Generating Plans for Qualitative Goal Preferences

Cory Siler and **Michael T. Cox**
Wright State Research Institute
Beavercreek, OH 45431

Abstract

When an autonomous agent can achieve only a subset of its goals, it must consider how pursuing one goal affects another, as well as which goals are most valuable in themselves. If the relative values of goals are quantifiable, goal selection might be done within a standard oversubscription planner. However, if the agent bases its priorities on preferences expressed by a human, it is likely to have only incomplete, qualitative knowledge of these preferences. We consider goal selection via planning when goal preferences are given as a partial order. We show how to use answer set programming to generate a set of optimal plans, where a plan's optimality is determined by the goals it achieves. Our experiments on a benchmark planning domain demonstrate how plan generation and goal selection are constrained as additional preference information is added.

1 Introduction

Problems in complex environments require the achievement of many goals for a satisfactory solution. Classical planning technology attempts to create a single plan that achieves all current goals no matter how many and how diverse they may be. Not all goals are equally valuable or easy to achieve, and some goals belong together; whereas others require distinctly different plans. An intelligent agent will reason about the characteristics of its goals, their desirability, and the relationships between them to decide which ones to achieve first. Often such an approach will use a serial pipeline in which the agent first selects a particular set of goals from its agenda, creates a plan for them, and then executes the plan to achieve the goals. Additional refinement typically occurs only in response to exogenous events during plan execution. Unfortunately, goal selection and planning are interdependent, and simple pipeline mechanisms can fail when confronted by negative interacting conditions.

Consider an autonomous vehicle such as a planetary rover collecting soil samples or a drone performing search tasks. These kinds of agents must deal with *orienteering problems* [Smith, 2004], where goals are associated with physical sites and not enough fuel exists to visit all sites. Traveling to the site of a goal not only consumes resources (fuel) but also changes the resource costs of achieving other goals (by changing the vehicle's distance to other sites). Informative trade-offs between goals exist that would benefit goal selection, but many are not known until planning time.

Greedy goal selection based on a heuristic cost-benefit estimate is effective in some domains [Kondrakunta, 2017], but relies on goals being relatively independent of one another. Alternatively, the brute-force generation of plans for all possible subsets of an existing goal agenda to evaluate which one will be most valuable to achieve can be prohibitively expensive computationally. Domains with complex trade-offs between goals and plans demand a tighter integration of goal reasoning and planning than a simple plan-first or selection-first policy.

Planning incorporating goal selection is a well-established topic within the planning community, and Paredes and Ruml [2017] go so far as to propose handling *all* of an agent's goal reasoning processes implicitly within online planning. However, this assumes that the agent has predetermined criteria about *which* goals to favor. If the goal priorities are dynamic and/or exogenous, an explicit goal reasoning module [Hawes, 2011; Roberts *et al.*, 2018; Vattam *et al.*, 2013] should determine the priorities before planning [Gunderson, 2000].

This is particularly true if the goal priorities are based upon the high-level preferences of a human as represented by a military commander's intent statement or an industry leader's corporate vision. In a natural interactions, a human is likely to give abstract directives that may not provide exact guidance for every situation as is the case with a fully-specified utility function. Thus, for realistic mixed-initiative interactions between intelligent agents and humans, an effective agent must be able to represent and plan for partial, qualitative preferences over the goal sets to pursue. This paper proposes qualitative goal preferences as a means of efficiently constraining the generation of plans used for goal selection.

It is possible to have *incomparability* in such preference representations, where there are pairs of goal sets for which no preference relationship between them holds in the model. The agent faces ambiguity when there are multiple "best" achievable goal sets due to incomparability: If the incomparability in the model represents an actual lack of preference by the human, the agent can choose one of the goal sets arbitrarily and be confident in its true optimality. But it is possible

that the human actually *does* have a preference between the goal sets, and the incomparability in the model simply represents the agent's lack of *knowledge* about the preference; some choices that are optimal with respect to the model may turn out to be undesirable for the human. When this ambiguity occurs, the agent should be able to enumerate the possibly-best goal sets to the human in order to seek clarification.

In this paper, we propose and implement a planning paradigm that is well-suited to goal reasoning agents based on the concerns discussed above. Given a planning instance and qualitative preferences over goals that induce preferences over plans, our problem is to find a maximal set of optimal plans where each achieves a different goal set. In Section 2, we survey existing forms of preference-based planning and highlight how they differ from ours. In Section 3, we review relevant planning formalisms and the semantics we will use to represent goal and plan preferences. In Section 4, we show how answer set programming can be used to find sets of preferred plans. In Section 5, we use our answer-set-programming-based implementation to show how preferences affect the plans under consideration in a transportation planning domain. In Section 6, we summarize our findings so far and consider directions for future work.

2 Related Work

Preference handling has attracted considerable interest in AI planning [Baier and McIlraith, 2008]; in fact, PDDL3 [Gerevini and Long, 2006] includes features for specifying preferences over "soft goals" as well as "soft constraints" on the plan trajectory. Exemplary problems include *net-benefit partial satisfaction planning* [van den Briel *et al.*, 2004], where the objective is to maximize the plan's total utility (of goals achieved) minus cost (of actions taken); and *cost-bounded oversubscription planning* [Smith, 2004], where the objective is to maximize utility without exceeding a given cost. In general, these use additive utility and cost functions.

In contrast, *Pareto-based multiobjective planning* [Khouadjia *et al.*, 2013] does not require the relative importance of different types of costs and utilities to be specified; instead of a single objective function, it measures plan quality with a *vector* of objective values. One plan *Pareto dominates* (in other words, is "strictly better" than) another if it scores better on some objectives and no worse on the rest. These planners search for the *Pareto front*, consisting of plans that are not Pareto dominated (see Figure 1); any given preference function over the objectives will have an optimal plan in the Pareto front. Nguyen *et al.* [2012] present a multiobjective approach that is conceptually similar to ours: Given partially-known user preferences (in their case, the preferences are described quantitatively), they search for a diverse set of Pareto-front plans that are also possibly optimal for the user. However, these works assume there exists a fixed goal that the plan must achieve the "objectives" are plan-trajectory properties like length and execution cost rather than goals achieved.

The idea of partial *goal* preferences is much less explored in the literature. Brafman and Chernyavsky [2005] and Feldmann *et al.* [2006] introduce algorithms for planning with

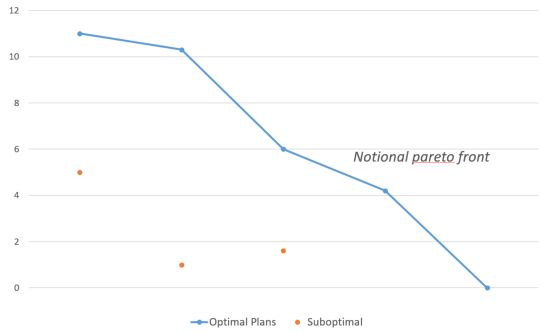

Figure 1: Illustration of Pareto dominance with two objectives (one score on each axis).

rich qualitative preference models that induce partial orders over goal sets. Given a problem instance, these algorithms find an individual plan achieving a goal set that is optimal with respect to the partial order. (Other optimal goal sets, incomparable to the one achieved by the found plan, may also exist.)

Specific research in the goal reasoning literature also applies here. Floyd *et al.* [2018] have recently investigated planning with goal preferences in a *goal-driven autonomy* [Cox, 2013; Klenk *et al.*, 2013; Molineaux *et al.*, 2010] context. They consider an agent taking initial goals and a partial preference specification from a human commander. The agent uses these preferences to assign utility values to the initial goals; when it encounters new goals later, it generates utilities for these goals as well, making predictions from the initial preference specification without additional human input.

In summary, prior work in planning with preferences has considered combinations of these ideas, but not (to our knowledge) all three together:

- Plan preferences may be known only partially and qualitatively.
- Preferences are based on the set of goals achieved by the plans.
- The planner returns a set of plans that differ with respect to the preferences.

3 Preliminaries

A classical planning domain is defined [Ghallab *et al.*, 2004] as a finite state-transition system in which each state $s \in S$ is a finite set of ground atoms. A planning operator is a triple $o = (head(o), pre(o), \mathit{eff}(o))$, where $pre(o)$ and $\mathit{eff}(o)$ are preconditions and effects. Each action, $\alpha \in A$ is a ground instance of some operator o. An action is executable in a state s if $s \vDash pre(\alpha)$.

For a classical planning domain, the state-transition system is a tuple $\Sigma = (S, A, \gamma)$, where S is the set of all states, and A is the set of all actions as above. In addition, γ is a state transition function $\gamma : S \times A \rightarrow S$ that returns a resulting state of an executed action given a current state, i.e., $\gamma(s, \alpha) \rightarrow s'$.

A classical planning problem is a triple $P = (\Sigma, s_0, G)$, where Σ is a state transition system, s_0 is the initial state, and

each goal $g \in G$ is a first-order formula. A state s satisfies a goal g if $s \models g$. A plan $\pi \in \Pi$ consists of sequence of plan steps $\langle \alpha_1, \alpha_2, \ldots \alpha_n \rangle$ that incrementally changes the world, starting from the initial state s_0; it represents a solution to P if it results in a final state s_n that satisfies all provided goals. That is, it is a solution if $\gamma(\ldots \gamma(\gamma(s_0, \alpha_1), \alpha_2) \ldots, \alpha_n) \models \bigwedge_{g \in G} g$.

A *preference-based planning* instance is defined [Baier and McIlraith, 2008] as a pair $I = (P, \succeq_\Pi)$, where P is a classical planning problem and $\succeq_\Pi$ is a reflexive, transitive relation in $\Pi \times \Pi$ giving a partial preorder over the plans $\pi \in \Pi$. If $\pi \succeq_\Pi \pi'$ but $\pi' \not\succeq_\Pi \pi$, we write $\pi \succ_\Pi \pi'$ and say that π is *preferred to* π' with respect to $\succeq_\Pi$. We say that a plan π is *optimal* with respect to $\succeq_\Pi$ if there is no plan π' such that $\pi' \succ_\Pi \pi$.

3.1 Qualitative Preferences

In this paper, we are interested in preference-based planning where the corresponding planning instance I includes an unsatisfiable problem P — i.e., no plan achieves P's entire goal set G — and the plan preferences are based on the subset of G achieved by the plans. While this concept is not limited to a specific preference model (and there exists a rich body of research on set preferences, e.g., Brewka *et al.* [2010]), in this paper we will restrict our attention to a formalism similar to that of Di Rosa *et al.* [2010]. In their semantics, a preference ordering over Boolean literals induces a preference ordering over assignments to Boolean formulas, favoring assignments where more-preferred literals are true; in our context, a preference ordering over goals induces a preference ordering over plans, favoring plans where more-preferred goals are achieved.

Our input language consists of a partial order $\succ_G$ giving preferences over goals $g \in G$. From this, we define the plan preferences $\succeq_\Pi$ as follows: Given plans π, π', let $Unique_\pi$ be the set of goals achieved by π but not by π', and let $Unique_{\pi'}$ be the set of goals achieved by π' but not by π. It holds that both $\pi \succeq_\Pi \pi'$ and $\pi' \succeq_\Pi \pi$ iff $Unique_\pi = Unique'_\pi = \emptyset$. It holds that $\pi \succ_\Pi \pi'$ iff, for each goal $g' \in Unique_{\pi'}$, there is a goal $g \in Unique_\pi$ such that $g \succ_G g'$.

In other words, one plan is preferred to another if either of the following holds – the first achieves all of the goals achieved by the second, plus some; or each goal uniquely achieved by the second is less preferred than some goal uniquely achieved by the first. Two plans are equally-preferred if they achieve the same goals. Two plans are incomparable if none of the above conditions hold.

Example 1 *Suppose we have plans for the following goal sets (refer to Figure 2):*

- π_1 *achieves* $\{g_1\}$
- π_2 *achieves* $\{g_1, g_2\}$
- π_3 *achieves* $\{g_2, g_3, g_4\}$

Firstly, note that π_1 is suboptimal for any goal preferences; it achieves a strict subset of the goals achieved by π_2. The remaining plans are incomparable if we do not have goal preferences relating g_1 to g_3 and g_4.

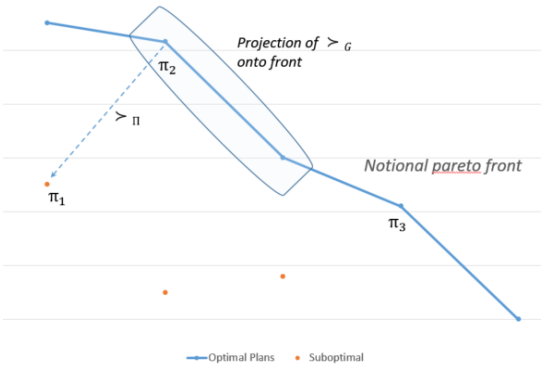

Figure 2: Illustration of Pareto dominance with two objectives (one score on each axis). Preferences constrain which plans in the Pareto front are considered optimal.

Now suppose we have the following goal preferences: $g_1 \succ_G g_2$; $g_1 \succ_G g_3$; and $g_3 \succ_G g_4$ (plus the transitive closure of these statements). With our preference semantics, $\pi_2 \succ_\Pi \pi_3$, since all of the goals unique to π_3 (namely, g_3 and g_4) are dominated by a goal unique to π_2 (namely, g_1).

3.2 Multiobjective Planning

We can generalize our preference semantics to multiobjective planning where we have numeric goals for which a plan can achieve varying scores, rather than propositional goals that a plan simply achieves or fails to achieve.[1] Instead of a set of propositional goals, assume that G is a set of objective functions that yield scores for numeric goals. For plans π, π', let $Better_\pi$ be the set of objective functions on which π scores higher, and let $Better_{\pi'}$ be the set of objective functions on which π' scores higher. Then we define $\succeq_\Pi$ with respect to $Better_\pi$ and $Better_{\pi'}$ in the same way that we defined it with respect to $Unique_\pi$ and $Unique_{\pi'}$ above.

Note that plans optimal with respect to $\succeq_\Pi$ are also Pareto optimal, but the reverse is not necessarily true. When all goals/objectives are incomparable with respect to $\succ_G$, the set of optimal plans for $\succeq_\Pi$ is equal to the Pareto front; but as the goal/objective preferences become increasingly strict, their "projection" onto the Pareto front becomes increasingly narrow, as illustrated in Figure 2. When $\succ_G$ is maximally strict — i.e., a total order — $\succeq_\Pi$ orders the plans lexicographically, and there is only one optimal goal set [2] or objective vector.

4 Answer Set Planning with Goal Preferences

As described in Section 2, qualitative preference-based planners exist that find a plan for a single optimal goal set, but

[1] For instance, we may want the agent to collect as many of a certain object type as possible; we could have it maximize a single numeric goal for collecting more objects, rather than keeping separate propositional goals for each object.

[2] We will use "optimal goal set" as shorthand for "goal set achievable by an optimal plan with respect to $\succeq_\Pi$"; we refer to goal sets that are optimal *among goal sets for which satisfying plans exist*, rather than optimal among all goal sets including the unsatisfiable ones.

approaches to finding plans for multiple optimal goal sets are lacking. Rather than building a planner from scratch, we have approached the planning problem by reducing it to an answer set program and leveraging existing answer set programming tools designed for preference-handling.

Before describing our implementation, we will briefly summarize the syntax and semantics of answer set programming itself. We use the syntactic style of the input language to the answer set solver Clingo [Gebser *et al.*, 2014], and we borrow some of our examples from Lifschitz [2008].

4.1 Overview of Answer Set Programming

Answer set programming [Brewka *et al.*, 2011] is a declarative programming paradigm that uses Prolog-like rules but is geared toward generating solutions to constraint problems rather than proving answers to queries. The following is an example of a simple answer set program:

```
p :- q.
q :- not r.
```

This program declares that the atom `p` holds whenever `q` holds, and `q` holds whenever `r` does not. It has a single answer set: {`p`,`q`}.

Note that the `not` operator denotes *default negation* rather than classical negation; for `not r` to hold, it is sufficient that `r` has not been derived. To further illustrate this, the following program has two answer sets, {`q`} and {`r`}:

```
q :- not r.
r :- not q.
```

Answer set programming also allows for *choice rules* that generate combinations of their elements, as in the following:

```
{ s ; t }.
```

This rule allows for answer sets with arbitrary inclusions from among `s` and `t`. The answer sets of the above program are {}, {`s`}, {`t`}, and {`s`,`t`}.

It is also possible to impose cardinality bounds on a choice rule, restricting the number of elements that can be generated together:

```
1 { s ; t ; u ; v } 3.
```

In this case, between 1 and 3 atom from among `s`, `t`, `u`, and `v`, may be generated in an answer set, but not 0 or 4. We can also limit what can be generated using a rule without a head, also known as a *constraint*:

```
:- u, v.
```

A rule with no head prevents answer sets where the body holds; in this case, an answer set cannot contain both `u` and `v`.

One more relevant feature is the use of predicates over variables (where a variable terms begins with an uppercase letter, as opposed to the lowercase of predicates and atoms):

```
p(a).
p(b).
q(X) :- p(X).
```

When the last rule is grounded, the variable `X` is replaced with atomic terms, producing the rules `q(a) :- p(a).` and `q(b) :- p(b).`; the program has the single answer set {`p(a)`,`p(b)`,`q(a)`,`q(b)`}.

4.2 Planning in Answer Set Programming

Answer set programming has been applied to a wide variety of hard combinatorial search problems, planning being among them.[3] Here we will summarize our own implementation of STRIPS-style planning rules.

An answer set to our planning program will consist of a the plan itself — represented by a group of atoms of the form `plan(A,T)`, giving the choice of an action `A` for each timestep `T` — as well as the auxiliary atoms used to help generate and constrain of plan. These include atoms of the form `action(A,T)`, indicating that action `A` is available (i.e., has its preconditions met) at time `T`, and `holds(P,T)`, indicating that some domain fact `P` is true at time `T`.

The following choice rules generate plan steps. The rules select some action as part of the plan at each timestep from those that are available, up until some stopping point for the plan.

```
1{ plan(A,T) : action(A,T) }1 :-
    time(T), not done(T).
1{ done(T) : time(T) }1.
```

To illustrate how we define the actions themselves, consider the following STRIPS operator from a logistics domain, which defines the action of loading a package onto a truck:

```
(:action load
:parameters(
    ?Pa - package
    ?Tr - truck
    ?Lo - location)
:precondition (and
    (at ?Tr ?Lo)
    (at ?Pa ?Lo))
:effect (and
    (not (at ?Pa ?Lo))
    (in ?Pa ?Tr))
)
```

A corresponding definition for our answer set program is (with lines beginning in "`%`" being comments):

```
action(load(Pa,Tr,Lo),T) :-
    % Parameter types
    package(Pa), truck(Tr), location(Lo),
    % Preconditions
    holds(at(Tr,Lo),T-1),
    holds(at(Pa,Lo),T-1).
% Effects
add(in(Pa,Tr),T) :-
    plan(load(Pa,Tr,Lo),T).
delete(at(Pa,Lo),T) :-
    plan(load(Pa,Tr,Lo),T).
```

We express positive and negative action effects in the style of *add* and *delete lists* respectively; then the following rules declare that a domain fact becomes true in the at a given time if it appears in a chosen action's add list, and persists over time unless it is revoked in a chosen action's delete list:

[3] A blocksworld planning program along with several non-planning examples are executable in the browser-based demo of Clingo at `https://potassco.org/clingo/run/`.

```
% Addition of new domain facts
holds(P,T) :-
    add(P,T), time(T).
% Persistence of domain facts
holds(P,T) :-
    holds(P,T-1), time(T),
    not delete(P,T).
```

A problem instance provides initial domain facts with `init(P)`, which are set to true at the initial timestep with the rule:

```
holds(P,0) :- init(P).
```

An instance also includes goals with `goal(P)`, whose achievement at the end of the plan is checked using:

```
achieved(P) :- goal(P),
    holds(P,T), done(T).
```

The following rule would require generation only of plans that achieve all goals, but we exclude it because we are interested in cases where not all goals can be achieved together:

```
:- goal(P), not achieved(P).
```

4.3 Optimizing for Goal Preferences

A particular appeal of answer set programming for our planning paradigm is the existing support for preference-based optimization of answer sets [Brewka *et al.*, 2003]. In particular, the Asprin framework [Brewka *et al.*, 2015] extends the Clingo solver from generating some or all of a program's answer sets to generating some or all of a program's *optimal* answer sets with respect to a given preference model. Asprin finds all optimal answer sets in an efficient manner by iteratively improving upon suboptimal sets, adding constraints to the program that forbid the generation of a next answer set that is worse than the previous one. Furthermore, it makes use of Clingo's *multi-shot solving* capabilities [Gebser *et al.*, 2014] where calls to the solver on changing programs will reuse data from the previous calls instead of starting the solving process from scratch.

Asprin has several built-in formalisms for specifying preferences over answer sets based on the atoms they contain — in our case, specifying preferences over generated plans based on the goals they achieve as indicated by the atoms `achieved(P)`. Of particular interest is Asprin's `poset` preference type, which implements Di Rosa *et al.*'s qualitative preferences [2010] on which we model our goal preferences as defined in Section 3.1.

When we have preferences $\succ_G$ over goals, we add to our planning instance a statement of the form `prefer(p1,p2).` for each comparison $p_1 \succ_G p_2$. We instruct Asprin to optimize for these preferences with the preference program:

```
#preference(goalprefs,poset){
    achieved(P1) >> achieved(P2) :
        prefer(P1,P2);
    achieved(P) : goal(P)
}.
#optimize(goalprefs).
```

The `#preference` directive creates a preference relation named `goalprefs` using the `poset` semantics. Its definition states that one goal is preferred to another if the instance pairs them with the `prefer` predicate; and that, all else being the same, achieving a goal is preferred to not achieving it. The `#optimize` directive instructs Asprin to use this preference relation for optimization.

We can extend this to numeric goals if the scores come from a bounded range of integers. Assume the instance declares numeric goals `numgoal(N)` and defines a predicate `score(N,S)` giving scores `S` for numeric goals `N`. Then the following program implements the numeric-goal version of our qualitative preferences:

```
score(N,0..S) :- score(N,S).
#preference(numgoalprefs,poset){
    score(N1,S1) >> score(N2,S2) :
        prefer(N1,N2),
        numgoal(N1), numgoal(N2),
        score(_,S1), score(_,S2);
    score(N,S) : numgoal(N), score(_,S)
}.
#optimize(numgoalprefs).
```

5 Experiments

We will now describe how we used our implementation to investigate the relationship between the relationship between strictness of goal preferences and the number of optimal achievable goal sets. [4]

In theory, preference strictness and the number of optimal plans or goal sets are negatively correlated: Let $\succ_G$ and $\succ'_G$ be goal preference orderings that induce plan preferences $\succeq_\Pi$ and $\succeq'_\Pi$ respectively. If the comparisons in $\succ_G$ are a superset of the comparisons in $\succ'_G$, then the optimal plans with respect to $\succeq_\Pi$ will be a subset of the optimal plans with respect to $\succeq'_\Pi$. As comparisons are added, the set of optimal plans converges toward the set only of those plans that achieve one particular goal set.

The empirical question is *how quickly* the addition of goal preferences reduces the space of optimal achievable goal sets. Ideally, we want an agent to be able to greatly reduce the number of goal sets under consideration with only a handful of preference statements given by a human, thus reducing both the communication burden on the human to give thorough preferences and the computational burden on the agent to consider a wide space of plans.

The pattern of convergence will vary across planning domains and instances, so we cannot characterize it universally. We instead focus on a domain representative of the vehicle-related problems that we discussed in Section 1 to motivate combined goal selection and planning — namely, Steinmetz and Hoffmann's NoMystery benchmark [2016] as used in the 2016 Unsolvability International Planning Competition.[5]

[4] Note that goal preferences do not affect the number of *overall* achievable goal sets; the only change is in which of those goal sets are considered optimal.

[5] PDDL domain files and generators for competition benchmarks are available at `https://bitbucket.org/planning-researchers/unsolve-ipc-2016`.

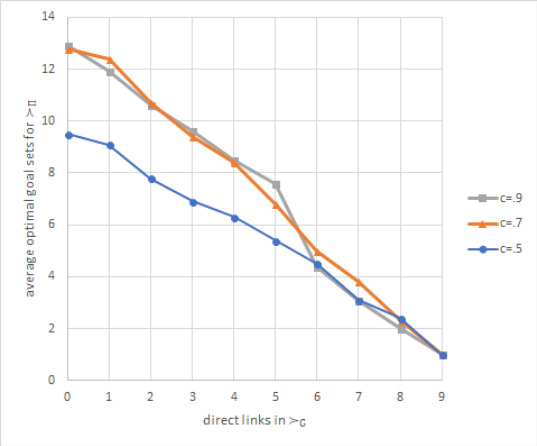

Figure 3: Unique optimal goal sets among NoMystery instances. Preference strictness increases from left to right.

NoMystery is a transportation domain with *trucks*, *locations*, and *packages*. A truck can *drive* between locations and *load* and *unload* packages at those locations. Each package has an associated goal of delivering it to a particular location. Trucks expend some of their limited *fuel* supply when they drive. A distinguishing feature of NoMystery is that the generator takes a parameter c that scales the fuel availability; $c = 1$ means the trucks have exactly the minimum amount of fuel to deliver all packages, $c = 2$ means there is twice that amount of fuel, and so on. We used $c < 1$, preventing any given plan from delivering every package.

We used the following parameters to generate instances: 1 truck, 10 locations, and 10 packages to deliver, with equal numbers of nodes and edges in the location graph (where nodes represent locations and edges indicate that the truck can go between the locations in a single drive action) and equal fuel costs for each travel action. We generated 10 distinct instances, with versions of each instance for fuel constrainedness $c = .5$, $c = .7$, and $c = .9$. We processed the PDDL specifications of the instances to create equivalent versions for the answer set program described in Section 4.2.

In addition, for each distinct instance we generated preferences over the package delivery goals at varying levels of strictness as follows: Let $p_1, p_2, \cdots, p_{10}$ be the package delivery goals. For maximum strictness, we imposed a total order by generating comparisons $p_1 \succ_G p_2, p_2 \succ_G p_3, \cdots, p_9 \succ_G p_{10}$. Then, to generate a less-strict preference ordering, we deleted one of those comparisons at random, in essence breaking the "chain" of goal preferences into two pieces. We repeated this process until all comparisons were deleted.

We ran Clingo with Asprin on each instance with each of its fuel constraints and goal preference orderings; we had Asprin find all optimal goal sets. Figure 3 shows the relationship between number of direct comparisons in the goal preference ordering (i.e., not counting the transitive closure) and the average number of optimal goal sets with respect to those preferences.

The number of optimal goal sets in our experiments declined in a roughly linear fashion as the number of direct comparisons increased. We also observe that the $c = .5$ versions had fewer optimal goal sets on average than the versions with higher values of c. This is in part because with less initial fuel, there are more packages that the truck cannot deliver in *any* plan, whereas with additional fuel they can be delivered at the expense of other goals.

Overall, these results suggest that the Pareto front of goal sets is largest when the amount of available resources is plenty for individual goals while still being inadequate for achieving all goals together. The impact of qualitative preferences on eliminating Pareto-front goal sets scales with the size of the Pareto front itself. Still, the convergence of the number of optimal goal sets may be too slow in situations where a human provides only a very sparse preference specification; thus, it would be valuable for an agent to guess some of the human's unstated preferences [Floyd *et al.*, 2018] to help further constrain the goal sets under consideration.

6 Conclusions

Goal reasoning agents interacting with humans in realistic settings will often need to select goals based on loosely-specified human preferences; when the specification is too vague for a particular problem, the agent should consider solutions based on differing interpretations of the preferences. In this paper, we have explored a planning paradigm based on this idea and shown how a preference-handling framework for answer set programming can be leveraged to generate sets of preferred plans. We do not claim to have designed any sort of state-of-the-art planner; rather, our contribution is to highlight a way in which goal reasoning, planning, and preference-based knowledge representation can be bridged to help create more-flexible autonomous agents.

Direct algorithms for our planning paradigm, as opposed to reductions like the one presented here, are an open area of investigation. A specialized planner, using a qualitative-preference-handling extension of PDDL like the one introduced by by Feldmann *et al.* [2006], may be worthwhile; recent empirical evidence by Jiang *et al.* [2018] suggests that PDDL-based planners scale better than answer set planners when longer plans are needed, because of the grounding bottleneck in answer set programming.

However, Jiang *et al.*'s experiments also suggest that answer set planners scale better than PDDL-based planners to domains that require *more-complex reasoning* — the same sorts of domains for which we would build goal-reasoning agents to begin with. Answer set planning is also attractive because of the ease of specifying new extensions to existing planning languages, the ability to define a wide variety of plan and goal preferences beyond the ones considered here, and efficient plan repair through multi-shot solving when new information is discovered that invalidates an existing plan.

The potential applications of answer set programming to goal reasoning extend beyond planning and goal selection. Consider the process of discrepancy detection, explanation, and goal generation in goal-driven autonomy. In place of our `holds(P,T)` predicate, a goal-driven autonomous agent might produce a series of `expected(P,T)` facts during

planning, and add `observed(P,T)` facts as it perceives the world; a fact that is `expected` or `observed` but not both is a possible discrepancy. A choice rule can be used to generate sets of facts that might explain a given discrepancy, yielding different answer sets for the possible worlds entailed by different explanations. A preference-handling framework like Asprin can be used to evaluate not only which solutions to problems are more desirable, but also which possible worlds are considered more plausible; the agent can then focus its goal generation on the explanations in which it has the most credence.

Acknowledgments

This material is based on research sponsored by the Air Force Research Laboratory, under agreement number FA8650-16-C-6763. This research was also supported by AFOSR grant FA2386-17-1-4063. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

References

[Baier and McIlraith, 2008] Jorge A Baier and Sheila A McIlraith. Planning with preferences. *AI Magazine*, 29(4):25–37, 2008.

[Brafman and Chernyavsky, 2005] Ronen I Brafman and Yuri Chernyavsky. Planning with goal preferences and constraints. In *International Conference on Automated Planning and Scheduling*, pages 182–191, 2005.

[Brewka *et al.*, 2003] Gerhard Brewka, Ilkka Niemela, and Miroslaw Truszczynski. Answer set optimization. In *International Joint Conference on Artificial Intelligence*, pages 867–872. Morgan Kaufmann Publishers Inc., 2003.

[Brewka *et al.*, 2010] Gerhard Brewka, Mirosław Truszczyński, and Stefan Woltran. Representing preferences among sets. In *AAAI Conference on Artificial Intelligence*, pages 273–278. AAAI Press, 2010.

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[Brewka *et al.*, 2015] Gerhard Brewka, James P Delgrande, Javier Romero, and Torsten Schaub. asprin: Customizing answer set preferences without a headache. In *AAAI Conference on Artificial Intelligence*, pages 1467–1474, 2015.

[Cox, 2013] Michael T Cox. Question-based problem recognition and goal-driven autonomy. In *Goal Reasoning: Papers from the ACS workshop*, pages 10–25. University of Maryland, Department of Computer Science, 2013.

[Di Rosa *et al.*, 2010] Emanuele Di Rosa, Enrico Giunchiglia, and Marco Maratea. Solving satisfiability problems with preferences. *Constraints*, 15(4):485–515, 2010.

[Feldmann *et al.*, 2006] Robert Feldmann, Gerhard Brewka, and Sandro Wenzel. Planning with prioritized goals. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 503–514, 2006.

[Floyd *et al.*, 2018] Michael W Floyd, Mark Roberts, and David Aha. Hybrid goal selection and planning in a goal reasoning agent using partially specified preferences. In *Florida Artificial Intelligence Research Society Conference*, 2018. Poster abstract.

[Gebser *et al.*, 2014] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo= ASP+ control: Preliminary report. *arXiv preprint arXiv:1405.3694*, 2014.

[Gerevini and Long, 2006] Alfonso Gerevini and Derek Long. Preferences and soft constraints in PDDL3. In *ICAPS Workshop on Planning with Preferences and Soft Constraints*, pages 46–53, 2006.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Elsevier, 2004.

[Gunderson, 2000] James P Gunderson. Adaptive goal prioritization by agents in dynamic environments. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 1944–1948. IEEE, 2000.

[Hawes, 2011] Nick Hawes. A survey of motivation frameworks for intelligent systems. *Artificial Intelligence*, 175(5-6):1020–1036, 2011.

[Jiang *et al.*, 2018] Yuqian Jiang, Shiqi Zhang, Piyush Khandelwal, and Peter Stone. An empirical comparison of PDDL-based and ASP-based task planners. 2018.

[Khouadjia *et al.*, 2013] Mostepha Redouane Khouadjia, Marc Schoenauer, Vincent Vidal, Johann Dréo, and Pierre Savéant. Pareto-based multiobjective AI planning. In *International Joint Conference on Artificial Intelligence*, pages 2321–2327. AAAI Press, 2013.

[Klenk *et al.*, 2013] Matthew Klenk, Matt Molineaux, and David W Aha. Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, 29(2):187–206, 2013.

[Kondrakunta, 2017] Sravya Kondrakunta. Implementation and evaluation of goal selection in a cognitive architecture. Master's thesis, Wright State University, 2017.

[Lifschitz, 2008] Vladimir Lifschitz. What is answer set programming? In *AAAI Conference on Artificial Intelligence*, volume 8, pages 1594–1597. AAAI Press, 2008.

[Molineaux *et al.*, 2010] Matt Molineaux, Matthew Klenk, and David W Aha. Goal-driven autonomy in a Navy strategy simulation. In *AAAI Conference on Artificial Intelligence*, pages 1548–1554. AAAI Press, 2010.

[Nguyen *et al.*, 2012] Tuan Anh Nguyen, Minh Do, Alfonso Emilio Gerevini, Ivan Serina, Biplav Srivastava, and Subbarao Kambhampati. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, 190:1–31, 2012.

[Paredes and Ruml, 2017] Alison Paredes and Wheeler Ruml. Goal reasoning as multilevel planning. *ICAPS-2017 Workshop on Integrated Execution of Planning and Acting*, page 36, 2017.

[Roberts *et al.*, 2018] Mark Roberts, Daniel Borrajo, Michael T Cox, and Neil Yorke-Smith, editors. *AI Communications: Special issue on goal reasoning*. IOS Press, 2018.

[Smith, 2004] David E Smith. Choosing objectives in over-subscription planning. In *International Conference on Automated Planning and Scheduling*, pages 393–401. AAAI Press, 2004.

[Steinmetz and Hoffmann, 2016] Marcel Steinmetz and Jörg Hoffmann. Towards clause-learning state space search: Learning to recognize dead-ends. In *AAAI Conference on Artificial Intelligence*, pages 760–768. AAAI Press, 2016.

[van den Briel *et al.*, 2004] Menkes van den Briel, Romeo Sanchez, Minh B Do, and Subbarao Kambhampati. Effective approaches for partial satisfaction (over-subscription) planning. In *National Conference on Artificial Intelligence*, pages 562–569. AAAI Press, 2004.

[Vattam *et al.*, 2013] Swaroop Vattam, Matthew Klenk, Matthew Molineaux, and David W Aha. Breadth of approaches to goal reasoning: A research survey. In *Goal Reasoning: Papers from the ACS Workshop*, pages 111–126. University of Maryland, Department of Computer Science, 2013.