A Distributed Control Loop for Autonomous Recovery in a Multi-Agent Plan

Roberto Micalizio
Dipartimento di Informatica
Università di Torino
micalizio@di.unito.it

Abstract

This paper considers the execution of a Multi-Agent Plan in a partially observable environment, and faces the problem of recovering from action failures.
The paper formalizes a local plan repair strategy, where each agent in the system is responsible for controlling (monitoring and diagnosing) the actions it executes, and for autonomously repairing its own plan when an action failure is detected.
The paper describes also how to mitigate the impact of an action failure on the plans of other agents when the local recovery strategy fails.

1 Introduction

Many real complex tasks find proper solutions in the adoption of a Multi-Agent Plan (MAP), where a team of agents cooperate to reach a complex goal G by performing actions concurrently.
The execution of a MAP, however, is a critical phase as *plan threats* (e.g., agent faults in [Birnbaum *et al.*, 1990]) can disrupt the nominal progress of the plan by causing the failure of some actions. The occurrence of a plan threat does not prevent, in general, the agents to complete their activities, but the MAP needs to be repaired; i.e., a new planning process is required to overcome the effects of the action failure, so that the global goal can be achieved in some alternative ways.

Wielding action failures in a multi-agent setting is particularly challenging. First of all, the failure of an action must be detected as soon as possible. Moreover, since the agents cooperate by exchanging services, the local failure of an agent can easily propagate in the global MAP: the services the agent in trouble can no longer provide are preconditions for the actions of other agents. If the local failure is not properly handled, these agents may fall in a *stalling condition* waiting indefinitely for services that will never be provided. Finally, even though the impaired agent does not provide services to other agents, it may represent a latent menace because it may lock critical resources.

To cope with these issues, the paper proposes a distributed approach to autonomous plan repair, where each agent performs a *closed control loop* over the actions in its local plan; this control loop allows the agents in the team to autonomously handle plan threats by overcoming (when possible) their harmful effects. Three main tasks are included in the control loop: the *plan monitoring*, the *agent diagnosis*, and the *plan repair* (consisting of a re-planning step driven by the agent diagnosis). However, due to the partial observability of the system, establishing the control loop is a complex task: the monitoring can just estimate the agent state as a set of alternatives (i.e., a belief state), and the agent diagnosis is typically ambiguous (i.e., a set of alternative explanations); as a consequence the re-planning step must be able to deal with uncertainty.

The paper is organized as follows: in section 2 basic notions on multi-agent plans are introduced; in sections 3 and 4 two steps of the control loop, the monitoring and the diagnosis respectively, are formalized; in section 5 is presented the plan repair methodology, which mainly focuses on what goals must be reached to repair a local plan. The effectiveness of the repair methodology is discussed in section 6 where some experimental results are presented. Finally, in section 7 the proposed approach is compared with other relevant works in literature.

2 Background

Global Plan. The paper considers MAPs where the agents of a team $\mathcal{T}$ actively cooperate for reaching a complex goal G. For the sake of discussion, the model of a MAP is a simplified version of the formalism presented in [Cox *et al.*, 2005]. In particular, the MAP P is the tuple $\langle A, E, CL, RE \rangle$, where:
- A is the set of the action instances the agents have to execute. Two pseudo-actions, a_0 and a_∞, belong to A; a_0 (the starting action) has no preconditions and its effects specify the propositions that are initially true; a_∞ (the ending action) has no effects and its preconditions specify the propositions which must hold in the final state i.e., the goal G of the MAP. Except a_0 and a_∞, each action instance $a \in A$ is assigned to a specific agent $i \in \mathcal{T}$.
- E is a set of precedence links between actions: a precedence link $a \prec a'$ in E indicates that the execution of a must precede the execution of a';
- CL is a set of causal links of the form $cl : a \xrightarrow{q} a'$; the link cl states that the action a provides the action a' with the service q (q is an atom occurring in the preconditions of a');
- RE is a set of precedence links ruling the access to the re-

sources. In fact, according to the *concurrency requirement* introduced in [Roos and Witteveen, 2007], two actions a and a', assigned to different agents, cannot be executed at the same time instant if they require the same resource res, therefore, the planning process puts in the plan a precedence link between them, either $a \prec_{res} a'$ or $a' \prec_{res} a$; these precedence links are labeled with the identifier of the specific resource the precedence link refers to, and are collected in the set RE.
Local Plans. The MAP P is decomposed into as many local plans as the agents in the team, and each local plan P^i is assigned to an agent i, and reaches a specific sub-goal G^i.
Formally, the local plan for agent i is the tuple
$P^i = \langle A^i, E^i, CL^i, T^i_{in}, T^i_{out}, RE^i_{in}, RE^i_{out} \rangle$ where A^i, E^i and CL^i have the same meaning of the sets A, E and CL, respectively, restricted to actions assigned to agent i. A^i includes also two special actions a^i_0 and a^i_∞ which specify, respectively, the initial and final conditions for the sub-plan P^i. T^i_{in} (T^i_{out}) is a set of incoming (outgoing) causal links of the form $a \xrightarrow{q} a'$ where a' (a) belongs to A^i and a (a') is assigned to another agent j in the team. Similarly, RE^i_{in} (RE^i_{out}) is a set of incoming (outgoing) precedence links of the form $a \prec_{res} a'$ where a' (a) belongs to A^i and a (a') is assigned to another agent j in the team.

We assume that each local plan P^i is *totally ordered*, that is P^i is the ordered sequence of actions $[a^i_0, a^i_1, \dots, a^i_\infty]$.
Distributed plan execution. An agent executes its next action as soon as the action preconditions have been satisfied (the notions of preconditions and effects of an action will be formalized in the following section); however an agent can execute no more than one action in a given time instant. In particular, the time is assumed to be a discrete sequence of instants, and actions are atomic.
In the following the notation $a^i_l(t)$ will denote that the l-th action in the local plan P^i is executed by agent i at time t.
Coordination during plan execution. Since agents execute actions concurrently, they need to coordinate their activities in order to avoid the violation of the constraints defined during the planning phase. Effective coordination among agents is obtained by exploiting the causal and precedence links in the global MAP.
As pointed out in [Decker and Li, 2000], coordination between agents i and j is required when i provides j with a service q; this is modeled by a causal link $cl : a^i_h \xrightarrow{q} a^j_k$ in the MAP P. (As an effect of the MAP decomposition, cl belongs both to T^i_{out} and to T^j_{in}.) Since an agent can observe (at most) the direct effects of the actions it executes, only agent i has the chance of observing the achievement (or the absence) of q; thereby, the agent i must notify agent j about the outcome of action a^i_h.
Similarly, the consistent access to the resources is a form of coordination which involves precedence links. For example, the precedence link $pl : a^i_h \prec_{res} a^j_k$ means that agent i will release resource res to agent j just after the execution of action a^i_h; resource res will be used by agent j to execute action a^j_k. Of course, pl belongs both to RE^i_{out} and to RE^j_{in}.

Since the system is distributed, an agent does not have a global view of the status of all system resources, but it knows just the status of the resources it holds. After having released the resource res, the agent i will not have access to the actual status of res. In the following we will denote as $AvRes(i, t)$ (available resources) the subset of resources assigned to agent i at time t; i.e., only agent i observes and knows the actual status of those resources.

3 Monitoring a MAP

In this section we formalize the first step of the local control loop: the plan monitoring activity. Before that, however, we first introduce some fundamental notions.
Agent status. The status of agent i is modeled by a set of status variables VAR^i, partitioned into three subsets END^i, ENV^i and HLT^i. END^i and ENV^i denote the set of endogenous (e.g., the agent's *position*) and environment (e.g., the resources state) status variables, respectively. Because of the partitioning, each agent i maintains a private copy of the resource status variables; therefore for each resource $res_k \in RES$ ($k : 1..|RES|$) the private variable $res_{k,i}$ is included in the set ENV^i. The precedence links in RE guarantee that, at each time t, a resource res_k is available just for an agent i (i.e., res_k belongs to $AvRes(i, t)$); therefore for any other agent $j \in \mathcal{T} \setminus \{i\}$ the status of the res_k is *unknown*, and we do not need to check the consistency among the private copies of the resource variables.

Since we are interested in monitoring the plan execution even when something goes wrong, we introduce a further set HLT^i of variables for modeling the health status of an agent. For each agent functionality f, a variable $v_f \in HLT^i$ represents the health status of f, the domain of variable v_f is the set $\{ok, abn_1, \dots, abn_n\}$ where ok denotes the nominal mode, while $abn_1, \dots, abn_n$ denote anomalous or degraded modes. An action failure can be therefore explained in terms of faults in a subset of functionalities of a specific agent.
System observability. We assume that after the execution of an action $a^i_l(t)$ the agent i receives a set $obs^i(t+1)$ of observations, that conveys information about a subset of variables in VAR^i. Given the partial observability, an agent can directly observe just the status of its available resources, and the value of a subset of variables in END^i, whereas the variables in HLT^i are not directly observable and their actual value can be just inferred. As a consequence, at each time t the agent i can just estimate a *set* of alternative states which are consistent with the received observations; in literature this set is known as *belief state*, and in the following the notation $\mathcal{B}^i(t)$ will refer to the belief of agent i inferred at time t.
Action models. In order to monitor the execution of action $a^i_l(t)$, agent i needs a model for estimating all the possible, nominal as well as anomalous, evolutions of the action itself. An action model is the tuple $\langle var(a^i_l(t)), PRE(a^i_l(t)), EFF(a^i_l(t)), \Delta(a^i_l(t)) \rangle$, where: $var(a^i_l(t)) \subseteq VAR^i$ is the subset of *active* status variables over which the preconditions $PRE(a^i_l(t))$ and the action effects $EFF(a^i_l(t))$ are defined; finally, $\Delta(a^i_l(t))$ is a transition relation defined over the agent status variables from time t (when the action starts) to time t+1 (when the action ends). Given action a^i_l, $healthVar(a^i_l) = HLT^i \cap var(a^i_l)$ denotes the set of variables representing the health status of the functionalities which di-

rectly affect the outcome of action a_l^i.
The healthy formula *healthy*(a_l^i) of action a_l^i is computed by restricting each variable $v \in$ *healthVar*(a_l^i) to the nominal behavioral mode *ok* and represents the nominal health status required to agent i for successfully completing the action.

Definition 1 *The set of the nominal effects of action* a_l^i *is nomEff*(a_l^i)=$\{q \in$ *EFF*$(a_l^i) \mid$ *PRE*$(a_l^i) \cup$ *healthy*$(a_l^i) \vdash q\}$.

On the contrary, when at least one variable $v \in$ *healthVar*(a_l^i) assumes an anomalous mode (i.e., a functionality is not in the nominal mode), the behavior of the action may be non deterministic and some of the expected effects may be missing.
In the following, $\mathcal{A}$ will denote the set of action models an agent exploits for monitoring the progress of its own plan.
The estimation of the agent status. The estimation process aims at predicting the status of agent i at time $t+1$ after the execution of an action a_l^i(t); however, because of the non determinism in the action model and the partial system observability, the estimation process can in general infer just a set of alternative agent states (i.e.; a belief state) rather than the actual agent state. The estimation can be formalized in terms of the Relational Algebra operators as follows.

Definition 2 *Let* $\mathcal{B}^i(t)$ *be the belief state of agent i, let* $\Delta(a_l^i(t))$ *the model of the action executed at time t, the agent belief state at time* $t+1$ *results from:* $\mathcal{B}^i(t+1)$ =
$\text{PROJECTION}_{t+1}\ \text{SELECTION}_{obs^i(t+1)}\ (\mathcal{B}^i(t)\ \text{ JOIN }\ \Delta(a_l^i(t)))$

The join operation $\mathcal{B}^i(t)\ \text{ JOIN }\ \Delta(a_l^i(t))$ is the predictive step by means of which all the possible agent states at time $t+1$ are estimated. The selection $\text{SELECTION}_{obs^i(t+1)}$, refines the predictions by pruning off all those estimates which are inconsistent with the agent observations. Finally, the belief state $\mathcal{B}^i(t+1)$ is obtained by projecting the resulting estimates over the status variables of agent i at time $t+1$.
Action outcome. The outcome of an action is either *succeeded* or *failed*; given the belief $\mathcal{B}^i(t+1)$, the agent i determines the successful completion of action $a_l^i(t)$ as follows:

Definition 3 *The outcome of action* $a_l^i(t)$*, is* succeeded *iff* $\forall q \in nomEff(a_l^i(t))$, $\forall s \in \mathcal{B}^i(t+1)$, $s \models q$.

In order to be conservative, we consider action $a_l^i(t)$ successfully completed only when all the atoms q in $nomEff(a_l^i(t))$ are satisfied in every state s in $\mathcal{B}^i(t+1)$; i.e., when all the nominal effects of $a_l^i(t)$ hold in every possible state estimated after the execution of the action. When we cannot assert that action $a_l^i(t)$ is succeeded, we assume that the action is failed.

4 Agent Diagnosis

In this section we formalize the step of *agent diagnosis*, which explain the action failure in terms of faults in the agent functionalities. As a further refinement of the failure analysis, we show also how to compute the set of *missing goals*, which in principle steers the recovery process.
Agent diagnosis. The diagnostic process is activated whenever the agent i detects the failure of an action $a_l^i(t)$. The purpose of the diagnostic process is to single out which fault (or combination of faults) is a possible cause (i.e., an explanation) for the detected failure. An explanation is therefore expressed in terms of the status variables in *healthVar*(a_l^i), as these variables model the health status of the functionalities required for the successful execution of action a_l^i.

Intuitively, given the agent belief state $\mathcal{B}^i(t+1)$, the agent diagnosis D^i is inferred by projecting $\mathcal{B}^i(t+1)$ over the status variables in *healthVar*(a_l^i). However, since $\mathcal{B}^i(t+1)$ is in general ambiguous, the agent diagnosis D^i results to be a set of alternative explanations: each explanation $exp \in D^i$ is a complete assignment of values to the status variables in *healthVar*(a_l^i).
Missing Goals. As noted earlier, the agents in the team $\mathcal{T}$ cooperate one another by exchanging services; that is, there exist causal dependencies between actions of different agents. As a consequence, the failure of action a_l^i prevents the execution of the actions in the plan segment $[a_{l+1}^i, a_\infty^i]$ and, since some services will never be provided, it can indirectly impact the local plans of the other teammates.
The set of the services that agent i can no longer provide due to the failure is denoted as the set of *missing goals*; singling out these services is important as, in principle, it would be sufficient to find an alternative way to provide them in order to reach the global goal G despite the failure of action $a_l^i(t)$.

Definition 4 *Given the failure of action* a_l^i*, let* $[a_{l+1}^i, \dots, a_\infty^i]$ *be the plan segment the agent i is unable to complete, the set of missing goals is:* $\mathcal{MG}(i)$=$\{$ *service* $q|$
$\forall a_k^i \in [a_l^i, \dots, a_\infty^i]$, $q \in nomEff(a_k^i)$*, and*
$q \in PRE(a_\infty^i)$ *or*
$\exists$ *a causal link* $cl \in CL$ *such that* $cl : a_k^i \xrightarrow{q} a_h^j; i \neq j)\}$

Namely, the service q is a missing goal when q is a nominal effect no longer provided by an action in the plan segment $[a_l^i, \dots, a_\infty^i]$, and when either q is an atom appearing in the sub-goal G^i (i.e., q is a precondition of the special action a_∞^i) or q is a service agent i should provide to another agent j.

5 Plan Repair: a local strategy

In this section we discuss a methodology for repairing the local plan P^i, interrupted after the failure of action a_l^i has been detected. Essentially, this repairing process consists in a replanning step intended to overcome, if possible, the harmful effects of the failure. Due to lack of space the planner activated by agent i is not described; it is sufficient to say that this planner is based on a forward-chaining strategy, and it is similar to the conformant planners described in [Cimatti and Roveri, 2000; Micalizio and Torasso, 2007] (in the following we will point out why this planner must be conformant).

Instead of describing the planner, in this section we focus on which goals should be reached for the recovering purpose. As noted above, the set of missing goals can be used to this end; unfortunately, when the recovery is driven by the missing goals it requires global changes; in fact, the missing goals are long term objectives, that can be reached by acquiring new resources; the acquisition of a resource, however, imposes the coordination with other teammates, and hence new causal and precedence links are to be introduced in the global MAP P; it follows that a number of other agents in the team have to change their local plans.

The local strategy we propose, instead, try to recover from the failure of a^i_l just by changing the local plan P^i, without any direct impact on the plans of other teammates. The idea of a local strategy stems from the observation that in many cases an agent is still able to do something useful even if its health status is not completely nominal. By exploiting this possibility, we first formalize a local replanning strategy intended to overcome the causes of an action failure; however, when such a replanning step fails, we show also how the agent in trouble can reduce the impact of the failure in the MAP P by moving into a safe status.

Repairing the interrupted local plan. This step is based on the observation that the plan segment $[a^i_{l+1}, \ldots, a^i_\infty]$ could be carried out if the root causes of the failure of action a^i_l were removed.

Of course these root causes have been singled out by the diagnostic inferences: the agent diagnosis D^i explains the failure of a^i_l as a combination of anomalous conditions in the functionalities of agent i; therefore, to overcome the causes of this failure the agent i has to restore a healthy condition in those functionalities. To this end the agent i can exploit a set $\mathcal{A}^R$ of *repairing actions*, each action $ar \in \mathcal{A}^R$ restore the healthy condition in a specific agent functionality; however, it is possible that for some functionalities no repairing action exists. For example, if our agents were robots, a low charge in the battery could be fixed by means of a `recharge` repairing action, whereas a fault in the mobility functionality would require the human intervention and it could not be fixed autonomously by the robots.

Therefore, relying on the agent diagnosis D^i, the agent i assesses whether one (or a set of) repair action(s) exists. If recover actions do not exist, agent i gives up the synthesis of a recovery plan and tries to reach a safe status (see later). If recovery actions exist, the agent i tries to reach the new high-level goal $\mathcal{K}$ consisting in: 1) restoring the healthy conditions in its functionalities by executing an appropriate set of repairing actions, *and* 2) restarting the execution of the plan from the failed action a^i_l. The repairing plan Pr^i is a plan which meets these two goals, and it can be found by resolving the following planning problem:

Definition 5 *The repairing plan* Pr^i=$[ar^i_0, \ldots, ar^i_\infty]$ *is a solution of the plan problem* $\langle \mathcal{I}, \mathcal{F}, \overline{\mathcal{A}} \rangle$*; where:*

- $\mathcal{I}$ *(initial state) corresponds to the agent belief state:* $EFF(ar^i_0) \equiv \mathcal{B}^i(t+1)$ *(i.e., the belief state inferred after the execution of action* $a^i_l(t)$*).*

- $\mathcal{F}$ *(final state) is the goal* $\mathcal{K}$ *defined as*
$PRE(ar^i_\infty) \equiv \{\forall v \in healthVar(a^i_l), v = ok\} \wedge PRE(a^i_l)$

- $\overline{\mathcal{A}} = \mathcal{A} \cup \mathcal{A}^R$ *is the set of action models which can be used during the planning process.*

The repairing plan Pr^i, however, must satisfy two further demanding requirements:

Requirement 1 *Since the repairing plan* Pr^i *can impose local changes only, no new resources can be acquired: the actions in* Pr^i *can just exploit the resources in* $AvRes(i,t)$*, already acquired by agent i at the time of the failure.*

Requirement 2 *Since the belief state* $\mathcal{B}^i(t+1)$ *is potentially ambiguous (the actual agent health status is not precisely known) the repairing plan* Pr^i *must be conformant, namely, it must be executable no matter the actual health status of agent i.*

An important consequence of the conformant requirement is the following property.

Property 1 *For each action* $ar^i_k \in Pr^i$ *it must hold* $healthy(ar^i_k) \cup D^i \not\vdash \perp$

Property 1 states that all the actions in the repairing plan must be executable despite the current status of agent i is not healthy; therefore, when no action is executable given the agent diagnosis D^i, the repairing plan does not exist.

Assuming that the plan Pr^i exists, the agent i yields its new local plan P^{*i} = $[ar^i_0, \ldots, ar^i_\infty] \circ [a^i_l, \ldots, a^i_\infty]$; where $\circ$ denotes the concatenation between two plans (i.e., the second plan can be executed just after the last action of the first plan has been executed).

Property 2 *The recovery plan* P^{*i} *is feasible and executable.*

Due to space reason the proof is omitted, intuitively the feasibility of the recovery plan P^{*i} stems by two characteristics: 1) every plan segment is feasible on its own as it has been produced by a specific planning step, and 2) the preconditions of the action ar^i_∞ of the first plan matches with the effects of the action a^i_l of the second one. A more important property is the following one:

Property 3 *The recovery plan* P^{*i} *meets all the services in* $\mathcal{MG}(i)$.

Property 3 guarantees that, by executing P^{*i} in lieu of $[a^i_l, \ldots, a^i_\infty]$, agent i can recover from the failure of action a^i_l and achieve its sub-goal G^i despite the failure.

Reaching the Safe Status. The repairing plan Pr^i, however, may not exist; in fact the faults occurred to the functionalities may be not repairable, or a conformant solution which guarantees to be executable given the ambiguous agent diagnosis may not exist. In any case, when the plan recovery process fails, the impaired agent can be seen as a latent menace for the other team members (e.g., when the agent locks indefinitely critical resources). We complement the first step of the local strategy by means of a further step intended to lead the agent i into a *safe status* $\mathcal{S}^i$; i.e., a condition where all the resources used by i at time t (the time of the failure of action $a^i_l(t)$) have been released. Also this step can be modeled as a planning problem as follows:

Definition 6 *The plan-to-safe-status* Ps^i=$[as^i_0, \ldots, as^i_\infty]$ *is a solution of the plan problem* $\langle \mathcal{I}, \mathcal{F}, \mathcal{A} \rangle$*; where:*

- $\mathcal{I}$ *(initial state) corresponds to the agent belief state:* $EFF(as^i_0) \equiv \mathcal{B}^i(t+1)$ *(i.e., the belief state inferred after the execution of action* $a^i_l(t)$*).*

- $\mathcal{F}$ *(final state) is the safe status* $\mathcal{S}^i$*, defined as*
$PRE(as^i_\infty) \equiv \forall res_k \in AvRes(i,t), res_{k,i} = free.$

- $\overline{\mathcal{A}}$ *is the set of action models which can be used during the planning process.*

Of course, also the plan-to-safe-status Ps^i must satisfy the requirements 1 and 2; thus property 1 can be extended to the actions in Ps^i too. Repairing actions can be used also during this planning step, in some cases in fact it is required to

```
LocalControLoop($P^i$, $\mathcal{B}^i(0)$){ $t$=0;
  while there are actions in $P^i$ to be executed {
    $a^i_l$=nextAction($P^i$);
    if $PRE(a^i_l)$ are satisfied in $\mathcal{B}^i(t)$ {
      ⟨ EXECUTE $a^i_l$⟩;
      gather observations $obs^i(t+1)$;
      $\mathcal{B}^i(t+1)$ = Monitoring($\mathcal{B}^i(t)$, $\Delta(a^i_l)$)
      if outcome($a^i_l, \mathcal{B}^i(t+1)$) equals failed {
        $D^i$ = Infer-Diagnosis($\mathcal{B}^i(t+1)$, $healthVar(a^i_l)$)
        $Pr^i$ = Conformant-Planner($\mathcal{B}^i(t+1)$, $\mathcal{K}$, $\overline{\mathcal{A}}$)
        if $Pr^i$ is not empty → $P^i$=$Pr^i \circ [a^i_l, .., a^i_\infty]$
        else { $Ps^i$ = Conformant-Planner($\mathcal{B}^i(t)$, $\mathcal{S}$, $\overline{\mathcal{A}}$)
          if $Ps^i$ is not empty → $P^i$=$Ps^i$
          else invoke a global recovery strategy } } }
  t=t+1; } }
```

Figure 1: The control loop algorithm.

restore the healthy status in some functionalities to release a resource.

When the plan-to-safe-status Ps^i exists, it becomes the new local plan assigned to agent i; that is $P^{*i} = Ps^i$, and all the actions in $[a^i_l, \ldots, a^i_\infty]$ are aborted. Therefore, even though the agent i is unable to reach its goal G^i, it moves into safe status in order to do not obstruct the other team members in their activities.

When the recovery strategy fails. Both steps of the recovery strategy described above require a local re-planning step, which fail when the sought plan does not exist. In this case, we adopt a conservative policy and we impose that the impaired agent gives up the execution of its local plan. Performing further actions, in fact, may lead the agent in dangerous conditions; for example, the agent could lock indefinitely some resources preventing others to access them.

The failure of the local recovery strategy does not imply, in general, that the action failure cannot be recovered from, but different, global strategies should be activated. These strategies (out the scope of this paper), are driven by the set of missing goals, and may require the cooperation of a subset of agents, or the activation of a global re-planning step.

The algorithm. The high-level algorithm of the control loop performed by each agent $i \in \mathcal{T}$ is showed in Figure 1. The algorithm consists in a while cycle, where at each iteration the agent i singles out the next action a^i_l to be executed. The action a^i_l is executed iff its preconditions are satisfied in the current belief state $\mathcal{B}^i(t)$. After the execution, the agent i gathers the available observations and detects the outcome of a^i_l (see Definition 3). In case the action outcome is *failed*, first the diagnostic inference are activated, then the conformant planner is invoked to find a repairing plan (Definition 5), or alternatively a plan-to-safe-status (Definition 6). When both the planning steps fail, the agent i sends a message to all the other agents about its failure and interrupts the execution of its local plan P^i.

6 Experimental results.

The proposed control loop has been implemented in Java JDK 1.6 by exploiting the symbolic formalism of the Ordered Binary Decision Diagrams (OBDDs) to encode the agents' belief states, and the non deterministic models of the actions. Monitoring, diagnosis and planning are therefore implemented in terms of standard OBDDs operators (see [Micalizio and Torasso, 2007; Cimatti and Roveri, 2000; Jensen and Veloso, 2000]). Agents are threads running on the same PC (Intel Core 2, 2.16GHz, RAM 2GB, WindowsXP).

In our experiments we have (software) simulated a service-robot scenario where a team of robotic agents offer a "mail delivery service" in an office-like environment. Resources are parcels, clerks' desks, doors, and one or more repositories. Resources are constrained: desks, doors and repositories can be accessed by only one agent per time; moreover, at most one parcel can be put on a desk. The environment we have simulated is fairly large involving 30 critical resources. In such an environment we have considered the execution of 15 MAPs, involving 6 agents, each of which executes 10 actions and has to reach 2 sub-goals (i.e., the complex goal G consists of 12 sub-goals). The execution of each MAP has been perturbed by the injection of a fault in the functionalities of one agent.

In order to prove the effectiveness of the local recovery strategy, we have compared the execution of the 15 MAPs in four scenarios: *no-repair*, the agent in trouble does not handle the failure (the agent just stops the execution of its local plan); *safe-status*, whenever an action failure occurs, the agent in trouble moves into a safe status; *repair*, the agent tries to repair its own plan, when the repair process fails the agent stops the plan execution; *repair+safe-status*, a combination of the previous scenarios: first the agent tries to repair its plan, in case this step fails the agent reaches a safe status.

The collected results are reported in Table 1. From these results it emerges that the local control loop is performed very efficiently being in the order of hundreds of milliseconds even when the replanning step is performed. As expected, the (average) computational time for monitoring and diagnosing a local plan is independent from the adopted repair policy. Whereas the computational time for plan repair is a little more expensive in the *repair+safestatus* scenario as, in some cases, the re-planner is invoked twice. More important, the effectiveness of the local control loop is demonstrated by the number of performed actions and by the percentage of reached sub-goals of the whole MAP. In *no-repair* scenario the impaired agent is two times harmful for the whole system: one because it does not provide some services to other teammates, and one because it may lock some system resources; as a consequence, just one-third of the sub-goals is reached. In the *safe-status* scenario it is evident how an agent can reduce the impact of a local failure simply by releasing all the resources it holds; while in the *repair* scenario it emerges the effectiveness of a repair strategy. Of course, the best results are obtained in the *repair+safe-status* scenario, where the two methodologies are combined. Note that, even in this last scenario one-third of sub-goals are not reachable, this is due to the fact that in some cases the failures are not repairable.

7 Related works

In this section, we compare our methodology first w.r.t. model-based approaches to the plan execution diagnosis, then w.r.t. works devoted to the plan repair task.

	no-repair	safe-status	repair	repair+ safe-status
CPU time [msec] mon.+diag. (avg)	193.90	215.45	201.27	210.45
CPU time [msec] re-planning (avg)	0	407.64	424.82	504.46
# MAP's executed actions (avg)	35.42	43.23	50.90	55.25
% MAP's reached sub-goals	32%	47%	68%	73%

Table 1: Experimental results

In [Birnbaum *et al.*, 1990] a model-based approach to plan diagnosis is presented, in this approach the authors relate the health status of a planning agent to the outcome of the planning activity. Also in this paper we relate the outcome of the actions executed by plan executors to the health status of these executors, however, in this work we will consider multi-agent plans whereas Brinbaum et al. considered just a single planning agent.

The multi-agent setting is discussed in [Roos and Witteveen, 2007], where the authors introduce the notion of *plan diagnosis* as the subset of actions whose failure is consistent with the anomalous observed behavior of the system. In contrast to our work, this approach does not relate the failure of an action to the health status of the agents; it focus just on the detection of abnormal actions.

In [Kalech and Kaminka, 2007] the authors introduce the notion of *social diagnosis* to find the cause of coordination failures. In their approach, however, they do not explicitly consider plans, rather they model a hierarchy of *behaviors*: each agent selects independently from others the more appropriate behavior given its own beliefs.

The plan repair task has been addressed by a number of works (see e.g., [van der Krogt and de Weerdt, 2005; Decker and Li, 2000; Horling and Benyo, 2001], which consider both self-interested and collaborative agents; however these works are not directly applicable in our framework.
These approaches in fact are mainly focused on repairing coordination flaws occurring during plan execution, thus they involve a re-scheduling task rather than performing a re-planning step (see the GPGP solution in [Decker and Li, 2000]). In [Horling and Benyo, 2001] a solution for reorganizing the tasks among the (collaborative) agents is presented: this approach is driven by the results of a diagnostic engine which explains detected plan failures. In this case, however, the explanations are derived from a causal model where anomalous events (e.g., resource unavailable) are organized in a fault tree, and the reaction to plan failure is a proper precompiled repairing solution.

8 Conclusions.

In this paper we have formalized a closed loop of control over the execution of a multi-agent plan.

The paper contributes to show the importance of a repair strategy driven by a failure analysis which highlight the root causes of an action failure. Depending on the (possibly multiple) faults and on the activities of the agent in trouble, different course of actions are synthesized either for recovering the action failure (if the local repairing plan exists) or to bring the agent in a safe status and limit the impact of the failure.

The preliminary experimental results show that the proposed methodology is adequate to promptly react to an action failure and to actually mitigate the harmful effects of the failure. Also the computational cost of the approach is affordable since the search for a recovery plan is strongly constrained by the agent diagnosis.

As future work, the proposed framework can be extended to deal with more sophisticated notions multi-agent plan. First of all, concurrency constraints can be introduced to model joint actions (see e.g., [Micalizio and Torasso, 2008]). A more interesting extension concerns the temporal dimension. Dealing with temporal plans has a strong impact on the conformant planner. In fact the planner has to find a repairing plan that meets the set of missing goals, and that can be executed without violating any temporal constraint.

References

[Birnbaum *et al.*, 1990] L. Birnbaum, G. Collins, M. Freed, and B. Krulwich. Model-based diagnosis of planning failures. In *Proc. AAAI90*, pages 318–323, 1990.

[Cimatti and Roveri, 2000] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *JAIR*, 13:305–338, 2000.

[Cox *et al.*, 2005] J. S. Cox, E. H. Durfee, and T. Bartold. A distributed framework for solving the multiagent plan coordination problem. In *Proc. AAMAS05*, pages 821–827, 2005.

[Decker and Li, 2000] K. Decker and J. Li. Coordinating mutually exclusive resources using GPGP. *Journal of AAMAS*, 3(2):113–157, 2000.

[Horling and Benyo, 2001] B. Horling and V. Benyo, B. Lesser. Using self-diagnosis to adapt organizational structures. In *Proc. ICAA'01*, pages 529–536, 2001.

[Jensen and Veloso, 2000] R. M. Jensen and M. M. Veloso. Obdd-based universal planning for synchronized agents in non-deterministic domains. *JAIR*, 13:189–226, 2000.

[Kalech and Kaminka, 2007] M. Kalech and G. A. Kaminka. On the design of coordination diagnosis algorithms for teams of situated agents. *AI*, 171:491–513, 2007.

[Micalizio and Torasso, 2007] R. Micalizio and P. Torasso. Recovery from plan failures in partially observable environments. In *Research and Development in Intelligent Systems XXVII*, pages 321–334, 2007.

[Micalizio and Torasso, 2008] R. Micalizio and P. Torasso. Monitoring the execution of a multi-agent plan: Dealing with partial observability. In *Proc. of ECAI'08*, pages 408–412, 2008.

[Roos and Witteveen, 2007] N. Roos and C. Witteveen. Models and methods for plan diagnosis. *Journal of AAMAS*, 16:30–52, 2007.

[van der Krogt and de Weerdt, 2005] R. van der Krogt and de Weerdt. Plan repair as an extension of planning. In *Proc. of ICAPS'05*, pages 284–259, 2005.