

Replanning as a Deliberative Re-selection of Objectives

William Cushing and J. Benton and Subbarao Kambhampati

Department of Computer Science and Engineering

Arizona State University

Tempe, AZ 85287-5406

{william.cushing|j.benton|rao}@asu.edu

Abstract

Although expressive conceptual theories of replanning have existed for a long time (c.f. Bratman et. al., 1988), computational approaches to replanning have mostly been restricted to plan stability and efficiency issues. In this paper, we will show how the recent work on over-subscription planning can be adapted to provide a more general computational framework for replanning that is sensitive to both opportunities and commitments. We view replanning as a process of re-selecting objectives. Each of the top level objectives are associated with two quantities—rewards and penalties. Reward is accrued when the agent achieves the objective and penalty is incurred when the objective is not met. Newly arriving goals are modeled with rewards, while commitments made by the existing plan are modeled with penalties. The planner continually improves its current plan in an anytime fashion, while monitoring the current situation to see if its selected objectives are still appropriate. Replanning (i.e. re-initializing the search) occurs whenever the situation deviates so far from the model as to merit selecting a new set of objectives.

Introduction

Planning agents operating in dynamic environments need the ability to replan. While execution failures are thought to be the most common trigger for replanning, equally important triggers are new opportunities—be they newly arriving goals or previously infeasible goals that are unexpectedly within reach. Consider, for example, a robot agent operating in concert with other human agents in a rescue scenario (Kitano and Tadokoro 2001). The robot, on its way to rescue an injured person cannot willy nilly ignore a group of children stuck in a burning home. The job of a replanner is further complicated by the fact that it is not enough to resolve the failures and exploit the opportunities, but the agent must do so while respecting the commitments inherent in the current (partially executed) plan. In the rescue example above, the robot should still make it to the injured person that it was originally trying to rescue. Striking an appropriate balance between respecting commitments and exploiting opportunities is at the very heart of general replanning. Although the existence of this fundamental tradeoff has been understood for at least twenty years (Bratman, Israel, and Pollack 1988; Cushing and Kambhampati 2005), the question of *how* to get a replanning agent to demonstrate desirable behavior in a domain independent fashion remains open.

Most current computational work in replanning centers around two incomplete models: replanning as restarting (planning again) (c.f. (Koenig, Likhachev, and Furcy 2004; Fritz and McIlraith 2007)), and replanning as repair (modifying the existing plan minimally) (c.f. (Fox et al. 2006; van der Krogt and de Weerd 2005)). Both of these approaches resolve execution failures, but fail to adequately balance respecting commitments and exploiting opportunities. Replanning as restarting is popular in the robotics community and focuses mostly on finding and refinding optimal or near-optimal motion plans as quickly as possible. This approach completely ignores commitments to other agents. The view of replanning as repair is popular in the AI community and casts replanning as reusing the immediately prior plan on the altered current and goal states. The real difficulty with this approach is that the connection between similarity of plans and quality of behavior is weak. It is quite possible that a plan that differs from the existing one in just one crucial action may be breaking more commitments than one that differs in multiple minor actions (and the differentiation between crucial and minor actions cannot be made without modeling the commitments).

Part of the problem is that most replanning frameworks are based in classical planning models which fall short in two crucial ways: they are “atemporal” and assume all goals are “hard goals.” It is impossible to model opportunities unless the planner can model and handle “soft goals” and it is impossible to understand the commitments made by the current plan unless the planner has a way of synchronizing with the external world.

In this paper, we assume a planning model that allows metric time as well as soft goals (partial satisfaction). Specifically, our approach is built on the recent advances in partial satisfaction planning (PSP). A PSP problem involves goals with differing rewards (as well as reward dependencies), and actions with differing costs. The objective is to find plans with high net benefit (cumulative reward of goals achieved minus the cumulative cost of actions used). Unlike classical planners, which must work on all objectives, PSP planners must intelligently consider which objectives are worth achieving. We show that this objective selection process can be adapted to provide an opportunity and commitment-sensitive framework for replanning. Our extension of the PSP model involves giving both rewards and

penalties to goals. In our execution model, rewards are used to model goals (including any newly arriving ones), while penalties are used to model commitments made by the current plan. Just as an agent can count on accruing the rewards for the goals it achieves, it should also count on incurring the penalty for the objectives it ignores. During each replanning epoch, the agent is called on to select objectives considering the current state, the partially executed plan, and all the available goals, including any opportunities and commitments (with their rewards and penalties).

An advantage of this model is that it can seamlessly handle multiple aspects of replanning and continual planning. Goals can be “added” by just making their reward positive (or removed by zeroing their rewards). An opportunity is an unexpected, and optional, chance to increase net benefit: either a newly added soft goal, or an old soft goal that has become feasible to achieve (less costly than its reward) due to an unexpected, and fortuitous, change in state. Upon encountering an unexpected state, the monitor considers reselecting objectives based on a holistic estimate of reachability, cost, reward, and penalty; while a previously distant goal might have suddenly become much cheaper, it can still be the case that pursuing it negatively interacts with important commitments. The stability of the current plan is no longer an end unto itself, but rather becomes a byproduct of the tradeoff between the penalties of commitments and the rewards of opportunities.

In the next section, we make explicit the fact that replanning is necessitated as most planning worlds are really projections of multi-agent planning into a single agent world. In the following section, we prepare important definitions and notation. This is followed by the section where we formally state the desired behavior and characteristics of a replanning agent. Briefly, we require that an agent should follow through on its claims (respect commitments) or be able to justify deviation either through improved performance (exploit opportunities) or via (justifiably) blaming the environment (unresolvable failures). Next, we describe a feasible¹ computational model of replanning implementing the developed theory. Finally, we discuss in greater detail some of the connections to prior research as well as identifying future directions before concluding.

Why Replan At All?

A broad class of replanning problems consists of an environment populated with friendly and neutral agents. We develop a replanner targeted at performing well in these and simpler situations. An even broader class of replanning problems permits malicious agents, but the core of research in planning is search in an OR-space, in particular, planning heuristics would seem to be ill-adapted to search in an AND^k – OR-space; and there seems to be little way around a game-theoretic treatment of malicious agents.

¹That is, a system which can be counted on to finish deliberating within a reasonable amount of time for problems of reasonable size; on the order of minutes, hours, or perhaps days, but not on the order of months, years, or decades. . .

The useful property of the targeted situations (with friendly and neutral agents) is that it is not necessary to treat the other agents as opponents in terms of the search; should failures arise it will not be due to maliciousness but to accident. This alone reduces the search from the game-theoretic AND^k – OR-space to a significantly simpler OR^{k+1}-space. Nonetheless, this is still a combinatorial and logistical nightmare; such a planning model asks an agent to plan for everyone it could potentially interact with, and subsequent to that it would have to find a way to convince the other agents to conform exactly to these plans.

This will certainly lead to failures, but the intuition is that overall performance will be far superior, as the agent will in fact be able to complete the search for a plan within available time and memory limitations (**bounded** rationality (Bratman, Israel, and Pollack 1988)). So it will be very important to implement an effective strategy for resolving failures.

Preliminaries

The basic units of a planning language are *actions* and *states*; executing an action in a state yields a new state, and repeatedly doing so, executing a *plan*, yields a trajectory over states. A *planning problem* specifies the current state and a property of state trajectories that is to be satisfied, the *goal*, typically a conjunction of propositions that should hold at the end of plan execution. A *planning domain* specifies the actions that may be executed, and defines the situations in which each particular action is *applicable*, and if so, then the result of executing the action.

Taken together, a definition of *states*, *actions*, *plans*, and *goals* is a *planning model*. We will treat planning models, as much as possible, as black boxes.

Definition 1 (State) A state, $S = (t, \mathcal{A})$, consists of the current time, t , and an assignment, \mathcal{A} , of the fluents of the domain. The assignment may be partial, in which case fluents not in the domain of the assignment are undefined.

Definition 2 (Action) An action, A , can be instantiated in time as a step, $a = (t, A)$, and is otherwise a black box.

Definition 3 (Plan) A plan, P , is a set of steps.

For plans, or any other time-stamped set, or sequences, we write the prefix as $Pt = \{(t', A) \mid t' \leq t \wedge (t', A) \in P\}$. Similarly, sP denotes the suffix after and including s , and sPt the sub-interval from and including s to t . To denote strict inequalities we write $P\overset{\circ}{t}$, $\overset{\circ}{s}P$, and $\overset{\circ}{s}P\overset{\circ}{t}$ for prefixes, suffixes, and sub-intervals respectively.

Let M be some planning model completing the above definitions, and let W stand for the world being modeled. Consider executing some plan P from the model. When the behavior predicted by M diverges from the actual behavior generated by W , the agent is faced with a decision: to replan, or not to replan. A theory of replanning, then, defines which is the correct choice; so that an agent can be evaluated as conforming more or less to the theory. We will use the following notation to precisely formulate our theory of replanning. Note that both the model M and the world W map plans to behavior. The key structural difference is that M can be asked to enumerate the future possibilities, whereas W can only be sampled from.

Definition 4 (Behavior) A behavior B is a trajectory on states: $B(t)$ is the assignment of fluents that holds at time t . Let $\Delta B(s,t)$ denote the net change in state from the assignment at s to the assignment at t : applying $\Delta B(s,t)$ to $B(s)$ gives $B(t)$.

An event, $E = \Delta B(t)$, is a discrete change in state: $\lim_{x \rightarrow t^-} B(x) \neq \lim_{x \rightarrow t^+} B(x)$. Let $B^-(t)$ and $B^+(t)$ denote the assignments immediately before and after the event, and let $\Delta^- B(t)$ and $\Delta^+ B(t)$ denote the previous and next events, respectively occurring at $t^- < t$ and $t^+ > t$.

While the world is free to generate whatever behavior it desires, we assume agents are not infinitely precise. So for any particular agent there is some maximum frequency, f , beyond which distinct events can no longer be distinguished. Instead, these are conflated together into a single observation (of the net change) at least $1/f$ away from the prior observation. Under this assumption (the agent's observations of B can be represented as a finite sequence of events and assignments, if desired).

Definition 5 (Plan Execution) Let $M(S,P)$ denote the model's prediction of the behavior that results from executing plan P from state S . The model can express ignorance by predicting a set or distribution of possible behaviors. Let $\hat{B} \in M(S,P)$ denote any behavior that is not unexpected by the model.

Let $B = W(S,P)$ denote the actual behavior given (non-deterministically) by the world W as a result of executing P (to completion).

A model failure occurs whenever $W(S,P) \notin M(S,P)$.

We separate the starting state S from P in order to consider continuing to execute the "same" plan even after model failures. An agent is of course free to change plans at any time, particularly in the face of model failure.

Definition 6 (Decision Sequence) Let I denote the agent under consideration, and S some initial situation. Denote the actual behavior experienced by I from S as $B = W(S,I)$ (ending whenever I so decides).

The decision sequence of B partitions B into sub-intervals separated by the decisions of I : $D(B) = t_0, P_0, B_1, t_1, P_1, B_2, t_2, P_2, \dots, B_n, t_n, P_n, B_{n+1}, t_{n+1}$. So $B_i = t_{i-1} B t_i$.

Decisions may either be plans, including the null plan, or a decision to deliberate (search for a plan). We write plan for this decision, for example, the first decision is always $P_0 = \text{plan}$. The last decision cannot be to deliberate.

Note that we require an agent's plans to be finite, so the decision to stop is implicit in completing P_n . Without some form of performance metric, then every decision is equally good. We consider learning a separate problem from replanning, so that if information is slowly revealed (as in reinforcement learning) about the true metric, we only require the agent to make effective decisions with respect to the metric it believes at the time of the decision.

Definition 7 (Performance) Let the quality of a behavior B be denoted $Q(B)$. The (model guaranteed) quality of a plan P (executed from S) is then:

$$Q(S,P) = \min_{\hat{B} \in M(S,P)} Q(\hat{B})$$

In summary, the future holds tremendous opportunities. It also holds pitfalls. The trick is to avoid the pitfalls, exploit the opportunities and get back home by 6pm. Woody Allen (MY Speech to Graduates)

These are the fundamental units of replanning: failures, commitments, and opportunities. Resolving failures is a necessary condition — a system that does not (attempt) to resolve failures can hardly claim to be replanning. However, we have nothing to add to the literature on resolving failures particularly effectively. Rather, our focus is entirely on the quality of the behavior of a replanning system. Consider, then, *respecting commitments*. Firstly:

Principle 1 An agent cannot be expected to respect an unknown commitment.

Corollary 1.1 An agent cannot be expected to commit to a condition it cannot be asked to achieve.

So, for example, it would be unfair to expect a replanner based on STRIPS to maintain a commitment to meeting at a particular time and place: a STRIPS planner does not accept deadline goals. It would be similarly unfair to expect a replanner based on STRIPS to invest a reasonable amount of effort in maintaining a commitment to attend a team lunch, because one cannot ask a STRIPS planner to achieve a goal only if its reward justifies the cost. One does not expect artificial systems to have children, but the continuation of the prior example is that the right plan to switch to if one's child falls deathly ill does not keep the commitment to attend lunch. Indeed, if one did keep the commitment, all those present would demand that one switch to the right plan (and begin to question one's humanity. . .).

So while one can certainly think of building a replanner on top of a STRIPS-based planner, and indeed STRIPS is actually the name of the classic AI replanning system (not the language), the kinds of commitments that can be fairly expressed are extremely limited. The lack of synchronization with the external world (no deadlines) is particularly cumbersome, as even one's boss (who would be pleased to have all commitments be hard constraints) will need to meet with one at particular places at particular times.

Claim 2 Respecting commitments requires² the underlying planner to support: (i) Metric Time (ii) Partial Satisfaction

Our plan is to model commitments as goals, and in the case of commitments to agents one is not beholden to, as soft goals. Consider opportunities. Opportunities come in two flavors: unexpected changes that reduce the cost of previously infeasible goals, and entirely new, optional, goals. In particular opportunities are soft goals, either directly or indirectly. This presents a problem: we cannot model both commitments and opportunities as soft goals. While both are goals, and both can be soft constraints, commitments and opportunities are not at all the same.

Consider holding down a job. As long as one maintains an acceptable level of performance one will not be fired — no matter how many opportunities one fails to exploit. No

²Except the narrowest class of commitments discussed above.

one is happy about this state of affairs, but still, at the end of the day, opportunities really are entirely *optional*. Commitments on the other hand, have definite cost. The exact number would vary from place to place, but failing to keep the commitment of showing up to work on time (especially if egregiously violated) with sufficient frequency will land one in a whole new level of replanning. Commitments are *not* optional, yet, at the same time they remain soft constraints — if circumstances warrant breaking a commitment, one can explain the difficulty to the injured party, apologize, make reparations, . . . , eventually the matter is closed and life goes on.

The approach we take is to extend the partial satisfaction problem to include *penalties* in addition to *rewards* for soft goals. Commitments, then, are characterized by possessing large penalties, while opportunities are characterized by possessing low/zero penalties and high reward. To complete the picture, goals with both high reward and high penalty are, in some sense, the “top level” goals. In the limit, goals with both high reward and high penalty approximate hard goals very well — one must achieve the goal (huge penalty) at any cost (huge reward).

Definition 8 (Goal) *A goal, g , is a proposition on state trajectories; g is either satisfied or unsatisfied by any given behavior. A plan, P , is valid (from S) only if all hard goals are satisfied in every possible behavior of $M(S, P)$. Every goal is associated with a reward $r(g)$ and a penalty $p(g)$; if $B = W(S, I)$ satisfies g then the agent I accrues the reward $r(g)$, otherwise (if g is unsatisfied by B), the agent accrues the penalty $p(g)$.*

Partial satisfaction problems already associate actions with costs: action A has cost $c(A)$.

Definition 9 (Net Benefit) *Let $B = W(S, P)$. The total reward accrued is $R(B) = \sum_{g \text{ is satisfied}} r(g) + \sum_{g \text{ is not satisfied}} p(g)$. The total cost incurred is $C(B) = \sum_{(t,A) \in P} c(A)$.*

The net benefit of the behavior is the reward minus the cost:

$$Q(B) = R(B) - C(B)$$

The (minimum) net benefit of the plan is given by $Q(S, P) = \min_{\hat{B} \in M(S, P)} Q(\hat{B})$.

These definitions alone do nothing to enforce a distinction between opportunities and commitments. One additional hard constraint, however, goes a long way:

$$Q(S, P) < 0 \text{ implies } P \text{ is invalid}$$

Consider a problem with no hard goals. Then every executable plan satisfying the constraint above is a solution. In particular, suppose all soft goals have no penalty. Then even the null plan is a solution: a lazy agent is permitted to forever ignore the opportunities surrounding it, no matter how large. Suppose we add a single commitment (i.e. a soft goal with penalty). Then even the laziest agent is forced into action, in proportion to the significance of the commitment. In the limit, only plans satisfying the commitment are solutions.

Note that we allow both rewards and penalties to be arbitrary (rational) numbers, and so to actually achieve the

intended effects one should set rewards to be positive and penalties to be negative. Inverting the signs establishes a negative goal. Using the same signs establishes a constant shift of the quality of all solutions: allowing one to raise or lower the bar on minimum acceptable performance in an indirect fashion.

Of course, we do not imagine that commitments between humans are in any way reducible to the simple arithmetic considered here. This is just a model; the real question is whether or not it would be a rational decision to believe the system if it claims to support a commitment. In the next section we investigate what guarantees can be given by a replanning agent.

Guarantees, Failures, and Blame

The essential difficulty in developing a theory of replanning is simply that one must allow the model of the artificial agent to be wrong; even catastrophically so. That is, the aim is to develop a system that reacts intelligently to developments unforeseen by the designers. An adequate theory of replanning, then, cannot place any bounds on just how dramatic an unexpected phenomena might be. Still, one can hardly demand that an agent avert disasters far beyond its abilities (e.g. stopping a tornado). A reasonable demand (one we make) of a replanning agent is to 1) live up to its claims, failing that, 2) justify the loss.

Definition 10 (Guaranteed Performance) *A replanning agent guarantees performance if, in any problem, subjected to arbitrary interference, the agent:*

- either** *achieves its maximum performance claim,*
- or** *the agent is not at fault*

In the following we give a formal account of *fault*. Briefly, frivolously wasting resources with no plan in mind is blameworthy, as is deliberately switching to an inferior plan, as is trying to claim ignorance on these two counts: an agent must evaluate whether or not it is frivolously wasting resources or switching to a worse plan. On the other hand, lack of mental or physical ability is never blameworthy.

Definition 11 (Unexpected Event) *An event $E = \Delta B(t)$, is unexpected if:*

$$\forall \hat{B} \in M(B^-(t), t^-P), E \neq \Delta \hat{B}(t)$$

Definition 12 (Failure) *Consider some event $F = \Delta B(t)$ encountered during the execution of P . Then the change in quality due to F , $\Delta Q(F)$, is given by:*

$$\Delta Q(F) = \left(\min_{\hat{B} \in \mathcal{B}} Q(Bt\hat{B}) \right) - \left(\min_{\hat{B} \in \mathcal{B}'} Q(Bt^-\hat{B}) \right)$$

where \mathcal{B} is the set of predicted behaviors given that F happened and \mathcal{B}' is the set of predicted behaviors before F is observed: $\mathcal{B} = M(B^+(t), tP)$ and $\mathcal{B}' = M(B^-(t), tP)$.

F is a failure if $\Delta Q(F) < 0$.

Besides deciding on which plan to execute, the agent, I , also faces the decision of continuing or suspending its current plan when unexpected events occur. We require all unexpected events to appear in the decision sequence of $B(D(B))$. If the agent chooses to ignore an unexpected event

$\Delta B_i(t_i)$ then $P_{i-1} = \hat{i}_i P_i$. Otherwise the agent suspends its current plan to look for a better one: $P_i = \text{plan}$.³

Let $Q'_i = Q(B^+(t_i), \hat{i}_i P_{i-1})$ be the predicted quality of continuing the old plan, and $Q_i = Q(B^+(t_i), P_i)$ the predicted quality of the switch to P_i .

I claims performance Q_i at every decision P_i (regardless of whether or not *I* wants to make claims). The maximum claim is then $Q^* = \max(Q_i)$; let $i^* = \text{argmax}(Q_i)$ be its index. Consider the actual quality achieved, $Q = Q(B)$, more importantly, the discrepancy: $Q - Q^*$.

Definition 13 (Fault) *If $Q < Q^*$ then someone is at fault for the loss of quality.*

Assigning blame in the general situation (when both the agent and external parties have made questionable decisions) is non-trivial. There are two situations where the party at fault is trivial to identify.

Judgement 1 *If for all $j > i^*$, $\Delta(F_j) \geq 0$ then all of the loss is the agent's fault. In particular:*

$$Q^* - Q \leq \sum_{j>i^*} Q'_j - Q_j$$

That is the loss in quality can be entirely explained by the subsequent blameworthy decisions ($Q'_j > Q_j$).

Judgement 2 *If for all $j > i^*$, $Q_j - Q'_j \geq 0$, and $Q_j \geq 0$, then some, possibly unknown, external agent(s) are to blame. In particular:*

$$Q^* - Q \leq - \sum_{j>i^*} \Delta(F_j)$$

In this case the agent only ever improves its plan, and if the environment interferes to the point of making the costs of the current plan outweigh its benefits ($Q'_j < 0$) the agent, at least, aborts the current plan in favor of replanning. In short, one cannot (justifiably) blame the agent for wasting resources ($Q_j < 0$) or making obviously bad decisions ($Q_j < Q'_j$).

In fact, even in the face of $Q'_j < 0$, if the agent is skilled and not too unlucky then it might have sufficient time to produce a plan with positive quality before the next action of the old plan would anyways have been dispatched ($\min_{(t,A) \in \hat{i}_i P_{j-1}} t$). For example, the replanner might employ triangle tables, or similar techniques, to aid in rapid recovery (Fikes, Hart, and Nilsson 1972). If so, then without knowledge of the specifics of P_{j-1} it would appear as if P_{j+1} was the intent of the agent all along; $P_j = \text{plan}$ could not be inferred from the behavior of the agent alone.

A Computational Model of Replanning

We take a simple architecture for a replanning agent; a monitor/executive serves as a buffer/interpreter between the world (or simulator) and the actual planner engine. The design questions then revolve around the two components within the agent: the monitor and the planner.

³Paranoid, or expert, agents have a third choice: switch to an already prepared plan for the situation resulting from the event deemed impossible by the model.

Monitor

The role of a monitor in an agent is to actually process sensory information, and implement choice over possible actions (deliberation always being possible). Effective monitors filter out irrelevant sensory input, focus attention on relevant events, and implement *rational* choice over possible action (i.e., evaluate known solutions and pick the best; deliberation is a universal solution (of very low quality)). Of these, filtering input and focusing attention are particularly challenging — we sidestep, ungracefully, the difficulties of implementing rational choice by limiting the memory of the agent to two possibilities, the current plan, and deliberation.

The approach we take to this problem is to perform *objective selection* on behalf of the planner. That is, the monitor performs an estimate of the impact of the changes in situation on its current objectives, and if a new set of objectives is estimated as superior, then the monitor interrupts the search effort to switch the description of the goal the planner is pursuing. The monitor further considers the possibility that the (estimated) effective methods of achieving the new objectives (possibly the same as the old) are significantly causally distinct from the situation the search was last initialized to (the planner performs forward search). If so, then the planning process is again interrupted, but now the entire search must be restarted. Our system performs continual planning, but it would be accurate to label this intervention of the monitor as *replanning*. In the following, we describe the objective selection and causal relevancy processes. First, however, we discuss communication with the world.

Updates We demand that updates from the world follow a simple PDDL(/LISP)-inspired syntax. By example:

```
(:update :objects
  aThing50 aThing51 - typeA
  bThing6 - typeB
:events
  (at 50 (propertyFoo aThing1 bThing3))
  (at 1000 (not (propertyBar aThing50 aThing51)))
  (at 80 (= (fluentBaz bThing6) 97.6))
:goal
  (at 2000 (propertyFoo aThing50 bThing6)[50, -10]) - soft
  (propertyBar aThing1 aThing51)[500] - hard
:now 70)
```

In general, updates consist of some number of:

1. New objects
2. Exogenous events
3. New or updated goals
4. The current time

Goals are on propositions of the domain, and there can be only one goal on any given proposition. By default goals are hard, lack deadlines, and have 0 reward and penalty. One can override the defaults as above, and one can get the effect of removing a goal by setting it to a soft goal with 0 reward and penalty.

All fields, except “:now”, may be repeated as many times as desired, or not at all, in any order. The assignment of the current time must happen last, in particular, just once per update. The intent is to allow simple accumulation of

updates, that is, without having to re-sort the data into the appropriate fields.

Systems situated in the real world would not get to demand input in a particular format, but rather would have to implement support for whatever form the input from sensors comes in. As we are pursuing a domain independent formulation, we assume the existence of higher level executives that translate to and from domain-dependent and domain-independent representations. We place some additional requirements on this translation:

1. Broadcast of the agent's plan (in some unobtrusive manner) is automatic (implicit in deciding to execute the plan).
2. Other agent's plans, if they choose to communicate such knowledge, are translated into the vocabulary of states; that is, actions get compiled out to exogenous effects and conditions.
3. Requests from other agents to achieve various conditions are represented as proscribed in Section , in particular, are, somehow, assigned numeric rewards and penalties.

The second condition is not hard to relax if the monitor is given a model of every relevant agent's abilities; by default we assume knowledge only of the abilities of the agent in question. Supporting the other conditions requires direct interaction with humans, worse yet, interaction in the form of communication. In short, a hugely important problem that goes way beyond the scope of this paper.

Lastly, we use the plan syntax of PDDL2. 1. For example:

```
0.01: (anAction aParameter anotherParameter)[10]
10.01: (anotherAction)[1.5]
```

Objective Selection We select objectives based off the estimate of reachability afforded by planning graphs; we make a straightforward modification of the relaxed temporal planning graph of SapaPS to support penalties (Do and Kambhampati 2004). This implementation already directly supports rewards, action costs, and soft goals; in support of soft goals it optimizes the heuristic estimate of net benefit by (greedily) eliminating unprofitable soft goals.

1. Build a cost-based relaxed temporal planning graph
2. Support all goals with a relaxed plan
3. Repeat until no change (hill climb)
 - (a) Repeat until no change (hill climb, steepest ascent):
 - i. For each goal, identify the set of actions in the relaxed plan with no purpose beyond achieving this goal (directly or indirectly)
 - ii. Compute the maximum of the penalty of the goal plus the cost of the identified actions minus the reward
 - iii. If greater than 0, remove the offending goal and the identified set of actions
 - (b) For each pair of goals, identify the set of actions in the relaxed plan contributing only to the pair, as above
 - (c) Compute the maximum, over the pairs, of the combined penalties plus the cost of the identified actions minus the combined rewards.
 - (d) If greater than 0, remove the offending pair of goals and the identified set of actions.

4. Set the current best set of objectives to the remaining goals
5. Remember the value of the selected objectives: the combined rewards of the selected goals and penalties of the removed goals.

The monitor then compares the new and old values to see which is superior, and if the new set, then make sure the new set actually differs from the old set (that is, the old value is simply out of date with respect to the new situation). If a new set has been selected, then the planner's search is interrupted to inform it of the current intended set of objectives (as well as its value, used for pruning) and then allowed to resume its search.

Causal Relevancy The analysis employed for selecting objectives first computes an entire planning graph: an estimate of the reachability of everything in the domain, not just goals. In particular, the analysis estimates the set of reachable actions. The planner already employs this information (to avoid even attempting to branch on actions not in the set); we track this information in the monitor as well. Even if new goals are not selected, the monitor compares the old and new estimates of reachable actions.

Consider the case when a previously reachable action (by the estimate) is marked as unreachable. The planner may be considering some, perhaps many, partial plans that employ the action. There are, at least, two choices:

1. Restart the search completely
2. Lazily filter out the invalid states

Restarting completely discards the existing search effort. Lazily filtering adds an additional set of operations per search expansion, resulting in exponentially many comparisons and deallocations. A single giant deallocation (restarting the search) saves the cost of those comparisons and any overhead incurred by managing memory. It is a difficult choice; we choose to restart the search in this case, but filtering is worth considering. So, in this case, the monitor interrupts the planner, clears and re-initializes the search, including passing in the new set of reachable actions (estimated), and then allows the planner to resume.

Otherwise — every action considered reachable is still considered reachable — there remains the possibility that the estimated set has increased. Supposing not, then the choice is easy: allow the planner to continue working without distractions. If so, then again there are, at least, two choices:

1. Swap in the new estimate, and restart search
2. Just swap in the new estimate

Consider restarting the search (with the new actions). This throws away prior search effort, on the other hand, the planner might find a way to exploit the action early on in the new search space. Then the cost of the second approach is clear: the opportunity is lost to exploit the new actions at those partial plans that have left the open list. The choice is not trivial, but seeing as how the latter cost is an *opportunity cost*, that is the choice we make. So the monitor interrupts the planner, swaps in the new estimate of reachable

actions, and then allows the planner to resume right where it left off. One could also consider employing further analysis to determine if the new actions are *relevant* to the selected objectives, and if not, then allowing the planner to continue without distraction.

Normally replanning evokes the image of alternating epochs of planning and execution. Our system *continually* plans, an approach typically distinguished from replanning. Even so, consider the sense of dread invoked by “replanning”: one must start all over. So we say our system “replans” when the monitor forces the search to start over completely.

We consider these design decisions to strike a middle ground between two extreme perspectives on when to re-plan. Consider a traditional approach to replanning, such as STRIPS. If the state changes, but the current plan remains executable, then replanning is not invoked. That is, only failures that would drop performance below executability trigger replanning. In the other extreme (Fritz and McIlraith 2007), replanning is triggered at the slightest provocation.

The analogy does only go so far. In particular, it is important to keep in mind that even though the monitor may make decisions to interrupt the planner, possibly even erasing all its work, it is not the case that the agent stands around doing nothing — as long as the current plan remains profitable ($Q(S,P) > 0$), implying executable, the monitor will continue its execution. The monitor stops to think only when the environment interferes dramatically enough to kill the executability of its current plan.

Planner

The core requirements of our replanning theory, upon the underlying planner, are:

1. Metric Time
2. Partial Satisfaction

There are quite a few systems that can reason with metric time (Younes and Simmons 2003; Halsey, Long, and Fox 2004; Do and Kambhampati 2003; Gerevini and Serina 2002; Chen, Hsu, and Wah 2006; Bacchus and Ady 2001), as well as a few systems that can solve partial satisfaction problems (Do and Kambhampati 2004; Nigenda and Kambhampati 2005), but just one metric temporal partial satisfaction planner (that we know of): SapaPS, an extension of Sapa to partial satisfaction problems (Do and Kambhampati 2004; 2003). Given that the International Planning Competition (IPC) has recently added a track on partial satisfaction planning, judging from the history of the IPC, within at most a few years there will be several significantly superior alternatives to Sapa (perhaps a further extension of itself?). For now, however, it is more or less our only choice for metric temporal partial satisfaction planning.

Updates Integrating domain-independent planners into agent architectures is non-trivial; planners, including SapaPS, take enormous advantage of the fact that the problem given is (supposedly) static. In particular a large amount of analysis is performed prior to search in order to improve efficiency. In the context of replanning, reusing the search

effort requires maintaining consistency with the internal representation produced by this analysis. In our system, we setup the interface between monitor and planner at the internal representation of the problem. This allows the monitor to preserve the state of the planner’s search while making modifications to the problem, or the set of goals being pursued, as the monitor can ensure that the internal names⁴ of entities remain consistent.

Objective Selection The monitor focuses the attention of the planner by selecting objectives; this is operationalized in the planner by extracting heuristics based on supporting all and only those objectives. As a natural result, states that cannot extract efficient relaxed plans for these objectives are ignored. The interesting issue that this raises is that there are now two evaluation functions for states — all the goals, or just the selected goals. In the search, should the planner stumble upon a search node that evaluates well against the true evaluation function, it reports this solution to the monitor.

In fact, since the planner is free to return solutions in any order desired (as long as it gets to them all eventually) we employ a third evaluation function (estimated plan size) to combat enormous magnitude differences in cost in some of the SapaPS benchmarks. If searching by cost, the effectively 0 cost actions form traps (plateaus) for the search. Ordering by estimated plan size will eventually get around to considering all plans, in a reasonably fair manner.⁵

Summary

In summary, the framework established by our theory of replanning allows the monitor to set the planner to aggressively search for a satisficing solution (relaxed plan execution, helpful actions, hill climbing or beam search, inadmissible cost-based pruning, . . .), and then, as time and computational power permit, to converge to an A* search with admissible heuristics, that is, eventually find an optimal plan.

More importantly, the framework does not compromise on representing the static dependencies this agent has with other agents (commitments and opportunities), while at the same time avoiding the combinatorial (and logistical) nightmare of full multi-agent planning.

Related Work

Proponents of replanning as plan reuse, as exemplified by (van der Krogt and de Weerd 2005), have doubly confused the issue of replanning quality. The first approximation, considered at length in this paper, is in assuming that reducing perturbation caused to other agents can be modeled by minimally altering the structure of plans across iterations: minimal perturbation planning. The second approximation is applying their plan reuse algorithm instead of truly minimal

⁴The number of names then grows monotonically over time; we retain space efficiency by universally transforming the use of these names from *indexes* to *keys*.

⁵Since estimated plan size is integer-valued on a small range, it helps greatly to add a small tie-breaking term. We add a small cost-sensitive term: the heuristic cost normalized by the initial state heuristic cost

perturbation planning. That is, minimal perturbation planning is just as inadequate in application to plan reuse as it is in application to replanning – though for different reasons (Nebel and Koehler 1995). Specifically, minimal perturbation planning has greater complexity than plan synthesis, which is in direct conflict with the speedup motivation of plan reuse. As noted by Nebel and Koehler, plan reuse systems actually return highly, but not maximally, similar plans.

The robot path planning community (Stentz 1995; Koenig, Likhachev, and Furcy 2004) has long looked at replanning slightly differently from the planning community.⁶ In particular they try to ensure that the plan produced by the "re-planner" is as optimal as the one that would have been produced by the from-scratch planner. Due to the nature of robot path planning, this work does not consider the commitments made by the partial execution of the prior plan. One point of similarity is that the robot path planning community does model failures that involve more than initial and goal state changes—action deletion (e.g. certain navigation actions made infeasible by the appearance of new obstacles). This is a kind of systematic failure, which we generalize further based on accounts of real-world replanning systems (Pell et al. 1997; Myers 1999).

Within the planning community, the work by Pell et al. comes closest to recognizing the importance of respecting the commitments generated by a partially executed plan. They however do not give any formal details about how the replanning is realized in their system.

Conclusion

The observation underlying our work is that replanning lacks a formal definition. At a high level, researchers bring up the concepts of commitments, reservations, approximate domain models, mixed-initiative and distributed planning, tightly bounded computational resources, and execution failures. Then a syntactic measure is introduced with only a loose connection to these motivations. While some, even many, approximate solvers will be superior to optimal solvers in any domain (*bounded rationality*), it is still necessary, in the design of such solvers, to evaluate the tradeoff of quality for computation time.

We tackle this problem by considering the intuitively desirable behavior for a set of interesting replanning scenarios. We find that it is easy to explain the superiority of the desired behavior in terms of commitments to external agents and new opportunities. We develop a formal theory of replanning that directly captures commitments and opportunities, and admits as solutions only those plans that treat these constraints in good faith. If the commitments and opportunities are transcribed accurately, then the solutions of the theory correspond to the intuitively desirable behavior first considered.

A theory of replanning is at most half of the problem, and interesting theories far predate this work. Our main con-

tribution is in demonstrating *how* to implement the theory, and that doing so is *feasible*. That is we go to no effort to speedup the replanning effort over the planning effort, and find that the additional constraints do not appear to significantly (or even noticeably) impact the computational performance of the planner. In conclusion, we present a proof of concept that existing planning technology, with straightforward modifications, is capable of supporting a direct treatment of commitments and opportunities without the sacrifices inherent in prior approaches to replanning.

References

- Bacchus, F., and Ady, M. 2001. Planning with resources and concurrency: A forward chaining approach. In *IJCAI*.
- Bratman, M. E.; Israel, D. J.; and Pollack, M. E. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence* 4:349–355.
- Chen, Y.; Hsu, C.; and Wah, B. 2006. Temporal planning using subgoal partitioning and resolution in SGPlan. *JAIR* to appear.
- Cushing, W., and Kambhampati, S. 2005. Replanning: A new perspective. In *Proc. ICAPS Poster Program*.
- Do, M. B., and Kambhampati, S. 2003. Sapa: A multi-objective metric temporal planner. *J. Artif. Intell. Res. (JAIR)* 20:155–194.
- Do, M., and Kambhampati, S. 2004. Partial-satisfaction (over-subscription) planning as heuristic search. In *Proc. Knowledge Based Computer Systems*.
- Fikes, R.; Hart, P. E.; and Nilsson, N. J. 1972. Learning and executing generalized robot plans. *Artif. Intell.* 3(1-3):251–288.
- Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *ICAPS*, 212–221.
- Fritz, C., and McIlraith, S. 2007. Monitoring plan optimality during execution. In *ICAPS*.
- Gerevini, A., and Serina, I. 2002. LPG: A planner based on local search for planning graphs. In *AIPS*.
- Halsey, K.; Long, D.; and Fox, M. 2004. CRIKEY - a temporal planner looking at the integration of scheduling and planning. In *Workshop on Integrating Planning into Scheduling, ICAPS*, 46–52.
- Kitano, H., and Tadokoro, S. 2001. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI Magazine* 22(1):39–52.
- Koenig, S.; Likhachev, M.; and Furcy, D. 2004. Lifelong planning a. *Artif. Intell.* 155(1-2):93–146.
- Myers, K. L. 1999. Cpef: A continuous planning and execution framework. *AI Magazine* 20(4):63–69.
- Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artif. Intell.* 76(1-2):427–454.
- Nigenda, R. S., and Kambhampati, S. 2005. Planning graph heuristics for selecting objectives in over-subscription planning problems. In *ICAPS*, 192–201.
- Pell, B.; Gat, E.; Keesing, R.; Muscettola, N.; and Smith, B. 1997. Robust periodic planning and execution for autonomous spacecraft. In *IJCAI*, 1234–1239.
- Stentz, A. 1995. The focussed d* algorithm for real-time replanning. In *IJCAI*, 1652–1659.
- van der Krogt, R., and de Weerd, M. 2005. Plan repair as an extension of planning. In *ICAPS*.

⁶See (Fritz and McIlraith 2007) for an example of this style of replanning being done in a PDDL-planning agent

Younes, H., and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *JAIR* 20:405–430.