A Decision Theoretic Replanning Algorithm and its Empirical Evaluation

Guido Boella and Rossana Damiano
Dipartimento di Informatica
Cso Svizzera 185 Torino ITALY
{guido,rossana}@di.unito.it

Abstract. In this paper, we present a replanning algorithm for a utility-based, hierarchical planner, and we discuss the empirical evaluation of its performance. The algorithm is embedded in an autonomous agent architecture which includes a meta-deliberation component. The aim of the experimental setting is to investigate, given a utility-based notion of plan failure, the role of replanning in different domains, and to evaluate the preferability of replanning versus planning from scratch when a plan fails during execution.

1 Introduction

Although the use of planning techniques has been often investigated in real world domains, the evaluation of their performance in the context of autonomous agent architectures constitutes a primary research issue. In particular, when situated in non-deterministic, dynamic worlds, agents are likely to experience the failure of their plans, and the need for re-deliberation on them.

Theories of rational behavior claim that intentions are characterized by stability and tend to be revised at lower level rather than at higher level [Boella, 2002, Bratman, 1987]. According to this claim, conservative re-deliberation should be preferred over forms of re-deliberation which do not consider current intentions: higher level intentions (represented in the form of abstract plans) should be maintained, while lower level intentions (refinements of those abstract plans) should be changed to adapt to a new situation.

The replanning strategy we evaluate in this paper is inspired to this notion of persistence of intentions, in that it tries to perform the most local replanning which allows the expected utility to be brought back to an acceptable difference with the previously expected one. The experimental setting for empirically evaluating re-deliberation strategies is an autonomous agent architecture inspired to [Boella and Damiano, 2002b]. Here, we compare two alternatives, replanning, and planning from scratch, and propose a methodology for evaluating their preferability in a given domain. In this methodology, the alternatives of replanning and planning from scratch are evaluated based on their success rate, their time performance, and the quality of output plans.

The paper is structured as follows. In the next section, we introduce the agent architecture and its deliberation and execution components. Then, we present the experimental methodology, and discuss the results obtained by applying it to two different domains. Conclusion and related work close the paper.
2 The agent architecture

The architecture is composed of a deliberation module, an execution module, and a sensing module, and relies on a meta-deliberation module to evaluate the need for re-deliberation, following [Wooldridge and Parsons, 1999]. The internal state of the agent is defined by its beliefs about the current world, its goals, and the intentions, or plans, it has formed in order to achieve a subset of these goals. Intentions are dynamic, and can be modified as a result of re-deliberation. The agent’s deliberation and re-deliberation are based on decision-theoretic notions: the agent is driven by the overall goal of maximizing its utility based on a set of preferences encoded in a utility function.

The agent is situated in a dynamic environment, i.e. the world can change independently from the agent’s actions, and actions can have non-deterministic effects. Moreover, it does not assume a perfect correspondence between the environment actual state and the agent’s representation of it.

The behavior of the agent is controlled by an execution-sensing loop with a meta-level deliberation step When this loop is first entered, the deliberation module retrieves a high-level plan from the plan library based on its match with the initial goal, and passes it to the planner for refinement. The best plan becomes the agent’s current intention, and the agent starts executing it. After executing each action in the plan, the sensing module monitors the obtained effects, and updates the agent’s representation of the world. In order to decide whether re-deliberation is needed or not, the meta-deliberation component is invoked on the current plan and the updated representation of the world.

The core of the meta-deliberation module is constituted by an execution-monitoring function, which relies on the agent’s subjective expectations about the utility of a certain plan: after the execution of each step, given the result of sensing, this function computes the expected utility of executing the remaining plan steps. If the difference between the previously expected utility and the new one is below a certain threshold, the plan has failed during its execution, and replanning is performed. If new deliberation is necessary, the deliberation module invokes the replanning component on the current plan with the task of finding a better plan. On the contrary, if new deliberation is not necessary, the meta-deliberation module simply updates the execution record and releases the control to the execution module, which executes the next action.

3 Deliberation and Execution

3.1 The planning algorithm

The planning component of the deliberation module builds on the DRIPS system, a decision-theoretic refinement planner which searches the plan space for optimal plans [Haddawy and Hanks, 1998,Haddawy and Suwandi, 1994]. The action library is organized along two abstraction hierarchies. The sequential abstraction hierarchy is a task decomposition hierarchy: an action type in this hierarchy is a macro-operator which the planner can substitute with a sequence of (primitive or non-primitive) action types. The specification hierarchy is composed of abstract action types which subsume more specific ones. In the following, for simplicity, we will refer to sequentially abstract actions as complex actions and to actions in the specification hierarchy as abstract actions.
procedure plan replan(plan p, world w)
/* find the first action which will fail */
(action a := find-focused-action(p,w);
mark a; //set a as the FA
plan p' := p;
plan p'' := p;
/* while a solution or the root not found */
while (not(achieve(p'',w, goal(p'')))
and has-father(a))
/* look for a partial plan with better EU */
{while (not (promising(p', w, p))
and has-father(a))
(p' := partialize(p');
project(p',w);) //evaluate action in w
/* restart planning on the partial plan */
p'' := refine(p',w);}
return p'');

Fig. 1. The main procedure of the replanning algorithm, replan.

A plan is a sequence of action instances and has associated the goal the plan has been planned to achieve. A plan can be partial both in the sense that some steps are complex actions and in the sense that some are abstract actions. Before a partial plan is refined, the agent does not know which plan (or plans) is the most advantageous among those it subsumes in the plan space. Hence, the expected utility of the abstract plan is expressed as an interval having as upper and lower bounds the expected utility of the best and the worst outcomes produced by substituting in the plan the abstract actions with all the more specific actions they subsume.

The planning process starts from the topmost action in the hierarchy which achieves the given goal and refines the current plan(s) by substituting complex actions with their decomposition and abstract actions with all the more specific actions they subsume, until it obtains a set of plans composed of primitive actions. After the refinement step, the expected utility of each plan is computed by projecting it from the current world state; suboptimal plans, i.e., plans whose expected utility upper bound is lower than the lower bound of some other plan \( p \) are pruned. On the contrary, plans which have overlapping utilities need further refinement before the agent makes any choice.

In our framework, the representation of a plan is associated with its derivation tree (including both abstract and complex actions), which has been built during the planning process and will be used in the replanning phase.

3.2 The replanning algorithm

If a replanning phase is entered, it means that the current plan does not reach the agent’s goal, or that it reaches it with a very low utility compared with the initial expectations. However, even if the utility of the current plan drops, it is possible that the current plan is ‘close’ to a similar feasible solution, where closeness is represented by the fact that both the current solution and a new feasible one are subsumed by a common partial plan at some level of abstraction in the plan space.
function plan partialize(plan p)
(action a := marked-action(p); // a is the FA of p
/* if subsumed by a partial action */
if (abstract(father(a)))
/* delete a from the tree */
{delete(a, p);
 return p;}
/* we are in a decomposition */
else if (complex(father(a))
{a1 := find-sibling(a,p);
if (null(a1))
/* there is no FA in the decomposition */
{mark(father(a)) //set the FA
 //delete the decomposition
 delete(descendant(father(a)),p);
 return p;}
else //change the current FA
{unmark(a);
 mark(a1);
 mark(a1);}}}

Fig. 2. The function for making a plan more abstract, partialize.

The key idea of the replanning algorithm is then to make the current plan more partial, until a more promising partial plan is found: at each partialization step, the current plan is replaced by a more partial plan by traversing the abstraction and decomposition hierarchies in a upsidedown manner, and the planning process is restarted from the new partial plan in search for more promising alternatives. The abstraction and the decomposition hierarchy play complementary roles in the algorithm: the abstraction hierarchy allows identifying the alternatives to the current plan steps, while the decomposition hierarchy focuses the replanning process on a portion of the plan. For an example of this strategy, see [Boella and Damiano, 2002a,Boella and Damiano, 2002b].

Notice that, by traversing the action hierarchy upsidedown, the replanning algorithm is bound to find a solution, if there is one; however, in the worst case, it will reach the action hierarchy root before finding it. This approach is somehow complementary to the approach proposed by [Lemai and Ingrand, 2003], which is based on the repair of broken causal links in partial order planning.

First, the find-focused-action function identifies the initially ‘focused action’, i.e. the next action whose preconditions do not hold (see the replan procedure in Figure 1). Then, given the derivation tree associated with the plan, the replanning algorithm partializes (i.e., makes more partial) the plan starting from the focused action (FA) (see the partializes function in Figure 2).

If the FA is directly subsumed by an abstract action type in the derivation tree, the focused action is deleted and the abstract action substitutes it in the tree frontier which constitutes the plan. On the contrary, if FA appears in a decomposition (i.e., it is directly subsumed by a complex action) then two cases are possible (see the find-sibling function in Figure 3):

1. There is some action in the plan which is a descendant of a sibling of FA in the decomposition and which has not been examined yet: this descendant of the sibling
function action find-sibling(a, p)
/* get the next action in the plan to be refined (in the same decomposition as a) */
(action a0 := right-sibling(a, p);
 action a1 := leftmost(descendant(a0, p));
 while not (null (a1))
   {if not complex(father(a1))) //if can be partialized
      {unmark(a); //change FA
       mark(a1);
       return a1;}
   a0 := right-sibling(a0, p);// move to next action
   a1 := leftmost(descendant(a0, p));}
/* do the same on left side of the plan */
 action a0 := left-sibling(a, p);
 action a1 := rightmost(descendant(a0, p));
 while not (null (a1))
   {if not complex(father(a1)))
      {unmark(a);
       mark(a1);
       return a1;}}
action a1 := left-sibling(a, p);)

Fig. 3. The procedure for finding the new focused action.

becomes the current FA. The order according to which siblings are considered reflects the assumption that it is better to replan non-executed actions, when possible: so, right siblings (from the focused action on) are given priority on left siblings.

2. All siblings in the decomposition have been already refined (i.e., no one has any descendant): all the siblings of FA and FA itself are removed from the derivation tree and replaced by the complex sequential action, which becomes the current FA (see Figure 3).

Each time a plan $p$ is partialized, the resulting plan $p'$ may subsume other plans whose outcomes is more advantageous than the outcome of $p$ (by the definition of abstraction previously discussed, the outcome of an abstract action includes the outcomes of all the actions it subsumes). If this is the case (see the promising condition in the procedure replan) the refinement process is restarted on the current partial plan; if not, the current partial plan is partialized.

For what concerns the complexity issues, it must be noticed that the replanning algorithm works in a similar way as the iterative deepening algorithm. The complexity of searching the plan space for a new solution starting from the current partial plan is alleviated by the fact the planning algorithm, when invoked in the replanning phase, exploits the pruning heuristics as in normal planning, and thus discards suboptimal alternatives.

3.3 Plan Execution

As described in Section 3.2, deliberation and meta-deliberation are based on the agent’s subjective representation of the world, updated by the sensing component after the execution of each new step.
The representation of a world is constituted by a set of attribute-value pairs \( (W_o = \{ \{ A_1, V_1 \}, ..., \{ A_n, V_n \} \} ) \). In the objective representation of the world, all attributes have a single value, while the subjective representation of the world, an attribute can have a set of values with a probability distribution \( (W_s = \{ \{ A_1, \{ V_{11}, P_{11} \}, ..., \{ V_{1n}, P_{1n} \} \} , \), \( ..., \{ A_n, \{ V_{1n}, P_{1n} \}, ..., \{ V_{nn}, P_{nn} \} \} ) \), where \( \sum (P_{11}, ..., P_{nn}) = 1 \) or \( W_s \) can be a set \( \{ \{ W_1, P_1 \}, ..., \{ W_n, P_n \} \} \).

Representing the agent beliefs about the world separately from the objective representation of the world allows us to model situations in which the subjective world representation is uncertain, while the objective world representation does not contain any uncertainty. When execution begins, the representation of the objective world in the simulation module consists of a world where each attribute has a certain value, representing the actual world, while the agent’s initial representation of the world is constituted by a probability distribution over worlds.

The representation of the external world is maintained by a simulator according to the input it receives from the execution module. When the agent executes an action, the simulator matches the action conditions against the world representation, and updates it with the effects associated to the condition which holds in it. If the agent executes an action which has non-deterministic effects, the simulator generates \( n \) worlds corresponding to the \( n \) alternative effects of the action being executed, then picks up a world according to the probability distribution associated to the action effects.

4 Empirical Evaluation

4.1 Experimental Methodology and Scenarios

In order to experimentally test the issues expressed in the Introduction, we arranged two different experimental settings, that we applied to two different domains. In both cases, the process of running execution experiments involve the following steps: generating the initial subjective and objective representations of the world, setting the deliberation and meta-deliberation parameters of the agent (the agent’s goal and utility function, its planning library, and the ratio which determines the replanning threshold), invoking the agent loop by recording the relevant data.

The first setting depicts the situation in which the agent has incorrect initial beliefs about the world. As a consequence, the expected utility of the plan is likely to drop during execution. Instead of generating arbitrarily different initial subjective and objective representations of the world, we obtain the objective world representation from the subjective representation of the world.

The process of generating the initial subjective and objective world representations is accomplished in the following way: first, the subjective world is generated, then, the objective representation is generated by altering it. The subjective representation is generated according to a set of attribute-range pairs, which express plausible value ranges for the world attributes in the initial world. For each attribute, it is also specified whether its value is to be uncertain or not in the initial world \( (P^U \{ \{ A_1, \{ V_{11}, ..., V_{1n} \}, U_1 \}, ..., \{ A_n, \{ V_{11}, ..., V_{nn} \}, U_n \} \} = \{ \{ A_1, \{ V_{11}, P_{11} \}, ..., \{ V_{1n}, P_{1n} \} \} , ..., \{ A_n, \{ V_{1n}, P_{1n} \}, ..., \{ V_{nn}, P_{nn} \} \} \} \); if \( P = 0 \), \( U_i = 0 \), for all but one \( V_{ni} \).
If the agent’s subjective representation of the world is characterized by uncertainty, then we exploit this feature to generate the objective world, by drawing one of the worlds it subsumes; the drawing accounts for the probability distribution on simple worlds associated to a complex world ($F_O(^{\{ A_1, \{ V_1^1, P_1^1 \} \}, \ldots, \{ A_n, \{ V_n^1, P_n^1 \} \}, \ldots, \{ V_n^m, P_n^m \} ) = \{ A_1, V_x^1 \}, \ldots, \{ A_n, V_y^n \} )$). On the contrary, if the agent’s subjective representation does not contain any uncertainty, then a random number of attributes in the subjective world are drawn from a predefined set and altered by substituting their values in the subjective world with a new random value in the prescribed range ($F'_O(\{ A_1, V_1 \}, \ldots, \{ A_n, V_n \} ) = \{ A_1, V_x^1 \}, \ldots, \{ A_n, V_y^n \} )$).

Then, given the initial subjective and objective world, the agent loop is launched on these representation. The agent’s deliberative component devises an executable plan and starts to execute it. If the agent’s meta-deliberation component detects a significant drop of the expected utility during the execution of this plan, the replanning process starts. In order to compare the strategies of replanning and planning from scratch, if the replanning algorithm outputs a new plan, the agent attempts to elaborate a new plan by planning from scratch given the subjective world representation at the moment of the drop in the expected utility.

The second setting is characterized by the initial coincidence of the subjective and objective representations of the world. In order to introduce an uncertainty factor in the execution (besides that constituted by the presence of non-deterministic plan steps), the effects of the plan steps are randomly altered after they have been executed, as a simulation of execution failures and unexpected world changes. In practice, the objective world representation is altered (prior to the agent’s sensing action) by modifying the effects of the last executed action in the following way: given the world attributes involved in the action effects, a subset of them is randomly drawn, and for each of them, a new value is randomly drawn in the range given by its value prior to the execution and its value after the execution. Formally, if $V_n^m$ and $V_m^n$ are the values of attribute $A_n$ before and after the execution of step $S$, the failure generation algorithm computes the range $\{ A_n, V_n \}$, then extracts a new value $V_x^m$, where $1 \leq x \leq m$ ($F_F(\{ A_n, m \} ) = \{ A_n, V_x^m \} )$).

In order to investigate the impact of plan failures and the trade-off between planning and replanning in a domain-independent way, the scenarios described above have been applied to two different domains. These domains differ along several dimensions: the plan library complexity (measured in terms of maximal and minimal plan length, action hierarchy depth), the degree of non-determinism in action definitions, and the complexity in the representation of the world (number of attributes involved and initial world uncertainty).

The first domain ($A$) concerns industrial brewing planning and is characterized by non-deterministic actions; plans have a minimal length of 3 steps and a maximal length of 7 steps [Haddawy and Suwandi, 1994]. The second domain ($B$) has been designed for this study, and represents an office toy world and concerns the planning of a complex mail delivery task (adopted also by [Dastani et al., 2003]. Its plan library is characterized by a lower depth, and actions are prominently non deterministic. The initial world definition is characterized by a lower number of defining attributes. However, the plan length varies between 9 and 12 steps.
<table>
<thead>
<tr>
<th>Domain</th>
<th>Replanning rate</th>
<th>Successful replanning rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>38.5</td>
<td>18% (relative) 6.7% (absolute)</td>
</tr>
<tr>
<td>B1</td>
<td>26</td>
<td>100% (relative) 26% (absolute)</td>
</tr>
<tr>
<td>A2</td>
<td>23.7</td>
<td>16.8% (relative) 4% (absolute)</td>
</tr>
<tr>
<td>B2</td>
<td>44</td>
<td>100% (relative) 44% (absolute)</td>
</tr>
</tbody>
</table>

Fig. 4. The replanning rate in each domain-setting pair and the successful replanning rate, relative and absolute.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Average replanning time</th>
<th>Average expected utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>0.6% (on avg time of planning from scratch)</td>
<td>196.7% (on avg EU of planning from scratch)</td>
</tr>
<tr>
<td>B1</td>
<td>14.3%</td>
<td>96.2%</td>
</tr>
<tr>
<td>A2</td>
<td>10.9%</td>
<td>126%</td>
</tr>
<tr>
<td>B2</td>
<td>14.7%</td>
<td>106.5%</td>
</tr>
</tbody>
</table>

Fig. 5. The relation between the replanning time and the time required by planning from scratch (left), and the relation between the expected utility of the plan obtained by planning from scratch and the expected utility of the plan obtained by replanning (right).

4.2 Discussion of the Data

In the following, we give a sketch of the data in the two domains, and make some tentative hypotheses about the correlation between the features of the domain and the result. Finally, we will try to assess the significance of each experiment set.

For each domain, we performed a separate set of experiments by applying each of the two settings presented in the previous section. Since these experiments constitute a preliminary attempt to assess the performance of the replanning algorithm in different domains, we arbitrarily set the size of the experiment set to a minimum of 100 items for each setting-domain pair.¹

For the evaluation of each set of experiments, we considered these parameters:

- First of all, we measured the overall replanning rate. This value provides useful indications about the probability that the current plan encounters a failure during its execution in the given domain. Apart from the inherent features of the domain (non-determinism, initial uncertainty), the absolute replanning rate is influenced by two main factors: the type of experimental setting, and the threshold under which replanning is triggered. The latter value is obtained by applying a constant ratio to the initial expected utility. In the experiments, we obtained the threshold by applying a fixed ratio of 0.1 to the upper bound of the initially expected utility.

In order to verify the effectiveness of the replanning algorithm we measured the

¹ The exact number of items contained in each set of experiments is the following: domain A, setting 1: 400 items (divided into four sets of 100, each characterized by a different configuration of utility parameters); domain A, setting 2: 400 items (again, four sets of 100); domain B, setting 1: 100 items; domain B, setting 2: 100 items.
success rate of the replanning process; the **successful replanning rate** approximately tells us the probability that the replanning process, when invoked in that particular domain, outputs a new plan. By calculating the **absolute successful replanning rate**, it is then possible to evaluate the relevance of successful replanning in the overall experiment set.

- The replanning time vs. the time required by planning from scratch. The **average replanning time** (expressed as a percentage on the time needed to plan from scratch) indicates how much time is saved (or wasted) by replanning instead of re-invoking the planning algorithm from the start. In other words, it provides a comparison of the efficiency of the two strategies.

- The difference between the (higher) expected utility of the plan obtained by replanning and the (higher) expected utility of the plan obtained by planning from scratch. This value, by comparing the quality of the plans obtained by the two approaches, provides an evaluation of the **effectiveness** of the two approaches. If the value of the **average expected utility** (again, expressed as a percentage) is below 100, the plans obtained by replanning are more advantageous than the plans obtained by planning from scratch.

The empirical evaluation conducted in the two domains (see Figures 4 and 5) seems to suggest that the domain constitutes a important factor in determining both the relevance of replanning and the effectiveness and efficiency of the re-deliberation strategies we tested (replanning and planning from scratch). For this reason, we will discuss the data grouped by domain A and B.

In domain A, the replanning rate is quite low if compared to the other domain (38.5% in setting 1 and 23.7% in setting 2), and the successful replanning rate is, respectively, of 18% and 16.8% in the two settings (this means that also planning from scratch is not able to find a solution after a failure); this value reduces to 6.7% and 4% if we consider the absolute successful replanning rate (see Figure 4, rows A1 and A2). Although the impact of plan failures and replanning does not appear to be high in this domain, the analysis of efficiency (time) and effectiveness (expected utility) of the replanning algorithm in this domain points out a better performance of this approach with respect to the planning from scratch approach (see Figure 5, rows A1 and A2). The average replanning time is 0.6% of the planning time in setting 1, and 10.9% in setting 2. The average expected utility of the plans obtained by planning from scratch is 196.7% of the average expected utility of the plans obtained by replanning in setting 1 (significantly higher), and 126% in setting 2.

In domain B, we observe a replanning rate of 26% in setting 1 and 44% in setting 2, and a replanning success rate of 100% in both setting 1 and 2, i.e. the replanning process always succeeded (see Figure 4, rows B1 and B2). For what concerns efficiency and effectiveness of replanning (see Figure 5, rows B1 and B2), in both settings planning from scratch took longer than replanning (with a value of 14.3% and 14.7% respectively). The average expected utility of the replanned plans is approximately the same as the average expected utility of the plans obtained by planning from scratch (96.2% in setting 1 and 106.5% in setting 2).

Given the preliminary data presented above, replanning constitutes by far the best approach in both domains. The ratio of average time performance of the replanning algorithm on the average time performance of the planning time is never over 0.15.
As the observations show, the replanning algorithm we tested appears to perform best in domain $A$ - characterized by a relatively high action hierarchy and a fair level of non-determinism - and slightly worse in domain $B$, whose complexity is lower.

5 Conclusions and Future Work

The data collected by preliminary experiments seem to confirm the claim according to which conservative replanning, represented here by a replanning strategy for utility-based hierarchical planning, is more advantageous than planning from scratch. However, the variance of plan failure rate across domains and experimental settings leads to hypothesize a methodology according to which it is essential to assess the impact on plan failures by extended simulations in the given domain (and the effectiveness of different replanning strategies) before committing to a re-deliberation strategy. In some cases, simulations may point out the irrelevance of replanning itself, while in other cases, they may reveal the need for predisposing effective replanning strategies.

Since the notion of failure we employed in this framework is based on the notion of initially expected utility, there is a correlation between the agent utility function, the acceptability threshold on utility, and the impact of plan failures in the experimental settings. The success of repair strategies also depends on these two factors. However, in the experiments discussed here we set these parameters to default values, and leave to future work the task of investigating the exact extent of these correlation.

References