Empirical Evaluation of a Replanning Algorithm

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

Abstract

In this paper, we present and discuss the empirical evaluation of a replanning algorithm for a utility-based, hierarchical planner. The algorithm is embedded in a BDI 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 the plan fails during execution.

Introduction

Although planning and reactive planning techniques have been often investigated in real world domains, by obtaining encouraging results, their evaluation in the context of intelligent 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.

In this paper, we present an experimental setting for empirically evaluating re-deliberation strategies in a BDI agent architecture which includes a meta-deliberation component (Boella and Damiano, 2002b), (Damiano, 2002). Theories of rational behavior claim that intentions are characterized by stability and tend to be revised at lower level rather than at higher level (Bratman, 1987), (Bratman et al., 1988). According to this claim, conservative re-deliberation should be preferred over forms of re-deliberation which abstract from current intentions.
Here, we focus our attention on 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. In line with theories of rational behavior, the functioning of the replanning component is inspired to the 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 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 three different domains.

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 (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 redeliberation 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 architecture is based on the assumption that 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, i.e., an action can result in a set of alternative effects (see (Boella and Damiano, 2002b) and (Damiano, 2002) for a detailed description). 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 (see Figure 1). When this loop is first entered, the deliberation module is invoked on the initial goal; the goal is matched against the plan schemata contained in the library, and when a plan schema is found, it is passed 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 effects of the action execution, 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, this function computes the expected utility of executing the remaining plan steps. Then, the difference between the previously expected utility

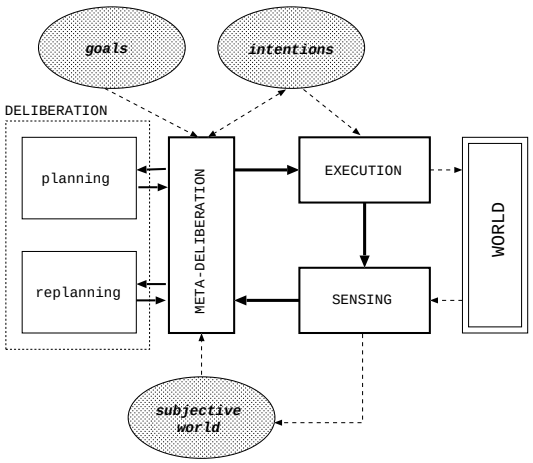

Figure 1: The structure of the agent architecture. Dashed lines represent data flow, solid lines represent control flow. The grey components determine the agent's state.

and the new one is computed: if there is a significant difference, the plan has failed during its execution, and replanning is performed.[1]

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. On the contrary, if new deliberation is necessary, the deliberation module invokes the *replanning component* on the current plan with the task of finding a better plan.

Deliberation and Execution

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 Suwandi, 1994), (Haddawy and Hanks, 1998). 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.

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. This property is a key one for the planning process as it makes it possible to compare partial plans which contain abstract actions.

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. 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.
At each refinement cycle the planning algorithm re-starts from a less partial plan, i.e., a plan that subsumes a smaller set of alternatives in the plan space: at the beginning this plan coincides with the topmost action which achieves the goal, in the subsequent refinement phases it is constituted by a sequence of 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.

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.

The key idea of the replanning algorithm is then to make the current plan more partial, until a more promising partial

[1] The higher bound (the maximal utility) of the new expected utility interval is significantly lower than the higher bound of the previous expected utility interval and the difference is above a certain (arbitrary) threshold.

```
procedure plan replan(plan p, world w)
begin
/* 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))
 begin
/* look for a partial plan with better EU */
  while (not (promising(p', w, p))
         and has-father(a))
   begin
    p' := partialize(p');
    project(p',w); //evaluate action in w
   end
/* restart planning on the partial plan */
 p'' := refine(p',w);
 end
 return p'';
end
```

Figure 2: The main procedure of the replanning algorithm, *replan*.

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 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 (see (Boella and Damiano, 2002a), (Boella and Damiano, 2002b) and (Damiano, 2002) for a detailed description of the algorithm).

Notice that, if a plan with a higher utility than the failed one exists, the replanning algorithm finds it: if a solution cannot be found locally, the partialization process finally invokes the refinement procedure on the action hierarchy root.

The task of identifying the next action whose preconditions do not hold (the 'focused action') is accomplished by the *find-focused-action* function (see the *replan* procedure in Figure 2). Then, starting from the focused action (FA), the replanning algorithm partializes the plan, following the derivation tree associated with the plan (see the *partializes* function in Figure 3).

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 4):

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 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 in the plan by the complex sequential action, which becomes the current FA (see Figure 4).[2]

```
function plan partialize(plan p)
begin /* a is the FA of p */
action a := marked-action(p);
/* if subsumed by a partial action */
 if (abstract(father(a)))
 begin
/* delete a from the tree */
   delete(a, p);
   return p;
 end
/* we are in a decomposition */
else if (complex(father(a))
 begin
  a1 := find-sibling(a,p);
  if (null(a1))
/* there is no FA in the decomposition */
    begin
      mark(father(a)) //set the FA
      //delete the decomposition
      delete(descendant(father(a)),p);
      return p;
    end
  else
    begin //change the current FA
     unmark(a);
     mark(a1);
    end
 end
end
```

Figure 3: The function for making a plan more abstract, *partialize*.

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, by discarding suboptimal alternatives.

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). In other words, the utility of p' may have an higher higher bound with respect to p, making p' a more promising plan than p. 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.

[2] Since an action type may occur in multiple decompositions, in order to understand which decomposition the action instance appears into, it is not sufficient to use the action type library, but it is necessary to use the derivation tree.

```
function action find-sibling(a,p)
begin
/* 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)))
  begin
/* if it can be partialized */
   if (not complex(father(a1)))
    begin
     unmark(a); //change FA
     mark(a1);
     return a1;
    end
/* move to next action */
   a0 := right-sibling(a0,p);
   a1 := leftmost(descendant(a0,p));
  end
/* 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)))
  begin
   if (not complex(father(a1)))
    begin
     unmark(a);
     mark(a1);
     return a1;
    end
  end
 action a1 := left-sibling(a,p);
end
```

Figure 4: The procedure for finding the new focused action.

Plan Execution

As described in the previous section, the agent's deliberation and meta-deliberation are based on its subjective representation of the world, maintained by the sensing component.

The representation of a world is constituted by a set of attribute-value pairs. In the objective representation of the world, all attributes have a certain value, while the subjective representation of the world, an attribute can have a probability distribution on its values. Associating attribute-value pairs with a probabilistic value is a way to represent a set of alternative worlds in a compact fashion: a probability distribution on the value of one or more attributes determines a probability distribution on a set of worlds.

Representing the agent beliefs about the world separately from the 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 updated by a simulator according to the messages 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.

Given the agent loop introduced before, the representation of the world contained in the simulation module can be altered between two subsequent sensing acts, by reproducing a multiplicity of real-world situations. By modifying the objective representation of the world between the execution of a step and the subsequent monitoring, it is possible to reproduce the situation in which the world changes immediately after the agent has executed an action, possibly altering the action effects. Or, by modifying the objective representation of the world between the agent's deliberation and execution, it is possible to model a situation where the world changes unexpectedly immediately after agent's deliberation, so that the execution of the selected course of action takes place in a different world than expected.

Empirical Evaluation

Experimental Methodology

In order to experimentally tests the issues expressed in the Introduction, we arranged two different experimental settings, that we applied to three 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. Due to a discrepancy by its subjective representation of the world and the world itself, as represented in the simulator, the expected utility of the plan devised by the agent is likely to drop during execution. However, the initial subjective and objective representation of the world are not assumed to be arbitrarily different: rather, the objective world representation is obtained from the subjective representation of the world fed to the agent's deliberation process.

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. By doing so, it is possible to obtain a subjective world representation characterized by uncertainty, which represents in a compact way a probability distribution on a set of worlds (see Section on Execution).

If the agent's subjective representation does not contain any uncertainty, then a random number of attributes in the subjective world (drawn from a predefined set) are altered, by

substituting their values in the subjective world with a new random value in the prescribed range. If the world is characterized by uncertainty, then the we exploit it 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).

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 expected utility drop.

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, to simulate execution failures and unexpected world changes. In practice, the objective word 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.

Experiment Scenarios

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 three different domains. These domains have been chosen for their availability, but differ along several dimensions: plan library complexity (maximal and minimal plan length, action hierarchy depth), non-determinism degrees in actions types, world representation complexity (number of attributes involved in world definition 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) represents an office toy world and concerns the planning of a mail delivery task. It is drastically less complex than the previous ones for what concerns the depth of the abstraction and decomposition hierarchy, actions are prominently non deterministic and the initial world definition is characterized by a lower number of defining attributes (Boella and Damiano, 2002b), (Boella and Damiano, 2002a).

The third domain (C) concerns medical diagnosis and therapy (Haddawy et al., 1996); it is characterized by high level of uncertainty in initial world and an high level of non-determinism in action definition. Plans have a minimal length of 2 steps, and a maximal length of 6 steps. The action hierarchy is quite complex, and includes several abstraction and decomposition levels.

Discussion of the Data

In the following, we give a sketch of the data in the three 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.[3]

For the evaluation of each set of experiments, we considered the following 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 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.[4]
  In order to verify the effectiveness of the replanning algorithm we measure the 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

[3] The exact number of items contained in each set of experiments is the following: domain A, setting 1: 400 items; domain A, setting 2: 400 items; domain B, setting 1: 100 items; domain B, setting 2: 100 items; domain C, setting 1: 200 items; domain C, setting 2: 300 items. We leave to future work the assessment of the statically appropriate test set size for each domain.

[4] In the experiments, we obtained the threshold by applying a fixed ratio of $0,1$ to the upper bound of the initially expected utility.

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 the plans obtained by planning from scratch.

The empirical evaluation conducted in the three domains (see Figures 5 and 6) seems to suggest that the domain constitutes a important factor in determining both the relevance of replanning and the effectiveness and efficiency of the redeliberation strategies we tested (replanning and planning from scratch). For this reason, we will discuss the data grouped by the three domains A, B, and C.

In domain A, the replanning rate is quite low if compared to the other two domains (26% in setting 1 and $38,5\%$ in setting 2), and the successful replanning rate is, respectively, of $18,44\%$ and 19% in the two settings; this value reduces to 5% and $6,75\%$ if we consider the absolute successful replanning rate (see Figure 5, 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 6, rows $A1$ and $A2$). The average replanning time is $8,68\%$ of the planning time in setting 1, and $0,6\%$ in setting 2. The average expected utility of the plans obtained by planning from scratch is $69,13\%$ of the average expected utility of the plans obtained by replanning in setting 1, and $51,87\%$ in setting 2.

In domain B, we observe a replanning rate of $20,5\%$ in setting 1 and $30,33\%$ in setting 2, and a replanning success rate of 100% in setting 1 (the replanning process always succeeded) and of $32,4\%$ in setting 2 (see Figure 5, rows $B1$ and $B2$). For what concerns efficiency and effectiveness of replanning (see Figure 6, rows $B1$ and $B2$), in both setting the successful replanning, on average, took longer than planning from scratch (with a value of $140,4\%$ $137,95\%$ respectively). The average expected utility of the replanned plans is the same as the average expected utility of the plans obtained by planning from scratch.[5]

In domain C, the replanning rate is the highest, with a 90% replanning in the two settings. However, a poor performance of the replanning algorithm corresponds to the high relevance of plan failures (and the consequent replanning attempts) in this domain (with a replanning success rate of 0% in setting 1, and of $4,44\%$ in setting 2, see Figure 5, rows $C1$ and $C2$). Consequently, no data are available about replanning efficiency and effectiveness in setting 1, and very few are available in setting 2 (the an average replanning time is $152,46\%$ of the average time needed for planning from scratch, and the average expected utility value obtained by planning from scratch is equal to the value obtained by replanning, see Figure 6, rows $C1$ and $C2$).

Given the preliminary data presented above, we believe that they express a correlation between the planning domain and the performance of the re-deliberation strategies. For what concerns the performance of the replanning algorithm, in particular, data show that it constitutes by far the best approach in domain A, while planning from scratch may be preferable in domains B and C. However, in none of the three domains the performance of the replanning algorithm is worse than the planning from scratch strategy by both processing time *and* quality of the output plans. Regarding processing time, the ratio of average time performance of the replanning algorithm on the average time performance of the planning time is never over $1,5$.

Moreover, for what concerns the replanning success rate, its efficiency (replanning time), and effectiveness (expected utility of the plans), a discussion of the results cannot abstract from the peculiarities of each domain. In particular, while the replanning algorithm is based on the assumption that it is more advantageous to look for a local, non-optimal solution instead of climbing the entire action hierarchy in the replanning process, the limited depth of the action hierarchy in domain B makes this hypothesis invalid: this consideration partly explains the performance of the replanning algorithm in this domain.
Regarding domain C, it is necessary to notice that the experiments conducted in this domain lead to an initial plan of two-steps in almost the 90% of the cases: in this domain, the method that we applied for the generation of initial worlds has proven to be inappropriate, as it heavily depends on the correct assessment of initial value ranges (see Section on Experimental Methodology). Unfortunately, the impossibility of finding an appropriate combination of value ranges for the generation of the initial worlds has resulted in a large majority of "trivial plans", which do not constitute an appropriate test bed for the simulation of plan failures during execution. Given these observations, the replanning algorithm we tested appears to perform best in domain A, characterized by a relatively high action hierarchy and a fair level of nondeterminism. The experiments performed in this domain show a preferability of replanning on planning from scratch, by both efficiency and effectiveness.

For what concerns significance of plan failures during execution, several factors seem to influence this value. On the one side, the agent deliberation and meta-deliberation parameters (utility function, plan failure ratio, etc.) contribute to determine the number of plan failures in a certain domains, together with the domain intrinsic features. On the other side, the experiment setting also appears to be relevant to determine the failure rate, although the data collected by using setting 1 and 2 seem to be comparable. In order to assess the relevance of each of factors mentioned above, further experiments are needed. For example, for what concerns the deliberation and meta-deliberation parameters, different plan failure ratios and different utility functions should be tested. For what concerns experiments settings, all experimental parameters should be varied in order to assess their correlation with the results.

[5]The reason for the coincidence of the expected utility values is that, in this domain, the replanning process outputs the same plans as the planning from scratch process.

DOMAIN	REPLANNING RATE	SUCCESSFUL REPLANNING % (relative)	SUCCESSFUL REPLANNING % (absolute)
A 1	26	18,44	5
B 1	20,5	100	20,5
C 1	90	0	0
A 2	38,5	17,995	6,75
B 2	30,33	32,4	9,66
C 3	90	4,44	4

Figure 5: The replanning rate in each domain-setting pair (first column) and the successful replanning rate, relative (middle column) and absolute (right column).

DOMAIN	AVG REPLANNING TIME (% on avg time of planning from scratch)
A1	8,68
B 1	140,4
C 1	-
A 2	0,6
B 2	137,95
C 2	152,46

DOMAIN	AVG EU BY PLANNING FROM THE SCRATCH (% on avg. EU by replanning)
A 1	69,13
B 1	100
C 1	-
A 2	51,87
B 2	100
C 2	100

Figure 6: 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).

Conclusions and Future Work

Although the data collected by preliminary experiments point out the need for more extended and more detailed experiments, we believe that the results we collected lead 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. Although there are intrinsic correlations between the planning domain, the relevance of replanning and the re-deliberation strategies, simulations certainly help point out the exact nature of these correlations. For example, the empirical evaluation we conducted seems to suggest that the replanning algorithm performs well when on complex plan libraries, if compared with the strategy of replanning from scratch: however, to test this claim, it is necessary to ensure that the initial plans generated along the experiments have a sufficient degree of complexity.

Investigating the interleaving of execution and re-deliberation is another issue which deserves future work. Preliminary studies, in fact, show that the agent architecture we used to carry out the experiments presented here is suitable for investigating this issue, as it provides an appropriate framework for the interleaving of execution and re-deliberation.

Finally, other reasons, external to the replanning time and plan quality considerations, may be relevant in certain contexts, like cooperative and interactive context: in these contexts, the rational properties of intentionality may set an a priori preference for conservative replanning solutions, making the issue of performance even more relevant.

References

Boella, G. and Damiano, R. (2002a). An architecture for normative reactive agents. In Kawabara, K. and Lee, J., editors, *Intelligent Agents and Multi-Agent Systems (Proc. of Prima 02), LNAI 2413*, pages 1–17, Tokyo.

Boella, G. and Damiano, R. (2002b). A replanning algorithm for a reactive agent architecture. In Scott, D., editor, *Artificial Intelligence: Methodology, Systems, and Applications (Proc. of Aimsa 02), LNCS 2443*, pages 183–192, Varna.

Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge (MA).

Bratman, M. E., Israel, D. J., and Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355.

Damiano, R. (2002). *The Role of Norms in Intelligent Reactive Agents*. Ph.d. thesis, Universitá di Torino, Torino, Italy.

Haddawy, P., Doan, A., and Kahn, J. (1996). Decision-theoretic refinement planning in medical decision making: Management of acute deep venous thrombosis. *Medical Decision Making*.

Haddawy, P. and Hanks, S. (1998). Utility models for goal-directed, decision-theoretic planners. *Computational Intelligence*, 14:392–429.

Haddawy, P. and Suwandi, M. (1994). Decision-theoretic refinement planning using inheritance abstraction. In *Proc. of 2nd AIPS Int. Conf.*, pages 266–271, Menlo Park, CA.

Wooldridge, M. and Parsons, S. (1999). Intention reconsideration reconsidered. In Müller, J., Singh, M. P., and Rao, A. S., editors, *Proc. of ATAL-98)*, volume 1555, pages 63–80. Springer-Verlag.