# Handling Soft Constraints and Goals Preferences in SGPlan*

***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 present the partition-and-resolve strategy in SGPlan (hereafter called $\text{SGPLan}_5$) for fully supporting all language features in PDDL3.0. Based on the architecture of SGPlan that supported PDDL2.2 (hereafter called $\text{SGPLan}_4$), $\text{SGPLan}_5$ 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 $\text{SGPLan}_5$, we generalize subgoal partitioning so that the goal state of a subproblem is no longer one goal fact as in $\text{SGPLan}_4$, 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 and for carrying out partitioning in the transformed space. The multi-valued representation leads to more efficient heuristics for resolving trajectory and temporal constraints and goal preferences.

## INTRODUCTION

In this paper, we present the partition-and-resolve strategy in $\text{SGPLan}_5$ for fully supporting all language features in PDDL3.0 (Gerevini & Long 2005). By extending the architecture of $\text{SGPLan}_4$ (Chen, Wah, & Hsu 2006) that supports PDDL2.2 (Edelkamp & Hoffmann 2004), $\text{SGPLan}_5$ partitions a large planning problem into subproblems, each with its own subgoal, and resolves those inconsistent solutions of subgoals using our extended saddle-point condition.

Inspired by real applications, Smith recently introduced the over-subscription planning problem (Smith 2004) that has a number of soft goals with different violation costs. Unlike PDDL2.2 domains whose goal state is a conjunctive list of facts, the planning task in an over-subscription planning problem entails the selection of an appropriate subset of soft goals when it is infeasible to achieve the entire set of goals. The idea has been extended in PDDL3.0 in such a way there are soft constraints over intermediate states.

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


To develop a planner that fully supports the PDDL3.0 planning models and to accommodate its new features on trajectory constraints and goal preferences, we have transformed each planning problem into a multi-valued domain formulation and have revised our partitioning strategy in $\text{SGPLan}_4$ based on the new representation. We have developed new search techniques, both at the global and the subproblem levels, for optimizing goal preferences and for resolving trajectory and temporal constraints.

## DESIGN GOALS

PDDL3.0 extends the previous PDDL2.2 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. We have developed new components in $\text{SGPLan}_5$ to support these features.

Given a plan $\pi$, an initial state $I$, a sequence of actions and possibly their schedule, we can derive the trajectory of $\pi$ under the domain definition. We can compute for this trajectory its violated constraints, which include mutex constraints as in PDDL2.2, inconsistent state-variable assignments, and trajectory constraints introduced in PDDL3.0. The objective is to satisfy all the hard constraints or goals as well as to optimize the soft-constraint violations and the plan quality.

$\text{SGPLan}_5$ uses a *multi-valued domain formulation (MDF)* based on the SAS+ formalism. MDF has been used in several planners, including Fast Downward (Helmert 2004) and the IP planner (van den Briel, Vossen, & Kambhampati 2005). Its advantage is that it allows a more compact representation of facts and their dependencies. For example, in the traditional representation of binary facts, the location of *truck1* in the *TPP* domain is represented in binary facts, such as *at(truck1, location1)*, ..., *at(truck1, location8)*. These facts can be denoted more compactly in MDF by one variable *location(truck1)* that takes multiple values: *location1*, ..., *location8*. The MDF variables further allows us to derive the possible transitions among their values. For instance, *location1* $\longrightarrow$ *location2* represents the connection between two locations.

Based on the MDF formulation, we can derive causal

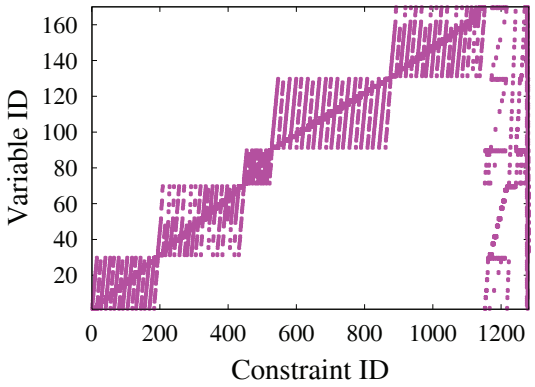


Figure 1: The locality of the constraint-variable structure in the fifth instance of the *TPP-SimplePreferences* domain.

dependencies among the variables, where variable $a$ has a causal dependency on another variable $b$ if a transition in $a$ has a state requirement on $b$. For example, in order for *truck1* to move *crate* from *location1* to *location2*, it has to be at *location1*. In this case, the transition for *crate* has a causal dependency on the location of *truck1*.

We have implemented our preprocessing engine for translating a PDDL3.0 problem into MDF. There are several reasons for using MDF.

a) It provides a more compact representation than a binary-valued representation and leads to a more effective partitioning of the constraints.

b) MDF facilitates the analysis of the transition graphs of variables and causal dependencies among the variables. These can be used to derive a much more accurate heuristic guidance than the previous Metric-FF heuristic.

c) Using the new heuristic function, high-quality approximate plans can be extracted for efficiently resolving temporal constraints in PDDL3.0 planning problems.

## CONSTRAINT LOCALITY

In PDDL2.2 planning, we have shown that constraint partitioning by subgoals leads to localized constraints when we partition the constraints in such a way that a majority of them are within a partition. This approach, however, cannot be directly applied in PDDL3.0 domains because there are some soft goals that are never present in an optimal solution. Constraint locality will also be different when problems are represented in MDF.

Because it is possible to have inconsistencies among soft constraints, our previous subgoal partitioning strategy that aims to satisfy a conjunctive list of conditions on the final state cannot be applied directly. To address this issue, we have extended our partitioning approach. Using multi-valued domain analysis, we first eliminate a number of mutual exclusions as well as inconsistencies among the soft constraints. Moreover, we have found that constraint locality is associated with some state variables with causal dependencies to other state variables. Conceptually, these

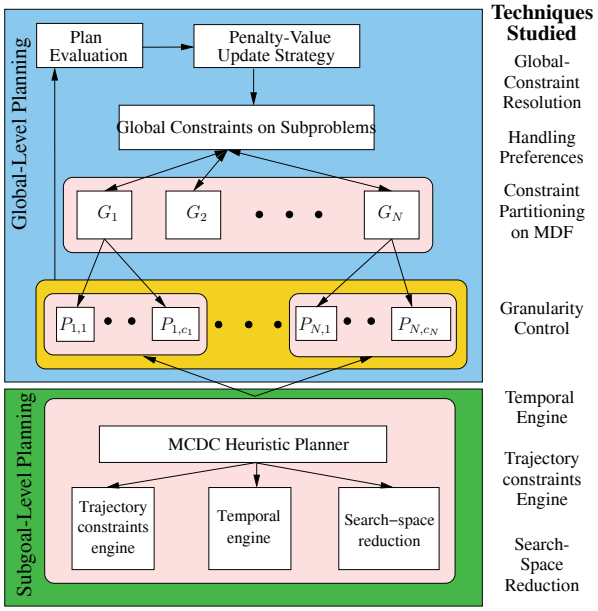


Figure 2: Architecture of SGPLan$_5$.

variables can be treated as goals because their transitions incur changes on other variables. As a result, we partition the problem by these variables and formulate global constraints as those that involve variables across different partitions.

We have evaluated our revised partitioning method on some IPC5 benchmarks. Figure 1 illustrates the constraint-variable structure in the fifth instance of the *TPP-SimplePreferences* domain that aims to plan the purchase and storage of five products. In this problem, each variable represents an action and its schedule in the plan, and a constraint can be an active mutual exclusion, an inconsistent state variable assignment, or a violated trajectory constraint. By partitioning those variables that represent the status of products (five in this instance) into five subproblems, each involving one product, it is obvious that a majority of the constraints can be localized. In this form, a global constraint involves variables that are across two partitions, such as a constraint on the equality of the quantity of two products.

Note that the difficulty of resolving constraints is domain dependent, although constraint locality is common in all IPC5 benchmarks. For instance, 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 goal fact (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 the remaining constraints have a strong locality if we can first eliminate those that involve a large number of state variables. From the causal graph, we can extract those (low-level) state variables that influence many other state variables. Dependencies due to these variables would cause active mutual exclusions across subproblems, regardless of how the constraints are partitioned. On the other hand, there are (high-level) variables whose state transitions involve a set of low-level variables. Since constraint locality is associated with high-level state variables, we can formulate constraints that involve 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. For example, in the *TPP* domain, there will be a lot of active mutual exclusions if we use the same truck to generate two subplans for buying two products, whereas the number of mutual exclusions will be minimal when two trucks are used.

**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. By formulating a penalty function that consists of the sum of the objective and the transformed constraint functions weighted by penalties, SGPLan$_5$ implements a search to find extended saddle points (ESPs) of the penalty function (Wah & Chen 2006). Here, 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 for all penalties larger than a threshold. The algorithm is based on the ESP condition (ESPC), which states the one-to-one correspondence between the ESPs and those feasible local optima in mixed space. To implement the ESPC, the search consists of two loops: an inner loop that looks for a local minimum of the penalty function and an outer loop that looks for any penalty value larger than the threshold. The ESPC 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 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 local constraints.** Since the *minimum causal dependency-cost* (MCDC) heuristic can generate a highly accurate approximate plan from a state, we can use it as a tight lower bound on the makespan when resolving temporal constraints. For temporal constraints in the form of deadlines, we prune any state whose MCDC value exceeds the deadlines. For trajectory constraints, we evaluate the approximate MCDC plan and add penalty for those that are violated in the approximate MCDC plan.

**Handling preferences.** There are three classes of methods developed for handling soft goals in a planning problem. The first class solves the problem by encoding it as an integer program, as in OptiPlan (van den Briel *et al.* 2004). The second class first heuristically selects a subset of goals and then applies an existing planner to achieve them (Nigenda & Kambhampati 2005). Last, the third class does not select a subset of goals upfront but treats soft goals as soft planning constraints (Benton, Do, & Kambhampati 2005). It then derives new heuristics and reachability analysis for each soft goal in order to guide the search to an optimal final state. The success of these approaches based on heuristic functions, however, depends on the assumption that either all the soft goals are independent or their interactions have been addressed in the heuristic function (Smith 2004).

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. These enumerations can be 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 a weighted sum of preference violations and plan quality) is unknown until the end of planning, we also perform backtracking to find better solutions.

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 ignores initially all those preferences belonging to the second class and that penalizes those unsatisfied preferences 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.

Note that the penalties on constraints are independent of the preference weights. For those soft constraints in the first class, they are enforced by the basic planner because they are local constraints when we partition by state variables. On the other hand, we update the penalties of constraints in the second class based on their violation but not their weights. Because we rely on changes in penalty values in order to find a better solution in terms of the plan metrics, our strategy does not guarantee plan optimality.

## 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 MDF.

**MCDC heuristic planner.** Using MDF, 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 complete 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 during planning.

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 evaluating the MCDC heuristic. These rules can help reduce the average 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 sum 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) in order to defer those actions that change the value of non-local state variables.

## Preliminary Experimental Results

We have defined predicates and functions for capturing the violation and the corresponding cost of each constraint. We have modified the parser and the pre-processor of SGPLan$_4$ in such a way that all the constraints are grounded and all ADL features in their conditions are compiled away. We have also integrated all the techniques for handling soft constraints in the search engine of SGPLan$_5$, both at the top-level of the search and in the local-level basic planner. At this time, we have evaluated SGPLan$_5$ on four IPC5 benchmarks: *TPP*, *OPENSTACKS*, *TRUCKS*, and *STORAGE* and have obtained promising results and solutions with good quality. Further, for the *SimplePreferences* domain, if the objective value is only determined by the final state, then we can compute the optimal assignment of the final state by only considering those constraints on the final state and by ignoring all other constraints. Since the solution is the optimal assignment of a relaxed problem, it is an optimal solution to the original problem when the assignment is actually achieved.

## References


Benton, J.; Do, M. B.; and Kambhampati, S. 2005. Oversubscription planning with numeric goals. In *IJCAI*, 1207–1213.

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.

Nigenda, R. S., and Kambhampati, S. 2005. Planning graph heuristics for selecting objectives in oversubscription planning problems. In *ICAPS*, 192–201.

Smith, D. E. 2004. Choosing objectives in oversubscription planning. In *ICAPS*, 393–401.

van den Briel, M.; Nigenda, R. S.; Do, M. B.; and Kambhampati, S. 2004. Effective approaches for partial satisfaction (over-subscription) planning. In *AAAI*, 562–569.

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.