Appears in Proc. IJCAI 2007, Hyderabad, India

# Planning with Goal Utility Dependencies

**Minh B. Do**† and **J. Benton** ‡ and **Menkes van den Briel** ‡ and **Subbarao Kambhampati** ‡

† Embedded Reasoning Area, Palo Alto Research Center, Palo Alto, CA 94304, USA, minh.do@parc.com

‡ CSE Department Arizona State Univ. Tempe, AZ 85287, USA, {j.benton,menkes,rao}@asu.edu

## Abstract

Work in partial satisfaction planning (PSP) has hitherto assumed that goals are independent thus implying that they have additive utility values. In many real-world problems, we cannot make this assumption. In this paper, we motivate the need for handling various types of goal utility dependence in PSP. We provide a framework for representing them using the General Additive Independence model and investigate two different approaches to handle this problem: (1) compiling PSP with utility dependencies to Integer Programming; (2) extending forward heuristic search planning to handle PSP goal dependencies. To guide the forward planning search, we introduce a novel heuristic framework that combines cost-propagation and Integer Programming to select beneficial goals to find an informative heuristic estimate. The two implemented planners, *iPUD* and *SPUDS*, using the approaches discussed above, are compared empirically on several benchmark domains. While *iPUD* is more readily amendable to handle goal utility dependencies and can provide bounded optimality guarantees, *SPUDS* scales much better.

## 1 Introduction

Classical planning aims at finding a plan that achieves a set of conjunctive goals. Partial satisfaction planning (PSP) relaxes this all-or-nothing constraint, focusing on finding a plan that achieves the "best" subset of goals (i.e. the plan that gives the maximum trade-off between total achieved goal utilities and total incurred action cost). The process of finding goals on which to focus is complicated by the fact that they interact with one another. For instance, actions may share in their achievement of goals (positive interaction) or conflict (negative interaction). These types of interactions introduce *cost dependencies* between goals because the cost of achieving them separately may differ from the cost of achieving them together. In the existing frameworks, goals only interact through cost dependencies. In this paper we extend PSP to handle *utility dependency* which allows users to specify changes in utility on sets of goals, thereby increasing its expressive power.

Two concrete examples of utility dependency are *mutual dependency* and *conditional dependency*. For mutual dependency, the utility of a set of goals is different from the sum of the utility of each individual goal. For example, (1) while the utility of having either a left or right shoe alone is *zero*, utility of having both of them is much higher (i.e. the goals "complement" each other); (2) utility of having two cars is smaller than the sum of the individual utilities of having each one of them (i.e. the goals "substitute" each other). Conditional dependency is where the utility of a goal or set of goals depend on whether or not another goal or set of goals is already achieved. For example, the utility of having a hotel reservation in Hawaii depends on whether or not we have already purchased a ticket to Hawaii.

The main technical challenges in handling utility dependencies are in finding (1) a model where different types of goal utility dependencies can be compactly expressed and (2) an approach that effectively combines utility interactions with cost interactions to find a high quality plan. The primary contribution of our paper is a systematic approach for handling cost and utility dependencies together in PSP. In particular we present:

- An approach for representing utility dependencies between planning goals using the *Generalized Additive Independence (GAI)* model [Bacchus and Grove, 1995], combining utility theory and deterministic planning.
- An extension of a state of the art Integer Programming (IP) planner [van den Briel *et al.*, 2005] to solve this extended PSP problem.
- A novel heuristic framework combining cost propagation and an IP encoding to capture mutual dependencies of goal achievement cost and goal utility. This leads to informative heuristics used for guiding a variation of a forward state space search planner.

The two approaches (IP encoding and forward heuristic search) are implemented, resulting in two planners *iPUD* and *SPUDS*, and tested on several planning benchmark domains. While the IP approach is more readily amendable to handle goal utility dependencies and can provide bounded optimality guarantees, we shall see that the heuristic search method scale much better.

## 2 Problem Formulation

A classical planning problem is a 4-tuple $\langle F, I, G, A\rangle$ where: fluents $F$ are a set of predicate symbols; initial state $I$ is completely defined by predicates in $F$; goal state $G$ is partially defined by a set of predicates in $F$; a set of actions $A$ with $a \in A$ are defined by pre- and post-conditions $Pre(a), Add(a), Delete(a) \subseteq F$. A plan $P$ is a sequence of actions that when executed from $I$ achieves all goals $g \in G$.

In partial satisfaction planning (PSP) [Smith, 2004; van den Briel *et al.*, 2004], goals $g \in G$ have utility values $u_g \geq 0$, representing how much each goal is worth to the user; each action $a \in A$ has an associated execution cost $c_a \geq 0$, representing how costly it is to execute each action (e.g. amount of time or resources consumed). Let $G_P \subseteq G$ be the set of goals achieved by a plan $P$. The objective is to find a plan $P$ that maximizes the difference between total achieved utility $u(G_P)$ and total cost of all actions $a \in P$ (i.e., plan benefit):

$$\underset{P}{\text{MAX}} \quad u(G_P) - \sum_{a \in P} c_a \tag{1}$$

Work on PSP has until now assumed that goals have no utility dependencies and thus their utilities are additive: $u(G_P) = \sum_{g \in G_P} u_g$. In order to represent the types of goal utility dependencies discussed in the previous section (i.e., complement, substitute, conditional), we adopt the *Generalized Additive Independence* (GAI) model [Bacchus and Grove, 1995]. We chose this model because it is expressive, general and we can compile to it from other commonly used models such as UCP-Net [Boutilier *et al.*, 2001]. We assume that the utility of the goal set $G$ can be represented by *k* local utility functions $f^u(G_k) \in R$ over sets $G_k \subseteq G$. For any subset $G' \subseteq G$ the utility of $G'$ is:

$$u(G') = \sum_{G_k \subseteq G'} f^u(G_k) \tag{2}$$

This model allows users to specify changes in utility over sets of goals. For the rest of this paper, we name the new PSP problem with utility dependencies represented by the GAI model $PSP^{\mathcal{UD}}$. If there are $|G|$ local functions $f^k(G_k)$ and each $G_k$ contains a single goal then $PSP^{\mathcal{UD}}$ reduces to the original PSP problem (no utility dependencies).

## 3 IP Encoding for $PSP^{\mathcal{UD}}$

Since classical planning can be solved by IP, and since IP provides a natural way to incorporate numeric constraints and objective functions, it follows that $PSP^{\mathcal{UD}}$ planning problems can be solved by IP as well.

We setup an IP formulation to handle $PSP^{\mathcal{UD}}$ problems by extending the generalized single state change (G1SC) formulation [van den Briel *et al.*, 2005]. Currently, the G1SC formulation is the most effective IP formulation for solving classical planning problems, and it outperforms the previously developed IP formulation used to solve PSP problems without utility dependencies [van den Briel *et al.*, 2004].

The G1SC formulation represents the planning problem as a set of loosely coupled network flow problems, where each network corresponds to one of the state variables in the planning domain. The network nodes correspond to the state variable values and the network arcs correspond to the value transitions. The planning problem is to find a path (a sequence of actions) in each network such that, when merged, they constitute a feasible plan. In the networks, nodes and arcs appear in layers, where each layer represents a plan period. The layers are used to solve the planning problem incrementally. That is, we start by performing reachability analysis to find a lower bound on the number of layers necessary to solve the problem. If no plan is found, all the networks are extended by one extra layer and the planning problem is solved again. This process is repeated until a plan is found (see [van den Briel *et al.*, 2005] for a complete description of the G1SC formulation).

In order to deal with utility dependencies we incorporate the following extensions to the G1SC formulation:

- In $PSP^{\mathcal{UD}}$ problems not all goals have to be achieved for a plan to be feasible. Therefore, we remove those constraints from the G1SC formulation which state that goals must be achieved.
- For each goal utility dependency function $G_k$ we add a variable $z_{G_k} \in \{0, 1\}$, where $z_{G_k} = 1$ if all goals in $G_k$ are achieved, and $z_{G_k} = 0$ otherwise.
- For each goal utility dependency function $G_k$ we add constraints to ensure that $G_k$ is satisfied if and only if all goals $g \in G_k$ are achieved, that is:

$$\sum_{f, g \in D_c : g \in G_k} y_{c,f,g,T} - |G_k| + 1 \leq z_{G_k} \tag{3}$$

$$z_{G_k} \leq \sum_{f \in D_c} y_{c,f,g,T} \quad \forall g \in D_c : g \in G_k \tag{4}$$

  where $D_c$ is the domain of a state variable $c$, $y_{c,f,g,T} \in \{0, 1\}$ are variables of the IP problem that represent value changes in the state variables, and $T$ is the plan horizon.
- We create an objective function to maximize the net-benefit (utility minus cost) of the plan.

$$\text{MAX} \quad \sum_{G_k} u(G_k) z_{G_k} - \sum_{a \in A, 1 \leq t \leq T} c_a x_{a,t} \tag{5}$$

  where $u(G_k)$ represents the utility of satisfying the goal utility dependency function $G_k$, and $c_a$ represents the cost of executing action $a \in A$.

The extended G1SC formulation is bounded length optimal (i.e., it generates optimal plans for a plan horizon $T$). Global optimality cannot be guaranteed as there could still be solutions with higher net benefit at longer plan horizons.

## 4 Heuristics for Maximizing Plan Benefit

While IP planners are optimal within a bounded horizon, prior work suggests that heuristic search planners are more efficient and are able to find better solutions in larger problems [van den Briel *et al.*, 2004]. For $PSP^{\mathcal{UD}}$ problems, the challenge in computing a heuristic for maximizing plan benefit is that we need to simultaneously (1) pick the best set of goals and (2) find the best plan for them. While the first challenge is complicated by the utility dependencies, the second is complicated by the cost dependencies.

Current heuristic approaches for solving PSP problems involve planning graph based heuristics that handle cost dependencies by providing an estimate of the cost for achieving

each goal. The heuristics then use the goals' defined utility values to select goal subsets that offer the best net benefit on the estimated cost. In this scenario goals have no dependencies on one another. In *PSP*$^{\mathcal{UD}}$ we must deal with the fact that both cost dependencies between actions and utility dependencies between goals exist.

In this section we define our search algorithm along with three heuristics that combine planning graph methods with a declarative IP encoding. We generate an IP encoding over the relaxed plan heuristic rather than the entire problem as discussed in Section 3. By solving the IP encoding, we select a goal set along with an estimated cost for achieving it. Our main innovation is the combination of a relaxed plan that handles cost interactions between goals and a declarative IP encoding that captures both mutual goal achievement cost and goal utility dependencies.

**Best-First Heuristic Search for *PSP*$^{\mathcal{UD}}$ :** To handle *PSP*$^{\mathcal{UD}}$ , we use the best-first anytime heuristic search framework in *Sapa*$^{\mathcal{PS}}$ [van den Briel *et al.*, 2004]. This forward metric temporal planner uses an anytime variation of the $A^*$ algorithm guided by a heuristic derived from the relaxed planning graph. Like $A^*$, this algorithm starts with the initial state $S_{init}$ and continues to dequeue from the open-list the most promising node $S$ (i.e. highest $f(S) = g(S) + h(S)$ value). For each search node $S$, $g(S)$ represents the benefit achieved so far by visiting $S$ from $S_{init}$ and $h(S)$ represents the projected maximum additional benefit gained by expanding $S$, with plan benefit defined in Section 2. Though calculating $g(S)$ is trivial, having a good estimate of $h(S)$ is hard and key to the success of best-first search algorithms. During exploration of the search tree the algorithm, which is called $A^*_{PSP}$, keeps outputting better quality plans whenever a node $S$ with the best-so-far $g(S)$ value is expanded. Like $A^*$, the algorithm terminates when a node $S$ with $h(S) = 0$ is picked from the top of the open list.

Though $A^*_{PSP}$ can keep all generated search nodes, in practice *Sapa*$^{\mathcal{PS}}$ prunes the search space by removing nodes that that appear *unpromising* (i.e., nodes where the estimated benefit is negative). Though this improves efficiency, one potential drawback is that when the heuristic $h(S)$ underestimates the value of a search node $S$, then $S$ will be discarded (when compared to the benefit of the best solution found so far $g(S_B)$) even if it can be extended to reach a better solution. To mitigate this issue, we modified the *Sapa*$^{\mathcal{PS}}$ search strategy to keep some search nodes that appear unpromising when first generated. During search we set a value $\epsilon$ as half the distance between the best node found so far $S_B$ and the worst-valued unpromising node. For each unpromising search node $S$ that is within a threshold $\epsilon$ of the current best solution, we find $\rho$, the complement of the percentage distance between it and the benefit of $S_B$ (i.e. $g(S_B)$). We then keep $S$ with probability $\rho$.

**Goal Achievement Cost Propagation:** In all of our heuristic methods we estimate the cost $\mathcal{C}(g)$ to achieve each goal [Do and Kambhampati, 2004]. Starting with $\mathcal{C}(f) = 0$ for facts $f$ in the initial state $I$ and $\mathcal{C}(f) = \mathcal{C}(a) = \infty$ for all other facts and all actions, the propagation rules to estimate costs to achieve facts $p$ and to execute actions $a$ are[1]:

- Facts: $\forall f : \mathcal{C}(f) = \underset{f \in Add(a)}{MIN} (\mathcal{C}(a) + c_a)$
- Either:
  1. Max-prop: $\forall a \in A : \mathcal{C}(a) = \underset{f \in Pre(a)}{MAX} \mathcal{C}(f)$; or
  2. Sum-prop: $\forall a \in A : \mathcal{C}(a) = \underset{f \in Pre(a)}{\Sigma} \mathcal{C}(f)$

The update rules are used while extending the (relaxed) planning graph structure [Blum and Furst, 1997]. After the propagation is done (i.e., no costs change), $\mathcal{C}(g)$ is an estimate on the cost to achieve $g$ for each goal $g \in G$.

**Deriving Heuristics from Propagated Costs:** We will use the notation $h^x_y$ to name our heuristics. Here $x$ is the method used to estimate the goal utilities and $y$ is the method used to estimate the goal costs. If utilities and costs are independent, both $x$ and $y$ can be "sum". The dependencies between goal utilities can be estimated using the GAI model while the dependencies between goal costs can be estimated using relaxed plans.[2]

It is easy to observe that if we use *max* propagation (max-prop), then $\mathcal{C}(g)$ will underestimate the cost to achieve $g$ while there is no such guarantee for *sum* propagation (sum-prop) [Bonet *et al.*, 1997]. Using $\mathcal{C}(g)$ calculated by the cost propagation process outlined in the previous section, we can estimate the achievable benefit value as below:

$$h^{GAI} = \underset{G' \subseteq G}{MAX} \; [u(G') - (\underset{g \in G'}{MAX} \mathcal{C}(g))] \qquad (6)$$

Notice that, as part of our heuristic, we the calculate the local utility functions as defined in Equation 2. As such, our heuristic directly applies the GAI model. If we use max-prop, then Equation 6 will give the $h^{GAI}_{max}$ heuristic and if we use sum-prop, it will give a corresponding $h^{GAI}_{sum}$ heuristic. While $h^{GAI}_{max}$ overestimates the real achievable benefit, there is no such guarantee for $h^{GAI}_{sum}$. Thus $A^*_{PSP}$ using $h^{GAI}_{max}$ is guaranteed to output an optimal solution given enough time.

**Relaxed Plan based Heuristic:** Because $h^{GAI}_{max}$ can significantly over-estimate the real benefit of achieving a set of goals, it performs badly in some domains (as we shall see). The $h^{GAI}_{sum}$ heuristic, while more informative, relaxes the cost interaction and assumes that plans achieving different goals are independent and do not overlap. To improve on those two heuristics, we adapt the relaxed plan heuristic, first introduced in the FF planner [Hoffmann and Nebel, 2001], that solves the planning problem ignoring the delete list. This heuristic improves over $h^{GAI}_{sum}$ by taking into account actions contributing to the achievement of several goals. The challenge in extending it to *PSP*$^{\mathcal{UD}}$ is how to efficiently find a high-benefit

[1] $c_a$, which is the execution cost of $a$, is different from $\mathcal{C}(a)$, which is the cost to enable the execution of $a$ (i.e., costs to achieve preconditions of $a$)

[2] Given this notation, we can view the heuristic used in *Sapa*$^{\mathcal{PS}}$ [Do and Kambhampati, 2004] as $h^{sum}_{relax}$ because it sums the individual goal utilities and extracts a relaxed plan to estimate cost.

relaxed plan in the presence of both cost and utility dependencies.

Let $G_{P^+} \subseteq G$ be the set of goals achieved by the relaxed plan $P^+$. The relaxed plan heuristic for $PSP^{\mathcal{UD}}$ is:

$$h^{*\ GAI}_{relax} = \underset{P^+}{\text{MAX}} \quad u(G_{P^+}) - \sum_{a \in P^+} c_a \qquad (7)$$

Note that Equation 7 looks like Equation 1 except that the optimal plan $P$ in Equation 1 is replaced by the optimal relaxed plan $P^+$ (i.e. achieving maximum benefit) in Equation 7. $h^{*\ GAI}_{relax}$ overestimates the real achievable benefit and can be used as an admissible heuristic in $A^*_{PSP}$ to find the optimal solution for PSP problems.

While finding a satisfying relaxed plan $P^+$ for any given goal set $G_{P^+} \subseteq G$ is polynomial, extracting $h^{*\ GAI}_{relax}$ requires finding an optimal relaxed plan (highest benefit). This task is NP-hard even when we already know the optimal goal set $G^*_{P^+}$ and actions have uniform cost [Bylander, 1994]. To approximate $h^{*\ GAI}_{relax}$ for $PSP^{\mathcal{UD}}$ we use the following three steps. The first step was introduced in $Sapa^{\mathcal{PS}}$ while the second and third steps are novel:

1. Greedily extract a low cost relaxed plan $P^+$ that achieves the *largest* set of achievable goals.
2. Capture the achievement cost dependencies between achievable goals using the causal structure of $P^+$.
3. Pose the problem of extracting the optimal subplan within $P^+$ that takes both cost and benefit dependencies into account as an IP encoding. A solution $h^{GAI}_{relax}$ of this IP encoding is used to estimate $h^{*\ GAI}_{relax}$.

Notice that if we compile the entire relaxed planning graph (rather than just the greedily extracted relaxed plan) we can post the problem of finding $h^{*\ GAI}_{relax}$ as an IP problem. Despite its conceptual elegance, we chose not to follow this route as the cost of heuristic computation increases significantly (especially for our progression planner which extracts a relaxed plan at each node).

**Step 1: Heuristically Extract a Low Cost Relaxed Plan**

Let $G' \subseteq G$ be the set of all achievable goals ($\mathcal{C}(g) < \infty$), we use the planning graph and the propagated achievement costs in Section 4 to heuristically extract a low-cost relaxed plan to support $G'$ as follows:

1. Start with supported facts $SF = I$, subgoal set $SG = G' \setminus I$ and the relaxed plan $P^+ = \emptyset$.
2. For each $g \in SG$ select a supporting action $a : g \in Add(a)$ with lowest execution cost $\mathcal{C}(a)$ value. Update: $P^+ \leftarrow P^+ \cup \{a\}$, $SG \leftarrow SG \cup (Pre(a) \setminus SF)$ and $SF \leftarrow SF \cup Add(a)$.
3. Repeat until $SG = \emptyset$.

This backtrack-free process is guaranteed to finish in time polynomial in the number of actions.

**Step 2: Build Cost Dependencies within $P^+$**

Because certain actions contribute to the achievement of multiple goals, there are dependencies between the costs to achieve them. Those relations can be discovered by using the causal structure of the extracted relaxed plan $P^+$.

To capture the mutual dependencies between the goal achievement costs, we find the set of actions shared between different partial plans achieving different goals. This procedure utilizes the causal links in the relaxed plan $P^+$.

1. Initialize: $\forall a \in P^+ : GS(a) = \emptyset$; $\forall\ p \in Add(a) \bigcup Prec(a) : GS(p) = \emptyset$; $\forall\, g \in G : GS(g) = \{g\}$.
2. Backward sweep from goals achieved by $P^+$ and update until fix-point:
   $GS(a) = \bigcup_{p \in Add(a)} GS(p); GS(p) = \bigcup_{p \in Pre(a)} GS(a)$

Using the procedure above, for each action $a$, $GS(a)$ contains the set of goals $g$ that $a$ contributes to, where the goal-supporting sets $GS(a)$ represents the achievement cost dependencies between goals.

**Step 3: Estimate the Maximum Achievable Benefit**

In this step, we combine the goal supporting set $GS(a)$ found in the previous step with the goal utility dependencies $f^u$ to find the most beneficial relaxed plan $P'$ within $P^+$. One simple approach to find this plan $P' \subseteq P^+$ is to iterate over $2^{|G_{P^+}|}$ subsets $G' \subseteq G_{P^+}$ of goals, with $G_{P^+}$ is the set of goals achieved by $P^+$, and compare the benefit of plans $P'$ achieving $G'$. However, when $|G|$ is large this approach becomes impractical.[3] Therefore, we use a declarative approach of setting up an IP encoding with solution representing the most beneficial relaxed plan $P' \subseteq P^+$. Note that while IP is generally slow for solving a complete planning problem, the number of actions in the relaxed plan is much smaller (by several orders) than the number of actions in the (relaxed) planning graph. This allows us to find an optimal solution in reasonable time. This is critical because we will build an IP encoding for each search node generated during forward heuristic search. The IP formulation that we set up is very similar to the one described in Section 3 but it only allows actions that are in the relaxed plan and it does not create constraints for negative interactions. Moreover, additional constraints representing the goal supporting set $GS(a)$ found in the previous step are also included to enforce that if a given goal $g$ is selected, then any action that contributes to the achievement of $g$ should also be selected. Specifically:

- Binary Variables:
  - $\forall a \in P, \forall g \in G, \forall G_k \subseteq G,\ f^u(G_k) \neq 0$: create one binary integer variable $X_a$, $X_g$, $X_{G_k}$.
- Constraints:
  - $\forall a \in P, \forall g \in GS(a) : (1 - X_g) + X_a \geq 1$
  - $\sum_{g \in G_k} (1 - X_g) + X_{G_k} \geq 1$
  - $\forall g \in G_k :\ (1 - X_{G_k}) + X_g \geq 1$
- Objective: MAX $(\sum f^u(G_k) * X_{G_k} - \Sigma X_a * c_a)$

Solving this IP encoding gives the benefit value of the most beneficial relaxed plan $P'$ within $P^+$. The benefit of this $P'$ plan can be used as a $h^{GAI}_{relax}$ heuristic to guide $A^*_{PSP}$.[4]

[3] In the empirical section, we test with problems used in the planning competition, which can have more than 40 goals.

[4] We use similar IP encoding to derive the heuristic estimates for $h^{GAI}_{max}$ and $h^{GAI}_{sum}$ as described in the earlier part of this section.

## 5 Empirical Results

We have implemented the heuristic framework on top of the *Sapa*$^{\mathcal{PS}}$ planner along with the extension to the G1SC encoding discussed in Section 3. We call the new planners *SPUDS* and *iPUD* and compare (1) *iPUD*; (2) *SPUDS* with three heuristics ($h^{GAI}_{relax}$, $h^{GAI}_{max}$, and $h^{GAI}_{sum}$); and (3) a version of *Sapa*$^{\mathcal{PS}}$ whose heuristic ignores the goal utility dependencies (but whose g-value does not)

*iPUD* runs with CPLEX 10.0, a commercial LP solver, while we use *lp_solve* version 5.5 (a free solver with a Java wrapper) to solve the IP encodings in *SPUDS*. We found that *lp_solve*, while less powerful than CPLEX, has a shorter setup time and is more suitable for *SPUDS*, which sets up an IP encoding at every search node. All tests use a P4 2.66GHz/1GB RAM computer with a 600 second time limit. *SPUDS* and *Sapa*$^{\mathcal{PS}}$ continuously find better solutions until a termination criterion is met.

**Test Problems:** We automatically generated the *PSP*$^{\mathcal{UD}}$ problems from a subset of the propositional planning benchmarks used in IPC3 and IPC5: In *Zenotravel*, airplanes move people between cities; in *Satellite*, satellites turn to objects and take pictures; in *Rovers*, sets of rovers navigate to take samples and images; and in *TPP*, trucks visit markets to buy products.

For each domain, we implemented a Java program that parses the original problem files and generates the *PSP*$^{\mathcal{UD}}$ version with action cost and goal utilities randomly generated within appropriate upper and lower bounds. The set of goal dependencies along with their utility values are also randomly generated. Thus, the number of dependencies, size of the dependencies, set of goals involved, utility values and action costs are all selected within varied lower and upper bounds for each domain. All goals are soft, and therefore planners can trivially solve each problem with the null plan.

We varied our bounds on action cost and goal set utility values such that each domain focuses on different aspects of utility dependency. In Zenotravel, ending a plan with people at various locations increases utility significantly. But flying a person from one location to another has a cost that is only slightly less than the individual utilities of achieving each goal. Thus, it is vital to have sets of people at their desired location. In TPP, purchasing items has a cost about equivalent to the individual utility of having the item. However, having items together can change the utility of a plan considerably. The idea is to simulate the benefit of having several items together (e.g., to build a crate you need wood, nails, a hammer and saw). The Satellite domain removes the emphasis on cost. Here actions have costs lower than the comparatively higher benefit of having several images together (e.g., to produce a mosaic image). The domain also adds negative goal utility dependencies (i.e., substitution) by including negative utility for having certain sets of images yet ending a plan by pointing to an inconvenient spot.[5] The Rovers domain focuses on substitution as having certain scientific data together gives redundant information and therefore removes a large portion of utility gained by having them separately.

[5]In this case we group the goal of having certain images along with a final ending position as a dependency with a randomly generated negative utility.

**Analysis:** The empirical evaluation is designed to: (1) compare the effectiveness of the IP-based and heuristic search approaches in solving *PSP*$^{\mathcal{UD}}$ problems; (2) determine the characteristics of *PSP*$^{\mathcal{UD}}$ planning problems which cause our approaches to return plans of differing quality.

Our results in Figure 1 show the plan quality achieved by each planning method and the time to reach that quality. On problems where only the null plan was found, we indicate the extensive search for a better plan by setting the time to 600 seconds. For every other instance, the time that the best plan was found is shown. As the figure shows, the tested approaches varied in their relative plan quality on each domain but *SPUDS* using the $h^{GAI}_{relax}$ heuristic always performed among the best.

First, we observe the planner behavior in Zenotravel and TPP. Both of these domains involve gathering objects, though Zenotravel focuses on delivering these objects as well. Utility dependencies play an important role in these domains, and we see that *Sapa*$^{\mathcal{PS}}$ does poorly, while the *SPUDS* heuristics and *iPUD* fared much better. Since the *Sapa*$^{\mathcal{PS}}$ heuristic is not informed about utility dependencies, this comes as no surprise. In easier problems, the $h^{GAI}_{sum}$ heuristic tends to return plans of similar or equal quality as compared with the other techniques used. However, as problem size increases, $h^{GAI}_{sum}$ begins to return plans of better quality, but still does worse than $h^{GAI}_{relax}$ in terms of the overall number of plans found with best quality. With *iPUD*, as the size of the problem encoding increases it is unable to find a good feasible solution.

For our version of the Satellite domain, some goal combinations remove utility from the overall quality of plans. Also, the plans of higher quality tend to require many actions. This can be seen in the quality of the plans that *iPUD* returns. Its reachability analysis is unable to properly estimate the distance to goals and it therefore begins its solution searching at a small horizon. For the $h^{GAI}_{relax}$ heuristic, it turns out that action selection helps guide search toward the goals.

For the Rovers domain, *iPUD* does well on several problems. However, like in the Satellite domain, it turns out that better quality plans require a larger horizon on some of the problems than its initial horizon provides. This gives *SPUDS* with the $h^{GAI}_{relax}$ heuristic an edge over *iPUD* in 8 of the 20 problems. The heuristics $h^{GAI}_{sum}$ and $h^{GAI}_{max}$ have information regarding utility dependencies, though $h^{GAI}_{sum}$ often performs worse than $h^{GAI}_{relax}$ (solving 5 of 20 problems with better quality plans) and $h^{GAI}_{max}$ is only able to find the null plan in every problem instance. Neither of these heuristics can detect the individual dependencies between actions.

Also of interest is the time it takes to solve each problem between the heuristic search methods and the IP encoding used in *iPUD*. Since the *SPUDS* heuristics solve an IP encoding at each search node, they take much longer to compute on larger problems than the procedural *Sapa*$^{\mathcal{PS}}$ heuristic. Unfortunately, *Sapa*$^{\mathcal{PS}}$ lacks the heuristic guidance necessary to properly select goals with utility dependencies. Though we found that the per-node IP encoding of $h^{GAI}_{relax}$ increased the

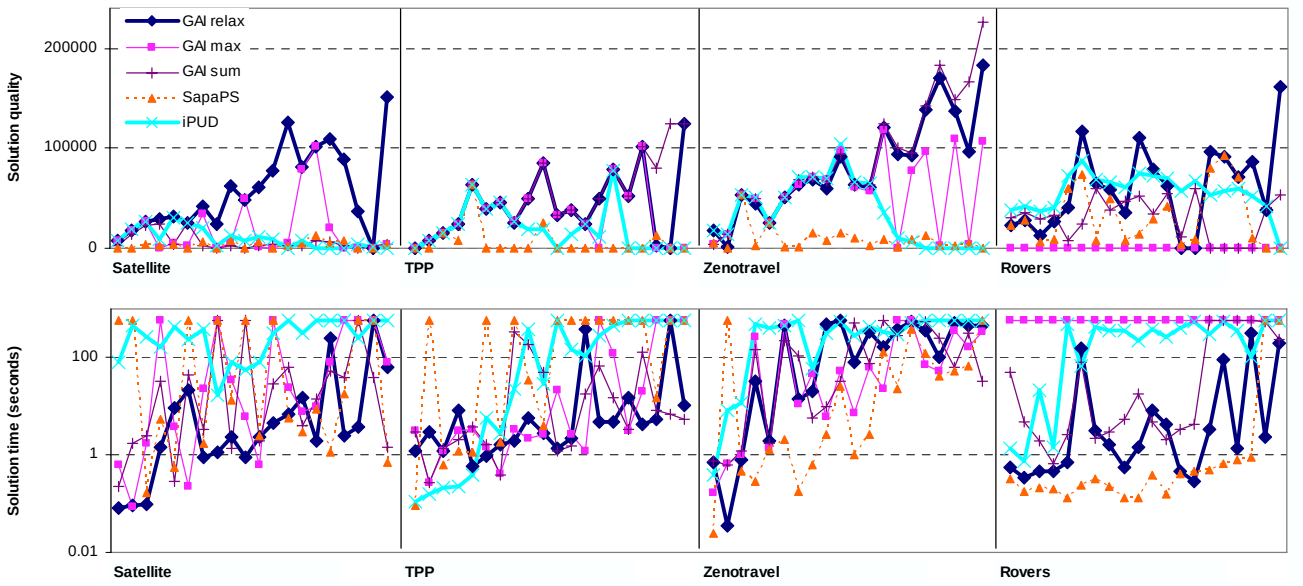


Figure 1: The problem results. For each domain we show the results of the problems solved (with later problems within the domain having increasing difficulty). The top graph shows the quality of the solutions found and the bottom graph shows the time taken to find the solutions for each of our approaches.

amount of time spent per search node by 3 to 200 times over that of *Sapa*$^{\mathcal{PS}}$ (with the highest increases on larger problems), *SPUDS* with the $h^{GAI}_{relax}$ heuristic does better overall.

When reaching the time limit (600 seconds for our results), *Sapa*$^{\mathcal{PS}}$ , *SPUDS* and *iPUD* return their best solution. In *SPUDS* and *Sapa*$^{\mathcal{PS}}$ this behavior comes from the best first anytime search and with *iPUD* this behavior comes from the CPLEX solver, which can return the best feasible solution found within a given time limit. Insights can be obtained by observing the amount of time it takes to find the solution that is eventually returned. We used the anytime behavior to illustrate the scalability of each approach. Figure 2 shows, of problems 10 through 20 in each domain (i.e., the most difficult), which each technique performs best in terms of quality throughout their search (e.g., $h^{GAI}_{relax}$ has the best quality for 16 of the problems at 2 seconds). Of our approaches, $h^{GAI}_{relax}$ performs the best overall. In the 80 tested problems, it solves 22 instances at 600 seconds better than any other planner. Also interesting is that in 45 instances it obtains the best plan of the approaches or one of similar quality (by "similar" we mean within 0.1% of the best solution). Figure 3 shows an example of the anytime behavior in a typical problem (Zenotravel problem 13). We can see that all of the *SPUDS* heuristic approaches quickly find a high quality plan, *iPUD* continues to improve its solution until the time limit, and *Sapa*$^{\mathcal{PS}}$ is only able to find a poor quality plan.

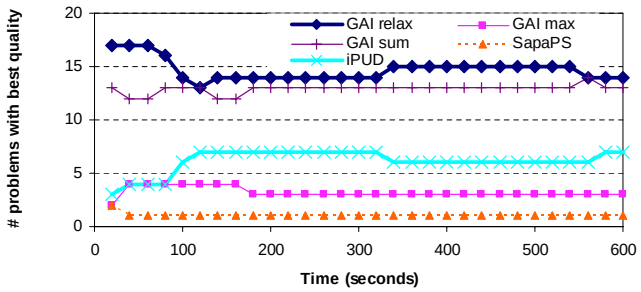


Figure 2: The number of highest quality solutions found during anytime search by each planner from problems 10 through 20 in the domains.

## 6 Related Work

There has been work on PSP problems using *orienteering* to select goal subsets by David Smith (2004). Also, van den Briel et. al. (2004) introduced several planners such as *AltAlt*$^{ps}$ , *Sapa*$^{\mathcal{PS}}$ , and $OptiPlan$ that tackle PSP by either greedily selecting goals up-front, heuristically searching for solutions, or compiling the problem into IP. None of those planners deal with the utility dependencies we described. The most recent International Planning Competition [Gerevini *et al.*, 2006] included problems with preferences that involved indicating costs on plans that failed to meet preferred constraints. Also, PrefPlan [Brafman and Chernyavsky, 2005] can find optimal plans with preferences between goals.

Besides the GAI model that we used to represent the utility dependencies, there are several other attractive models such as UCP-Net [Boutilier *et al.*, 2001] and the graphical model [Bacchus and Grove, 1995]. While both provide a graphical representation that can make it easier for users to understand the dependencies, the GAI model is more general and these modelling methods can be compiled into GAI. We also note that PDDL3 can represent GAI utility dependencies, albeit in an unnatural way.[6]

It is possible to solve PSP problems by modelling them as MDPs [Boutilier *et al.*, 1999] and extracting a solution from

[6]We can represent utility by compiling it into "positive cost" that a PDDL3 metric maximizes.

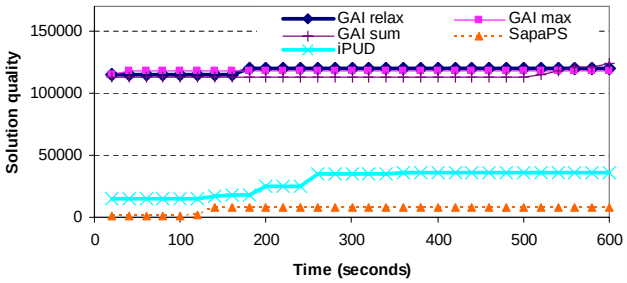


Figure 3: Zenotravel, Problem 13 anytime behavior for each of the planners.

an optimal policy. However, past experiments have shown this approach fails to scale well even when solving problems without utility dependencies on state-of-the-art MDP solvers [Do and Kambhampati, 2004].

While we are not aware of work on using an IP encoding in combination with a greedy search for a planning heuristic as in this paper, there has been work on using IP encoding to handle a subset of planning constraints involving continuous resource or temporal variables [Wolfman and Weld, 1999; Long and Fox, 2003b].

## 7 Conclusion

In this paper, we showed how the GAI framework can be used to represent utility dependencies in planning problems and introduced both a heuristic search framework and an IP encoding for solving partial satisfaction problems with utility dependencies ($PSP^{\mathcal{UD}}$ ). Though we saw variation in the performance of the planners, *SPUDS* using the $h^{GAI}_{relax}$ heuristic typically did better than the other methods shown. Performance of the $h^{GAI}_{relax}$ heuristic indicates that high quality relaxed plan computation can involve significant computation effort, but the extra guidance received pays off more than when lower-quality methods are used. We also see that while IP approaches solve these types of problems, they may have difficulty scaling in problem size. They also typically find solutions at a slower rate than heuristic-based approaches as observed from *iPUD*'s performance over time.

We plan to extend this work further to combine both the quantitative preferences of PSP with qualitative models as handled in planners like [Brafman and Chernyavsky, 2005]. We are also considering extensions to our model that handle preferences like those seen in the most recent International Planning Competition (IPC5) [Gerevini *et al.*, 2006]. To improve the performance, we plan on investigating more effective admissible heuristics that more aggressively take into account negative interactions, such as residual cost as described in AltWlt [Nigenda and Kambhampati, 2005] to improve the heuristic quality.

**Acknowledgement:** We want to thank David Smith, Wheeler Ruml, Ronen Brafman, William Cushing, and the IJCAI reviewers for helpful comments on this paper. Kambhampati's research is supported in part by the NSF grant IIS–308139, the ONR grant N000140610058 and by a Lockheed Martin subcontract TT0687680 to ASU as part of the DARPA Integrated Learning program.

## References

[Bacchus and Grove, 1995] F. Bacchus and A. Grove. Graphical models for preference and utility. In *Proc. of UAI-95*, 1995.

[Blum and Furst, 1997] A. Blum and M. Furst. Planning through planning graph analysis. *Artificial Intelligence Journal*, 90:281–330, 1997.

[Bonet *et al.*, 1997] B. Bonet, Loerincs G., and H. Geffner. A robust and fast action selection mechanism for planning. In *Proceedings of AAAI-97*, 1997.

[Boutilier *et al.*, 1999] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research (JAIR)*, 11:1–91, 1999.

[Boutilier *et al.*, 2001] C. Boutilier, R. Brafman, H. Hoos, , and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *Proc. of UAI-2001*, 2001.

[Brafman and Chernyavsky, 2005] R.I. Brafman and Y. Chernyavsky. Planning with goal preferences and constraints. In *Proceeding of ICAPS-05*, 2005.

[Bylander, 1994] T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence Journal*, 69:165–204, 1994.

[Do and Kambhampati, 2004] M.B. Do and S. Kambhampati. Partial satisfaction (over-subscription) planning as heuristic search. In *Proceedings of KBCS-04*, 2004.

[Gerevini *et al.*, 2006] A. Gerevini, B. Bonet, and R. Givan. Fifth international planning competition. In *IPC06 Booklet*, 2006.

[Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.

[Long and Fox, 2003b] D. Long and M. Fox. Exploiting a graphplan framework in temporal planning. In *Proceedings of ICAPS-2003*, 2003.

[Nigenda and Kambhampati, 2005] R.S. Nigenda and S. Kambhampati. Planning graph heuristics for selecting objectives in over-subscription planning problems. In *Proceedings of ICAPS-05*, 2005.

[Smith, 2004] D.E. Smith. Choosing objectives in over-subscription planning. In *Proceedings of ICAPS-04*, 2004.

[van den Briel *et al.*, 2004] M.H.L. van den Briel, R.S. Nigenda, M.B. Do, and Subbarao Kambhampati. Effective approaches for partial satisfaction (over-subscription) planning. In *Proceedings of AAAI-04*, 2004.

[van den Briel *et al.*, 2005] M.H.L. van den Briel, T. Vossen, and S. Kambhampati. Reviving integer programming approaches for ai planning: A branch-and-cut framework. In *Proceedings of ICAPS-05*, 2005.

[Wolfman and Weld, 1999] S. Wolfman and D.S. Weld. The LPSAT engine and its applicationto resource planning. In *Proceedings of IJCAI-99*, pages 310–317, 1999.