that minimizes the number of shared variables in order to reduce the number of global constraints.

Resolution of Global Constraints. A planning problem solved by SGPLan$_5$ is defined in mixed space with a nonlinear objective and one or more constraints that may be procedural. SGPLan$_5$ implements a search to find extended saddle points (ESPs) of a penalty function derived from the problem (Wah & Chen 2006), where the penalty function consists of the sum of the objective and the transformed constraint functions weighted by penalties, and an ESP is a local minimum of the penalty function with respect to the original variables and a local maximum with respect to the penalties. The search algorithm is based on the extended saddle-point condition (ESPC) that shows the one-to-one correspondence between the ESPs and the feasible local optima.

An important property of the ESPC is that it is true for all penalty values larger than a minimum threshold. This property allows the search of points that satisfy the ESPC to be found iteratively, with an inner loop that looks for a local minimum of the penalty function, and an outer loop that looks for any penalty values larger than the threshold. The property also allows the search of ESPs to be partitioned into multiple searches, each looking for a local ESP in a subproblem, and an outer loop that resolves the inconsistencies among the subproblems.

A direct implementation of the ESPC in a search algorithm may get stuck in an infeasible region when the objective is too small or when the penalty values and/or constraint violations are too large. To address this issue, SGPLan$_5$ performs backtracking to escape from infeasible local traps.

Handling Constraints by the MCDC Heuristic. Since the *minimum causal dependency-cost* (MCDC) heuristic can generate a highly accurate approximate plan for each state, we can obtain a tight lower bound on the estimated makespan from each state and use it to eliminate the state when temporal constraints are violated. For temporal constraints in the form of deadlines, we prune a state whose MCDC value exceeds the deadlines. For side trajectory constraints, we prune a state whose approximate MCDC plan violates the constraints.

Handling Preferences. We have classified all trajectory preferences into two categories.

The first class of preferences consists of those soft constraints on the final state and the persistent soft constraints (model operator *always*). We consider them with the original goal definitions because they have temporal overlaps on the final state. Although it is not easy to find an optimal set of soft constraints to be satisfied, it is trivial to compute their violation cost, when given an assignment of all state variables involved in the goal preferences. Therefore, we enumerate all reachable elements of each state variable involved and choose an optimal combination of facts to achieve. The enumerations can be highly decomposed because those constraints on the final state also have strong localities. It is still possible to make an unreachable assignment, even though the MDF analysis can detect many implicit mutual exclusions. For those unreachable assignments, we perform backtracking to find alternative assignments. When the cost of the assignment (such as in the form of a weighted sum of preference violations and plan quality) is unknown until planning ends, we also perform backtracking to find better solutions after a plan has been found.

The second class of preferences are those with insufficient information on their satisfiability. This may happen because the related soft constraints are not always active. To address this issue, we have devised a *relax-and-tighten* strategy that initially ignores all those preferences belonging to this class and that penalizes those unsatisfied preferences in order to generate a solution. As is done earlier in resolving constraints, we have developed a number of heuristics for estimating the reachability of preferences and have applied iterative refinements until no better solutions can be found.

Local-Level Basic Planner

Our basic planner follows the heuristic search algorithm used in Metric-FF (Hoffmann 2003), but employs a new heuristic based on the multi-valued formulation.

MCDC Heuristic Planner. Using the multi-valued formulation, we have implemented a new search heuristic by exploring the value transition graph of each variable and the causal dependencies between the transition graphs. The general idea is inspired by and similar to the heuristic used in the Fast Downward planner (Helmert 2004). However, our MCDC heuristic is very different from the Fast Downward heuristic in a number of aspects.

First, our MCDC heuristic employs a recursive depth-first search for generating a heuristic plan with the minimum cost without pruning the causal graphs. In contrast, the Fast Downward heuristic is incomplete since it performs strongly-connected-component analysis and removes nodes with low connectivity. We have found that our complete recursive search leads to much less node expansions.

Second, we have developed a set of strong necessary conditions for pruning infeasible or dominated paths when searching for the best approximate plan. We have also developed an algorithm for detecting symmetric objects in a given state to further reduce the cost of the MCDC heuristic calculations. These pruning rules can reduce the computing time of our heuristic by one to two orders of magnitude.

Third, in addition to sequential propositional planning supported by the Fast Downward heuristic, MCDC supports parallel temporal planning and can generate estimates of makespans for temporal plans.

The heuristic plan found by MCDC is approximate because the transition graphs found are not complete and some of its actions may not be supported. Moreover, numerical and trajectory constraints are ignored in MCDC.

MCDC is not admissible because, when computing an approximate plan, we consider each subgoal individually and add up the costs of all subgoals in order to estimate the overall heuristic value. Thus, MCDC ignores the positive interactions among the subgoals and is not admissible.

Search-Space Reduction. Before solving a partitioned subproblem, we can often eliminate many irrelevant actions in its search space. We identify those relevant actions by traversing the causal graphs in MDF and by ignoring actions that are not useful for achieving the current subgoal state variables. We also prioritize actions that do not cause an inconsistent assignment of multi-valued state variables. This is done by following our partitioning setting to compute a set of local state variables for each subproblem, and by applying the helpful action idea introduced in FF (Hoffmann & Nebel 2001) to defer those actions that change the value of shared state variables.

References

Chen, Y. X.; Wah, B. W.; and Hsu, C. W. 2006. Temporal planning using subgoal partitioning and resolution in SGPlan. *J. of Artificial Intelligence Research*.

Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The language for the classic part of the 4th International Planning Competition. Technical report, Tech. Rep. 195, Institut für Informatik, Freiburg, Germany.

Gerevini, A., and Long, D. 2005. Plan constraints and preferences for PDDL3. Technical report, R.T. 2005-08-07, Dept. of Electronics for Automation, U. of Brescia, Brescia, Italy.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *ICAPS*, 161–170.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. of Artificial Intelligence Research* 14:253–302.

Hoffmann, J. 2003. The Metric-FF planning system: Translating ignoring delete lists to numeric state variables. *J. of Artificial Intelligence Research* 20:291–341. `http://www.mpi-sb.mpg.de/~hoffmann/metric-ff.html`.

van den Briel, M.; Vossen, T.; and Kambhampati, S. 2005. Reviving integer programming approaches for AI planning: A branch-and-cut framework. In *ICAPS*, 310–319.

Wah, B., and Chen, Y. X. 2006. Constraint partitioning in penalty formulations for solving temporal planning problems. *Artificial Intelligence* 170(3):187–231.