Humans-Robots Sliding Collaboration Control in Complex Environments with Adjustable Autonomy

Nicolas Côté, Arnaud Canu, Maroua Bouzid, Abdel-Illah Mouaddib

Humans-Robots Sliding Collaboration Control in Complex Environments with Adjustable Autonomy

Nicolas Côté, Arnaud Canu, Maroua Bouzid, Abdel-Illah Mouaddib
GREYC/CNRS, Université de Caen, France
{*nicolas.cote, arnaud.canu, maroua.bouzid, abdel-illah.mouaddib*}*@unicaen.fr*

***Abstract*—Autonomous agents dealing with partial knowledge about the environment are a classical subject of study for the decision making community. Moreover, such agents sometimes have to deal with unpredictable situations, which makes any previously computed behavior useless. In this paper, we address such problems using multi-human/multi-robot interactions, where the agents evolve in a complex environment and ask humans for help when they meet unpredicted situations. We introduce a model called HHP-MDP (Human Help Provider-MDP), that aims at handling the difficult situations met by the agents by using the human's help. For this purpose, we show how the agents can detect difficult situations and send different types of requests to the set of humans. The model describes how a controller can handle different requests and assign agent requests to the humans by taking into account their previously learned abilities. This controller is designed to reduce the human's cost of bother. Moreover, we show how to optimize the human's situation awareness and limit inconsistencies between her recommendations and the agent's plans.**

I. Introduction

Robots autonomy has been widely studied in the artificial intelligence field, due to the increasing use of robots in everyday life. Despite this, computing an optimal agent behavior in a complex environment remains a highly complex task. In this paper, we study how the complexity of such planning problems can be reduced, and how the resulting performances of the agent can be improved, by adding a human in the decision making process. In such a problem, involving multiple humans and multiple agents, an efficient Human-Robot collaboration is crucial.

This collaboration is illustrated by the unmanned aerial vehicle (UAV) airport navigation problem. The *UAV-an* problem involves a set of autonomous aircrafts (UAVs), evolving on a set of taxiways. These aircrafts have to reach different objectives (runways), but a given taxiway can be obstructed by environmental factors (rain, or snow for example). When an agent meets such an obstacle, its effectors do not behave as planned. In our context, the agent can ask for help, and a human is selected to give a guidance.

More formally, this paper adresses a problem where the human and the agent have two different points of view. Punctual changes can occur in the environment. The agent receives limited information about these modifications, which prevents it from fully understanding the situation, while the human has more information. So, in order to reach its goal, the agent should ask the human for help when facing an unexpected situation.

The human can help the agent by teleoperating it or by giving recommendations (as introduced in Cote *et al.* [2011]). Teleoperation is used when the agent reaches a critical situation (strongly limited actions). Otherwise, that is, when the agent has execution troubles but the situation is not critical, the human can use recommendation and give the agent a set of subgoals to be reached before the main goal, in order to escape the difficulties. Teleoperation requires more time of the human than recommendation: the required helping time determines the agent's autonomy quality.

We split the problem of requiring help while minimizing the human's cost of bother into three subproblems :

1) problem detection: an agent facing troubles while acting has to detect this, decide whether the situation is critical or only difficult , and finally send a request to the humans,
2) request allocation: the requests must be dispatched at best to the available humains, taking their individual abilities into account,
3) situation awareness and inconsistency reduction: the agent has to communicate all the relevant information to the human, in order to reduce her bother cost and the inconsistencies between her decisions and its own plans and goals.

On the agent's side, we assume that the decision making problem is modelled by a Markovian Decision Process (MDP). Precisely, we assume that the agent is given an initial MDP describing the environment, the stochastic effects of its actions, and initial goals, and that it computes an initial action policy based on this. Then difficult situations may be foreseen at planning time, but they may also show up while the agent is executing its policy. This arises in particular when the environment suddenly changes (e.g., rain), implying unexpected changes in the behaviour of actions.

Our contribution is the HHP-MDP model, which gives a formalization of difficult situations, requests, and humans' abilities, together with strategies for problem detection, request allocation, and maximization of situation awareness. The model is flexible, and it optimizes the global system,

which works well even when the controller receives more requests than the number of available humans.

The paper is organized as follows: first, we introduce related work and background material. Then we develop our contributions along the three problems of detection, allocation, and maximization of awareness. Finally, we describe a set of proof-of-concept experiments which show the good behaviour of our model.

II. Background

Our work is based on a set of results coming from the Human-Robot Interaction (HRI) community and the Markovian Decision Process (MDP) community. In this section, we introduce these two fields and we describe the different approaches related to our model.

A. Human Robot Interaction

Human-Robot Interaction has been widely studied in the literature. Our model is based on adjustable autonomy, as described in Bradshaw *et al.* [2005], which allows one to dynamically change an agent's autonomy. Brooks [1986] introduced different kinds of autonomy (studied further by Goodrich *et al.* [2001]), Dorais *et al.* [1999] demonstrated the advantages of adjustable autonomy as compared to traditional approaches.

The model which we describe in this paper switches the agent's autonomy between three levels: (i) full autonomy, (ii) teleoperation (as described by Goldberg *et al.* [2000]), and (iii) goal biased autonomy (introduced by Cote *et al.* [2011]). This last level is a tradeoff between (i) and (ii), where the human only gives recommendations in the form of subgoals.

There are several parameters that a system involving humans should optimize. First, the *neglect time* [Crandall *et al.*, 2005] is the expected amount of time during which a request from an agent can be ignored before its performance drops down below some threshold. According to this definition, the **neglected time** should be minimized: it expresses the average time during which an agent needs human help, but has to work without it. Second, **situation awareness** [Endsley, 1988] must be maximized: it describes how well the human perceives its environment within a volume of time and space, understands its meaning, and projects its status in the near future. Yanco [2004] showed that a lack of *Situation Awareness* has a negative effect on the quality of teleoperation or recommendations. Finally, the **bother cost** should be minimized: it describes how often the human is asked for help, with frequent requests reducing the overall system efficiency Cohen *et al.* [2011].

This paper deals with Human-Robot Collaboration, which is only one of the possible kinds of Human-Robot Interaction. Some works involving human-robot collaboration make the assumption that a robot is a "tool", used by the operator. In Bechar and Edan [2003] for example, a single human operator is working with a set of robots and controls them with different levels of autonomy. Other works are based on a dialogue between the human and the robot: in Fong *et al.* [1999, 2003] for example, the robot can ask questions to a single human operator when its autonomy is limited. In this paper, we use a collaboration control in which the robots see the available humans as resources, able to compensate for the limitations of their autonomy.

In Ishikawa and Suzuki [1997], the robots execute their tasks in a hazardous environment and the operator observes them. When an agent reaches a difficult situation, the human teleoperates it. In the same vein, Lee *et al.* [2010] and Wang *et al.* [2009] describe a problem involving multi-humans and multi-robots teams, where the humans observe the agents and assist them during the difficult situations. However, the humans have to detect when an agent is in a difficult situation: Ohba *et al.* [1999] showed the negative effect of the time-delay on such teleoperation systems. In our approach, such delays are minimized, because we seek to optimize *Situation Awareness*, *Bother Cost* and *Neglected Time*.

B. Markovian Decision Process

The Markovian Decision Process (MDP) model [Puterman, 1994] is an elegant framework, which is widely used for modelling decision-making problems of autonomous agents evolving in an uncertain environment. An MDP is a tuple $\langle S, A, T, R\rangle$ where (1) S is a finite set of states, (2) A is a finite set of actions, (3) $T : S \times A \times S \to [0; 1]$ is a transition function giving the probability to move from a state s to s', when the action a is applied and (4) $R : S \times A \to \mathbb{R}$ is a reward function giving the positive or negative reward resulting of an action a taken in a state s.

Solving a problem described with an MDP means computing a policy π, that is, a function which gives, for each state s, an action $a = \pi(s)$ to apply so as to maximize immediate and expected (discounted) future reward. *Bellman's equation* gives the expected value $V(s)$ of a state (expected long-term discounted reward) with $0 < \gamma < 1$:

$$V(s) = \max_a (R(s, a) + \gamma \sum_{s'} T(s, a, s')V(s')) \qquad (1)$$

An optimal policy π^*, that is, a policy giving the best action to take in each state, can be computed using this equation. The intended behaviour of an agent is to compute π^* off-line, given an MDP modelling its actions, goals, and the environment, and then, at *execution time*, to act reactively according to π^*.

In this paper, we used goal-oriented MDPs, in which only one state (the goal) brings a positive reward, while each other state has a null or negative reward (whatever the action). Moreover, once the goal is reached, the agent can receive

a new one, and then must re-compute the whole policy (as this changes the R component of the MDP).

In Mouaddib *et al.* [2010], an MDP based approach is described which deals with agents asking the humans for teleoperation, but this model suffers from several limitations: in particular, the latency of the model is high, and it deals with only one robot and a permanently available human. OPOMDPs [Armstrong-Crews and Veloso, 2007] were introduced for planning requests for human help under partial observability, but this model assumes that the human is permanently available and only plays the role of an observation provider. HOP-POMDPs [Rosenthal and Veloso, 2011] are similar to OPOMDPs, with a model of the human (a probability distribution on its availability), but here again the role of the human is strongly limited.

In this paper, we used an MDP based approach: this model not only allows the agents to evolve autonomously in complex environments, but we can also use its properties to quickly assign help requests to the humans (without needing any dialogue) and to maximize the human's *Situation Awareness* while he is helping an agent. Finally, our approach integrates *goal-biased* techniques: the human gives recommendations to the agent, as a set of sub-goals to be reached before the main goal. Hence we have an intermediate level of autonomy between full autonomy and teleoperation, which reduces the time spent by the humans helping the agents.

C. Teleoperation and recommandation

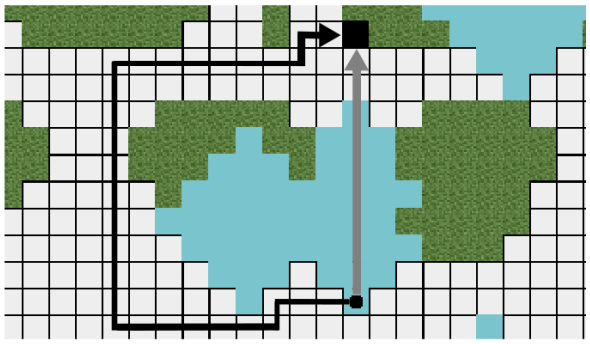

Figure 1. Example of teleoperation utility

In this section, we present the utility of human recommandation on a small example described on Figure 1 (excerpted from our experiments). The agent has just fallen into snow and now has a small probability to reach its goal in a reasonable amount of time if it follows its initial policy. Hence it sends a help request.

The human can help the agent in two different ways:

- By teleoperating it to its goal by following the dark edge. The human gives the actions to apply for each successive state to the agent.
- By giving the agent some waypoints, in this case the two extreme points on the left of the dark edge, as recommended subgoals. The agent changes its reward function by adding a new reward to the indicated subgoals.

In both cases, the human decides to stop helping the agent as soon as she think that the agent can reach its goal alone.

III. Critical States Detection (*Neglected Time* minimization)

Efficient techniques for detecting critical states are crucial: the faster an agent detects such states and asks for help, the more it reduces its *Neglected Time*. Two complementary approaches are described here: we can detect a set of critical states off-line, during policy computation, and detect additional ones during the execution of this policy.

A. Absorbing states

A set of absorbing states is a "trap", from which an agent has a null probability to escape (to reach another state, out of this set). More formally, with g the goal of the agent, a state s is absorbing if $P(g|s,\pi) = 0$, that is, the probability to reach the goal from s is null, no matter how long the policy π is followed. This set can be computed straight forwardly with a backwards search from the goal in the policy graph.

If the agent is in such an absorbing state, then its capabilities are not good enough to reach the goal, so it must resort on the human's higher abilities and send her a request. Precisely in this case, the agent has no action allowing it to reach the goal even with a small probability, while the human has such actions. In the *UAV-an* problem, an agent could be in such an absorbing state if it does not have enough space on its taxiway to maneuver. Then it needs help from a human, until it reaches larger taxiways. However, such an approach only detects the "predictable" critical states: the agent could reach a critical state which was not predicted, because, say, of a new (intially unknown) obstacle.

B. Online detection

According to Bellman's equation (Eq. 1), an optimal policy maximizes the expected (long-term) value at each step. It follows that when everything goes well at execution time (random drawings of action outcomes are beneficial to the agent), the expected value of each state should grow higher and higher as the agent executes its policy. We will use this fact as a means to detect problems at execution time.

Formally, define an *ideal trajectory* of a policy as one which goes at each state from one state to the reachable one with maximal expected value:

Definition 1: Let $\langle S, A, T, R\rangle$ be a goal-oriented MDP (i.e., $R(s_g, \_) = 1$ and $R(s, \_) \leq 0$ for all $s \neq s_g$), let π^* be an optimal policy for it, and let $s \in S$ be a state. A sequence of states $(s_0, s_1, \ldots, s_n)$ is called an *ideal trajectory* for π^*

in s if $s_0 = s$, $s_n = s_g$, $s_t \neq s_g$ for $t \neq n$, and for $t = 1, \ldots, n$ it holds that

$$s_t \in \underset{s', T(s_{t-1}, \pi(s_{t-1}), s') > 0}{\operatorname{argmax}} V(s')$$

Proposition 1: Let $\langle S, A, T, R \rangle$ be a goal-oriented MDP with goal s_g, and let $(s_0, s_1, \ldots, s_n)$ be an ideal trajectory for some optimal policy π^* and state s. Then $V(s_0) \leq V(s_1) \leq \cdots \leq V(s_n)$ holds.

Proof. Because π^* is optimal, it must satisfy Bellman's equation in the sense that for $t = 0, \ldots, n-1$:

$$\pi^*(s_t) \in \underset{a \in A}{\operatorname{argmax}} (R(s_t, a) + \gamma \sum_{s'} T(s_t, a, s') V(s'))$$

Now by definition of an ideal trajectory, we have $V(s_{t+1}) \geq V(s')$ for all s' reachable from s_t through the action $\pi^*(s_t)$, and because $T(s, \pi^*(s_t), \cdot)$ is a probability distribution, we get

$$V(s_{t+1}) \geq \sum_{s'} T(s, \pi^*(s_t), s') V(s') \tag{2}$$

Now because the MDP is goal-oriented and $s_t \neq s_g$ we have $R(s_t, \pi^*(s_t)) \leq 0$, which together with 2 yields

$$\gamma V(s_{t+1}) \geq R(s_t, \pi^*(s_t)) \gamma \sum_{s'} T(s, \pi^*(s_t), s') V(s')$$

Because π^* is optimal, the right-hand side is at least $V(s_t)$, and because γ is less than 1 we get $V(s_{t+1}) \geq V(s_t)$, as desired. □

Based on this analysis, we take the changes of the current expected value as an indicator to at what extent the execution goes well. Whenever expected values do not grow as fast as expected as execution time, this may mean either that random drawings of action outcomes are adversarial to the agent (for instance, it fails to brake three times in a row while this has only a 0.1 probability, or that the agent is not executing an optimal policy and hence, that the model has changed without the agent detecting this (e.g., undetected water on the taxiway modifies the model of its effectors)).

The only exception to expected value growing along an ideal trajectory is when a goal is reached: then, a new goal is assigned to the agent, and the expected values temporary drop down (this new goal being usually far from the previous one). Figure 2 describes such an example of expected values changes.

Summarizing, if the expected values do not grow as fast as expected, the agent probably faces a problem. The growth can be too slow (in the *UAV-an* problem, undetected water on the taxiway forces the agent to slow down), or critical (snow on the taxiway, making the aircraft unable to move). Then help is required from the human, who we assume to be able to detect changes in the environment and help the agent accordingly.

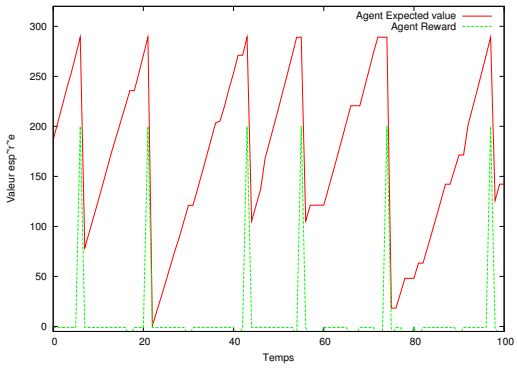

Figure 2. Expected values history and rewards history

In order to detect such problems (slow or critical growth) while avoiding to require help as soon as a few random drawings of action outcomes are adversarial, we propose a strategy in which the agent keeps a restricted history of recent expected values, and compares the associated growth to the average one, on the whole history. When expected values remain constant or drop down (for another reason than goal completion), then the agent asks for help.

Formally, write s_i for the state in which the agent is at timestep i, t for the current timestep, and w for the size of the observation window. We define *Average* and *Windowed Average* by:

$$\begin{aligned} Avg &= \frac{1}{t} \sum_{i=0}^{t} (V(s_i) - V(s_{i-1})) \cdot \mathbb{1}[R(s_{i-1}) \leq 0] \\ WAvg &= \frac{1}{w} \sum_{i=(t-w)}^{t} (V(s_i) - V(s_{i-1})) \cdot \mathbb{1}[R(s_{i-1}) \leq 0] \end{aligned}$$

where $\mathbb{1}[R(s_{i-1}) \leq 0]$ is 0 for $R(s_{i-1}) > 0$ and 1 otherwise, hence excluding from the definition any timestep following a goal achievement. Then at execution time, the agent evaluates the current growth rate by comparing $C := WAvg/Avg$ to some threshold. When C is low enough (for example, $C < 0.5$, meaning expected values grow twice slower than usual), then the agent asks for help (providing the value of C in the request). Moreover, for $WAvg \leq 0$, meaning the agent moves away from the goal, the agent asks for a critical help. We discuss the value of C in the experiments section.

IV. Controller (*Bother Cost* minimization)

Our model includes a centralized controller, which receives requests sent by the agents and assigns them to the available humans. In the simple case of one human and one agent, such a controller would be useless. However, most of the time, agents issue more requests than the number of available humans (or, sometimes, there are more available

humans than pending requests). Then, we propose a controller for optimizing collaboration efficiency, by dispatching the requests to the available humans according to their abilities.

A. Request types

Recall that agents can send three kinds of requests, each one associated to a given priority:

- **absorbing state**: these requests are the most critical ones. The agent knows it will not be able to reach the goal with its own abilities. Usually, such a request implies a breakdown that requires human intervention, or a bad initial state of the agent. The resulting intervention can be a teleoperation, or repairing the robot.
- **Critical values changes** ($WAvg \leq 0$): such a request is critical, since the agent is not going towards its goal. Usually, the agent is in a difficult situation for which it cannot compute a solution with its limited sensors and actuators. The resulting intervention can be a teleoperation, until the agent can be autonomous again.
- **Low values changes** e.g., $0 < C < 0.5$: these requests are not critical. The agent heads its goal, but too slowly. The human will only have to give recommendations to the agent, in order to help it find a new way to reach the goal.

As an illustration of the latter case, assume an aircraft moves forward more slowly than expected because of undetected rain on the taxiway. A recommendation by a human (who can detect where the taxiway is wet) could be to go backwards (contrary to the agent's initial plan) so as to escape the wet taxiway, then go to another (dry) taxiway and finally meet the initial one again at a place where it is dry.

B. Request allocation

When many agents are involved, the controller will receive a lot of requests at the same time, usually more than the number of available human operators. In order to minimize the bother cost, we choose not to interrupt an operator while he is helping an agent. Then the controller has to choose an efficient allocation of requests to the operators, in order to maximize the impact of the collaboration. We formalize such expected efficiency through a model of each operator's abilities.

1) Models for operators: The efficiency of a collaboration depends on two factors:

- the time needed by the human to deal with the request,
- the resulting expected gain.

Hence we assume the controller to be given two functions: $time(r, o)$ gives the average time needed for the operator o to deal with a request r (depending on its type), and $gain(r, o)$ gives the expected gain, when the operator o deals with r. We assume $time(r, o)$ to be given on an absolute scale common to all request types and operators but not necessarily related to the timesteps of the agent. This indeed essentially measures the *Bother Cost* for the human. As for gain, we define this as an expected value of $V(s_{end}) - V(s_{start})$ with s_{start} the state in which the agent asks for a teleoperation or recommendation, and s_{end} the state in which the collaboration ends.

We assume these functions to have initial values estimated from average data. Moreover, these values can be updated each time a human indeed helps an agent, in a manner similar to reinforcement learning approaches. However, this part is not an important contribution of this paper, so we do not discuss it further.

2) Choosing the allocations: Given each operator's model, the controller is able to compute a weighted gain for each operator o and request r:

$$G(r, o) = \frac{gain(r, o)}{time(r, o)}$$

Then when a human operator o becomes available and there are pending requests, the controller sends o the best pending request, given by $r^*(o) = \text{argmax}_r\, G(r, o)$. Dually, when several human operators are available and a request r comes in, the chosen human operator is given by $o^*(r) = \text{argmax}_o\, G(r, o)$.

V. Information sent to the human (*Situation Awareness* maximization)

We have described how an agent detects problems and issues requests for help, and how these requests are allocated to human operators. However, if the human does not have enough information about the problem, its help will not be optimal. So we have to maximize the human's awareness of the situation faced by the agent. This *Situation Awareness* is a crucial factor for the performances of our system. Good awareness will lead to fast decisions, and limit inconsistencies with the policy of the agent. The HHP-MDP model takes this into account by using the MDP model to improve the human's understanding of the situation.

When a human is helping an agent, irrelevant information makes it difficult for her to find a good solution. Hence we must minimize the amount of information given by the agent to the human while giving the relevant data. Generally speaking, call an information *relevant* if it allows the human to identify the agent's problem. We split this information into two parts: global information (about the agent's longterm will) and local information (about its neighborhood).

A. Global information

Information about the long-term goal can be used for planning problems. Most of the time, such information is used when the human gives recommendations to the agent. Here there are two relevant kinds of data: the global strategy (planned to be) used by the agent to reach the goal, and

information about the goal itself. We extract such data the MDP as follows:

- **Information about the goal:** only the goal has a positive reward, so we highlight any state s such that $\exists a \in A, R(s,a) > 0$.
- **Information about the agent's strategy:** we use the policy graph to extract the global strategy, by building clusters of adjacent states with the same action according to the agent's policy. Such clusters are easier for the human to understand than a full policy.

As an example, Figure 3 (left) describes a full policy, and (right) gives the associated clusters, as communicated to the human operator (the smallest clusters — less than 5% of the state space — were ignored). Such a representation

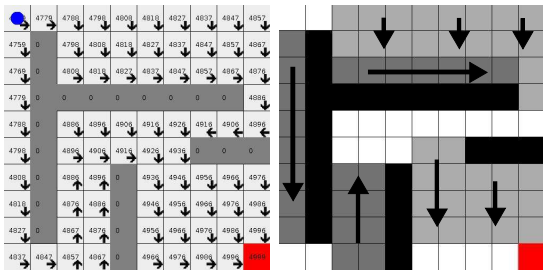

Figure 3. The long-term strategy of an agent

highlights the important parts of the policy, maximizing *Situation Awareness*.

B. Local information

When a human starts a teleoperation, he needs information in order to choose an action which will be as consistent as possible with the agent behavior. Moreover, again such data should be easy to understand. There are two kinds of such data:

- **Agent's history**: the human needs to understand the problem of the agent. For this purpose, we display the states reached by the agent during the last timesteps and the policy applied in each one.
- **Agent's policy**: the human has to choose actions as consistent as possible with the agent's policy, hence she needs information about the reachable states (all the states with a nonzero probability to be reached in n timesteps), and about the agent's policy on these states.

An example of the resulting display is depicted on figure 4. Here, we understand that the agent policy is to go down. It came from the left, tried to go down several times, but stayed in the same state, so it sends a critical ask for help.

Using these two views (local and global), we maximize the *Situation Awareness* of the human, in that we avoid giving useless information about the agent, and we keep the human focused on the problem he has to solve.

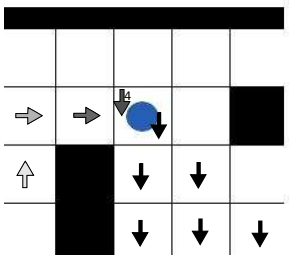

Figure 4. The short-term strategy of an agent

VI. Experimentation

This section describes experimental results obtained with our model, on the *UAV-an* problem previously introduced. Our aim is to give a proof of concept and to investigate the influence of various parameters (such as the number of operators).

A. Experimental setting

We used a map of the Montpellier airport (Figure 5), which involves two possible airways connected by several taxiways around a central building.

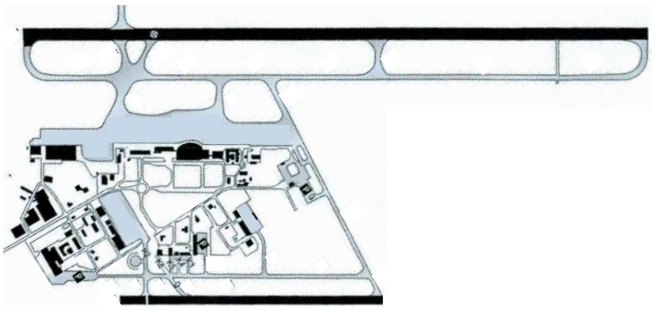

Figure 5. Montpellier airport

*UAV*s are randomly launched from the building and have to reach an airway. Usually, they are able to reach their objective using a fully autonomous approach. However, sometimes, the weather (rain or snow) makes it impossible for them to evolve safely on the taxiways. Human operators are then asked for help. When an agent reaches the airway, it receives a reward of 200. Moreover, the operator model is initialized as follows, based on collected data:

- low changes request: the average gain on the expected value is 5 and the average time needed is 20 timesteps,
- critical changes: average gain = 8, average time = 15,
- absorbing state: average gain = 12, average time = 10.

We built a benchmark, based on this airport, as depicted in figure 6. Rain or snow start randomly during the execution of the problem: in figure 6, the blue spots are puddles due to the rain. One agent is currently evolving on the taxiways

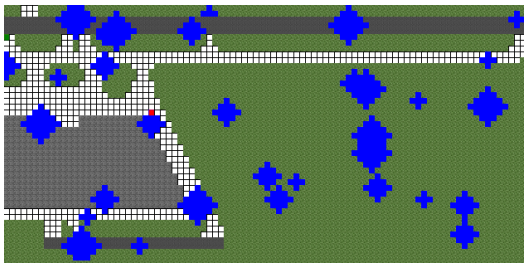

Figure 6. Our benchmark

(the red square) towards the top airway (green square), but it is not able to move anymore, so it asks for help.

The agent can move forward, stay where it is and turn to the left or the right. In normal situations, it has a probability $P = 0.1$ to stay in its current state when trying to move forward, due to its own limitations. When it drives through a puddle, this probability not only rises to $P = 0.3$ (because the agent has to move slowly), but there is also a probability to deviate on the sides (with $P = 0.1$ for each side). Finally, when the agent drives on snow, it has a probability $P = 0.4$ to deviate on each side.

The initial policy is computed assuming all situations are normal, i.e., only a 0.1 probability not to move forward as desired. Moreover, the agent cannot directly detect snow or rain at execution time.

As concerns humans, we involved up to three persons, which used a specific graphical interface for teleoperating or recommending in the simulation. One of them (Human 1) we familiar with the problem and the interface, while Human 2 only experimented the interface a few times, and Human 3 had no prior experience with it.

B. Detecting a slow or critical changes

We ran a set of experiments, each one over 3,000 timesteps. For observation of expected value evoluation, we used a window of 4 timesteps. In our first experiment (Figure 6), we only used one agent and one operator. Each time the agent evolved through a puddle, it managed to detect it and sent a request for help. These requests were distributed as follows:

- $11.5\,\%$ of low changes requests,
- $88.5\,\%$ of critical changes requests.

This distribution depends on the bound above which an evolution is considered as critical (we used $C = 0.5$). This first test confirms the ability of the agents to detect problems at execution time and to send requests accordingly.

C. Influence of the number of agents

We ran another set of experiments, involving 1 to 15 agents at the same time. During these experiments, we computed the percentage of critical requests ignored by the controller, as depicted in figure 7.

With less than 10 agents, one operator is enough to deal with more than 70% of the critical requests. However, with

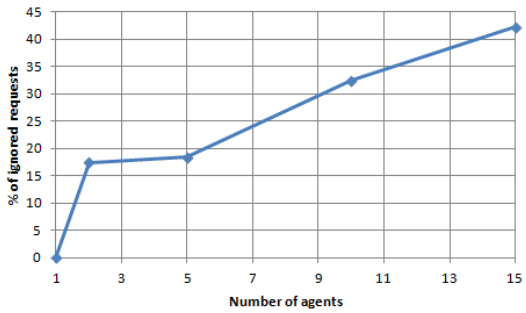

Figure 7. Percentage of ignored critical requests

more than 10 agents, the performances of the system drop down (half the requests are ignored with 15 agents).

Hence, naturally we need to add more operators: with k available humans, the number of ignored requests is divided by k. Still, the complexity of allocating them remains low (linear in the number of operators), making it possible to scale up.

D. Involving several humans

We ran a set of experiments with three persons, as described in the experimental setting. Figure 8 describes our results.

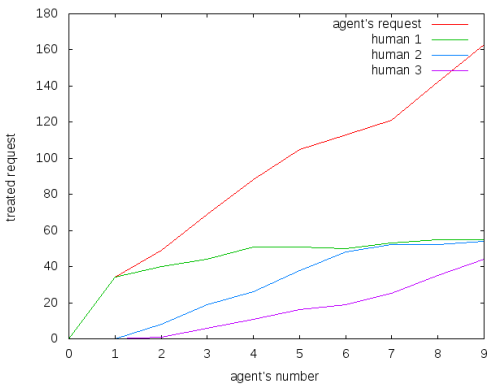

Figure 8. Number of treated requests

This figure shows the number of requests treated by Humans 1, 2 and 3, as a function of the number of agents. With a small number of agents, Human 1 deals with a lot of requests, while Human 2 deals with a smaller number and Human 3 deals with almost none. This is because Human 1 is more familiar with the simulation and hence, handles requests much faster. However, the number of handled requests seems to converge to 60 for each human when the number of agents is large enough, which appears to be the limit of a human for this problem.

The relative abilities of the humans, as computed from average data gathered in our own experiments, are given in Table I. This data is clearly consistent with the number of requests handled as depicted on Figure 8.

Human	Average time of help	Expected Value
Human 1	6.635	315.31
Human 2	9.928	295.72
Human 3	7.947	204.97

Table I
HUMANS CAPACITIES

VII. CONCLUSION

In this paper, we described a new approach to deal with decision making problems where unpredictable situations can change the environment at execution time. We proposed a multi-agent/multi-human collaboration based approach, where the agents can ask humans operators for help. We described three main contributions: how an agent detects such an unpredictable situation and asks for help, how to choose which agent needs help the most, and how to give an efficient description of the problem to the human operator. Finally, we illustrated our approach on the *UAV-an* benchmark: we investigated in particular the influence of the number of operators and agents. We are now working on a new approach, in order to reduce the number of help requests, in which the human sends information to the agent about its model when an event occurs.

VIII. ACKNOWLEDGMENTS

This work has been supported by the NRA (French National Research Agency) and the DGA (Defense Procurement Agency) (ANR-09-CORD-103).

REFERENCES

N. Armstrong-Crews and M. Veloso. Oracular partially observable markov decision processes: A very special case. *Internationnal Conference on Robotics and Automation*, pages 2477–2482, 2007.

A. Bechar and Y. Edan. Human-robot collaboration for improved target recognition of agricultural robots. *Industrial Robot: An International Journal*, 30(5):432–436, 2003.

J.M. Bradshaw, H. Jung, S. Kulkarni, M. Johnson, P. Feltovich, J. Allen, L. Bunch, N. Chambers, L. Galescu, R. Jeffers, et al. Kaa: Policy-based explorations of a richer model for adjustable autonomy. pages 214–221, 2005.

R. Brooks. *A robust layered control system for a mobile robot*, volume 2. IEEE Journal of Robotics and Automation, 1986.

R. Cohen, H. Jung, M.W. Fleming, and M.Y.K. Cheng. A user modeling approach for reasoning about interaction sensitive to bother and its application to hospital decision scenarios. *User Modeling and User-Adapted Interaction*, pages 1–44, 2011.

N Cote, M Bouzid, and A-I Mouaddib. Integrating the human recommendations in the decision process of autonomous agents: A goal biased markov decision process. *Fall Symposium AAAI*, 2011.

J.W. Crandall, M.A. Goodrich, D.R. Olsen Jr, and C.W. Nielsen. Validating human-robot interaction schemes in multitasking environments. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(4):438–449, 2005.

G. Dorais, R.P. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost. Adjustable autonomy for human-centered autonomous systems. *Working notes of the Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems*, pages 16–35, 1999.

M.R. Endsley. Design and evaluation for situation awareness enhancement. 32(2):97–101, 1988.

T. Fong, C. Thorpe, and C. Baur. Collaborative control: A robot-centric model for vehicle teleoperation. 1999.

T. Fong, C. Thorpe, and C. Baur. Robot, asker of questions. *Robotics and Autonomous systems*, 42(3-4):235–243, 2003.

K. Goldberg, B. Chen, R. Solomon, S. Bui, B. Farzin, J. Heitler, D. Poon, and G. Smith. Collaborative teleoperation via the internet. 2:2019–2024, 2000.

M.A. Goodrich, D.R. Olsen, J.W. Crandall, and T.J. Palmer. Experiments in adjustable autonomy. pages 1624–1629, 2001.

N. Ishikawa and K. Suzuki. Development of a human and robot collaborative system for inspecting patrol of nuclear power plants. pages 118–123, 1997.

P.J. Lee, H. Wang, S.Y. Chien, M. Lewis, P. Scerri, P. Velagapudi, K. Sycara, and B. Kane. Teams for teams performance in multi-human/multi-robot teams. 54(4):438–442, 2010.

Abdel-Illah Mouaddib, Shlomo Zilberstein, Aurelie Beynier, and Laurent Jeanpierre. A decision-theoretic approach to cooperative control and adjustable autonomy. *European Conference on Artificial Intelligence*, pages 971–972, 2010.

K. Ohba, S. Kawabata, N.Y. Chong, K. Komoriya, T. Matsumaru, N. Matsuhira, K. Takase, and K. Tanie. Remote collaboration through time delay in multiple teleoperation. 3:1866–1871, 1999.

M.L. Puterman. Markov decision processes: Discrete stochastic dynamic programming. 1994.

S. Rosenthal and M. Veloso. Modeling humans as observation providers using pomdps. *In Proceedings of International Symposium on Robot-Human Communication*, 2011.

H. Wang, M. Lewis, P. Velagapudi, P. Scerri, and K. Sycara. How search and its subtasks scale in n robots. pages 141–148, 2009.

Holly A. Yanco. where am i? acquiring situation awareness using a remote robot platform. *In IEEE Conference on Systems, Man and Cybernetics*, pages 2835–2840, 2004.