Integrating the Human Recommendations in the Decision process of autonomous agents: A Goal Biased Markov Decision Process

Nicolas Côté, Maroua Bouzid, Abdel-Illah Mouaddib

Integrating the Human Recommendations in the Decision process of autonomous agents: A Goal Biased Markov Decision Process

Nicolas Côté - Maroua Bouzid - Abdel-Illah Mouaddib
GREYC - CNRS (UMR0672), Université de Caen, Basse-Normandie, ENSICAEN
Boulevard du Maréchal Juin 14032 Caen CEDEX, France

Abstract

In this paper, we address the problem of computing the policy of an autonomous agent, taking human recommendations into account which could be appropriate for mixed initiative, or adjustable autonomy. For this purpose, we present Goal Biased Markov Decision Process (GBMDP) which assume two kinds of recommendation. The human recommends to the agent to avoid some situations (represented by undesirable states), or he recommends favorable situations represented by desirable states. The agent takes those recommendations into account by updating its policy (only updating the states concerned by the recommendations, not the whole policy). We show that GBMDP is efficient and it improves the human's intervention by reducing its time of attention paid to the agent. Moreover, GBMDP optimizes robot's computation time by updating only the necessary states. We also show how GBMDP can consider more than one recommendation. Finally, our experiments show how we update policies which are intractable by standard approaches.

Introduction

Robots become more and more important in the daily low activity. A robot can perform repetitive or unpleasant tasks instead of the human, as for example a vacuum cleaner robot (VC problem). Most of the time, the robot is autonomous. However, the capabilities of the robot sensors could be not precise enough to perceive all the obstacles. In that case, the human can give recommendations about forbidden or dangerous areas. The key issue in this paper is to formalize human recommendations and how the agent interprets them.

Different approaches dedicated to the full agent's autonomy have been proposed in the literature (S. Thrun 2005). However, most of them are highly complex. To avoid this limitation and to increase the reliability, the human-robot interaction (HRI) community integrates human interventions into the agent's decision process such as adjustable autonomy (Scerri *et al.* 2002; Cheng and Zelinsky 2001). We focus, in this paper, on how to introduce the human recommendations in an autonomous agent policy. Such recommendations are required when the full autonomy is not desired or not feasible, but an adjustable autonomy is required.

Adjustable autonomy is a concept in which the robot can share the control with an external entity and a human can interact with an agent to achieve a mission. The agent has information about the world dynamics, but it can make some errors when executing an action. To avoid this, or at least to make the agent recover faster, the human gives some information to the agent about the world model. In this paper, the agent describes the stochastic world using a Markov Decision Process, and we discuss how to extend this model to consider human recommendations.

The idea of human-agent approach based on Goal Biased Autonomy has been introduced in (Brooks 1986). To our knowledge, no work has been developed about this idea except in (Crandall and Goodrich 2002). The human gives informations to the agent in order to improve its world model and then to reach its goal. In the VC example, the robot computes a path to reach a place for cleaning while ignoring that there are roadworks. The human provides his information to reinforce robustness and adaptability of the robots.

The degree of autonomy is an important parameter for the quality of the collaboration between the robot and its human partner. Moreover, in adjustable autonomy, another important parameter is the number of agents that a human can manage (it depends on the time required by each agent). The goal biased principle can be useful when the human has more than one agent to supervise since giving recommendation is a lower workload than controlling the robot.

The paper is outlined as follows: first we present some related works. Second, we develop our approach: goal biased and Markov Decision Processes. Then, we describe our two main contributions : giving recommendations to the agent and optimizing the policy update (according to those recommendations). We explain the two types of recommendations given by the human and how to use them. Then, we present experiments showing the benefits of our approach. We show that we deal with large problems in reasonable time, while existing approaches suffer from a combinatorial explosion.

Related Works

Developing autonomy for multiple robots in the real world has attracted the attention of the Artificial Intelligence and robotics communities. Some issues that need Human-robot collaboration have been studied (Dorais *et al.* 1999) where one describes how we use the adjustable autonomy in real-

world. These different papers do not give a general approach that can be used in multiple robot settings, but only method applied to specific issues. (A.Mouaddib *et al.* 2010) or (Scerri *et al.* 2001) give framework developed for real world and take into account the uncertainty of the world model. In these papers, the agent asks the human for teleoperation.

The goal biased approach (Brooks 1986) describes some behavior-based robotic. (Crandall and Goodrich 2002) presents an implementation of this work where the human gives waypoints to a robot, guiding it through a maze. The robot has no ability of reasoning since it is controlled by the human: it changes its action when the human recommends it. There are no studies about integrating adjustable autonomy in the robot decision problem. From those different ideas, our aim is to show how to give some information to an agent about the real-world and how the agent takes them into consideration. A fully autonomous agent requires no time from a human in contrast to a teleoperated agent which requires the permanent attention of the human. The goal biased is a middle ground between full autonomy and teleoperation. The following table summarizes the advantage and drawbacks of each approach.

Type of autonomy	Agent Effectiveness	Time required
Fully autonomous	poor	no time
Teleoperated	very good	full time
Goal Biased	good	some time

The neglect time has been studied for how to supervise more than one agent. (Crandall *et al.* 2005) studies the neglect time, but not under uncertainty. In the next section, we present the Markov Decision Process (MDP) and the notion of goal biased. After this section, we explain how an agent and a human can interact without teleoperation.

MDP and Goal Biased Autonomy

An MDP (Puterman 1994) is a tuple $< S, A, T, R >$ where (1) S is a finite set of all possible states, (2) A is a finite set of all possible actions, (3) $T : S \times A \times S \rightarrow [0; 1]$ is a transition function giving for an action a and a state s, the probability to move to s', and (4) $R : S \times A \rightarrow \mathbb{R}$ is a function giving the reward associated to an action a taken in a state s. Solving a problem represented by an MDP means computing an optimal policy. Usually, the Bellman equation is used to compute the value V of a state, with $0 < \gamma \leq 1$ the discount factor : $V(s) = R(s, a) + \gamma \max_a \sum_{s'} T(s, a, s')V(s')$.

Solving the Bellman equation allows us to compute an optimal policy π which determines for each state, the best action to execute. Similarly, $Q(s, a)$ is the expected cumulative value reward received by taking action a in state s. When the expectation gives a weak reward, the human can give recommendation to improve the agent's performances. The human gives recommendations concerning the environment. The agent takes into account these recommendations and updates its optimal policy by updating only some states due to the recommendations. In the first section, we present this idea in detail.

The Human Recommendations

The principle of the Goal Biased Autonomy is that a human gives some recommendations to the autonomous agent. We distinguish two kind of recommendations :the desirable and undesirable states. First, we explain how a human expresses recommendations that are in two types. Second, we explain how an agent uses the recommendations to update its policy. Assuming that the human is rational.

Human Recommendations We define two kind of human recommendations as follows:

- **Recommendation expressing a weak constraint** : The human recommends that a state is desirable. The agent could not follow this recommendation.
- **Recommendation expressing a strong constraint** : The human recommends that a state is undesirable and the agent should never reach this state.

If the human recommends that a state is desirable, it means that the agent should reach this state before reaching its goal. In the opposite, for unreachable state, an agent will find another way to reach its goal if the human forbids some states in its policy. This shows that if a recommendation is given on a state, it can directly modify the probability to reach the goal. If there is no path to the desirable state, the agent asks the human for teleoperation. A state is undesirable if the human forbids the agent to reach this state. As we are in a stochastic environment, the agent can even reach this state. In this case, it asks the human for teleoperation.

These two recommendations are really useful for the autonomous agent when meeting difficult situations. In the VC problem, the human indicates to the agent where it should never go (stairs, ...) or where it must necessarily go (ground dirty). An agent can partially update its policy depending on the human recommendations. It means that we have to find all the states where the expected value would be affected by those recommendations. Our aim is to define accurately the set of states to update. When the agent has this list, it uses the value iteration algorithm restricted to this list to update the policy.

Updating the set of undesirable states

The objective is to compute the set of states affected by the list of undesirable states. The human's recommendations can entirely change the policy of the agent. We define an absorbing state named w. When the agent attains this state, it can't move to another state, and its reward is null. More formally, an absorbing state w is such that: $\forall a \in A, \forall s \in S \setminus \{w\}, T(w, a, s) = 0, T(w, a, w) = 1$ and $\forall a \in A, R(w, a) = 0$. If the agent reaches the state w, then it would ask the human for teleoperation.

Let f be a forbidden state given by the human, and let $\hat{T}$ be the new transition function taking into account the human recommendations. For each state s, for each action a the probability to reach f is null. Formally: $\hat{T}(s, a, f) = 0$, $\hat{T}(s, a, w) = T(s, a, f)$ and $\forall s' \neq f, s' \neq w, \hat{T}(s, a, s') = T(s, a, s')$. To compute the set of undesirable states, we consider the policy graph of an agent where a state is represented by a node. Two nodes s_1 and s_2 are connected if the

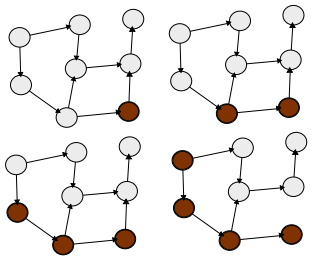

Figure 1: Example of the algorithm 1

probability to transit from s_1 to s_2 when following the optimal policy is not null. We introduce the algorithm 1 which allows the agent to compute a set of updated states called U. We define $Parent(s, \pi(s)) = \{s' \in S \mid T(s', \pi(s), s) > 0\}$ such that a state $s \in U$ iff $s \in Parent(f, \pi(f))$, or $s \in Parent(s', \pi(s'))$ (where $s' \in U$).

Algorithm 1: Undesirable State Dependences Detection

Input: f: the recommended states, G: the policy graph
Data: P: a stack of vertex
Result: U: the set of states to update
$P \leftarrow f$
while P *not empty* **do**
  $sp \leftarrow pop(P)$
  foreach $s \in Parents(sp, \pi(sp))$ **do**
    if $s \notin U$ **then**
      $P \leftarrow P \cup \{sp\}$
      $U \leftarrow U \cup \{sp\}$
return U

We have to prove that a state $s' \notin U$ has a value of the optimal policy and doesn't need to be updated because of the human's recommendations. For this purpose, let $\hat{Q}(s, a)$ be the new expected cumulative value reward for an action a and a state s after the recommendations and $\hat{V}(s)$ the new expected cumulative value reward for a state s.

Lemma 1 *For any state $s \notin U$ and any action $a \in A$, the Q value in the new model is equal or less than the value in the old model: $\forall a \in A, \forall s \notin U, \hat{Q}(s, a) \leq Q(s, a)$*

Proof. We assume without loss of generality $\forall s, R(s, a) \geq 0$. By induction, at horizon i :

- $i = 0 : \hat{Q}^0(s, a) = R(s, a) = Q^0(s, a)$
- induction hypothesis (IH) : $\hat{Q}^i(s, a) \leq Q^i(s, a)$.
  $\hat{Q}^{i+1}(s, a) = R(s, a) + \gamma \sum_{s'} \hat{T}(s, a, s')\hat{V}^i(s')$
  and (IH) $\Rightarrow \hat{V}^i(s) \leq V^i(s)$
  $\hat{Q}^{i+1}(s, a) \leq R(s, a) + \gamma \sum_{s'} \hat{T}(s, a, s')V^i(s')$
  $= R(s, a) + \gamma \Big[\hat{T}(s, a, f)V^i(f) + \hat{T}(s, a, w)V^i(w)$
  $+ \sum_{s' \neq f, w} \hat{T}(s, a, s')V^i(s')\Big]$

$$= R(s, a) \quad + \gamma \sum_{s' \neq f, w} T(s, a, s')V^i(s') \quad \textbf{(1)}$$

$$\leq R(s, a) \quad + \gamma \sum_{s'} T(s, a, s')V^i(s') = Q^{i+1}(s, a) \quad \textbf{(2)}$$

Steps (1) and (2) come from :

(1) $\hat{T}(s, a, f) = 0$; $V^i(w) = 0$ and $\forall s' \neq f, w, \hat{T}(s, a, s') = T(s, a, s')$

(2) $T(s, a, f) = 0, T(s, a, w) = 0$ because $s \notin U$.

□

Lemma 2 *For any state in U, the Q value of the optimal policy is equal in the old and the new model:*

$$\forall s \in U, \hat{Q}(s, \pi(s)) = Q(s, \pi(s))$$

Proof. We proceed by induction on the horizon i :

- $i = 0 : \hat{Q}^0(s, \pi(s)) = R(s, a) = Q^0(s, \pi(s))$
- We assume the Induction Hypothesis:

$$\hat{Q}^i(s, \pi(s)) = Q^i(s, \pi(s)) \Rightarrow \hat{V}(s) = V(s)$$

$$\begin{aligned}
\hat{Q}^{i+1}(s, \pi(s)) &= R(s, \pi(s)) \\
&+ \quad \gamma \left[\sum_{s' \notin U} \hat{T}(s, a, s')\hat{V}^i(s') + \sum_{s' \in U} \hat{T}(s, a, s')\hat{V}^i(s') \right] \\
&= \quad R(s, \pi(s)) + \gamma \sum_{s' \notin U} T(s, a, s')\hat{V}^i(s') \quad \textbf{(1)} \\
&= \quad R(s, \pi(s)) + \gamma \sum_{s' \notin U} T(s, a, s')V^i(s') \\
&= \quad R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, a, s')V^i(s') = Q^{i+1}(s, \pi(s))
\end{aligned}$$

Step (1) come from:

(1) $\forall s \notin U, \hat{T}(s, a, s') = 0; \forall s \in U, \hat{T}(s, a, s') = T(s, a, s')$

□

We show that algorithm 1 selects the good set of states.

Theorem 1 *The policy of a state s not belonging to U remains optimal without an update operation.*

Proof. We prove $\forall a \in A, s \notin U, \hat{Q}(s, \pi(s)) \geq \hat{Q}(s, a)$. By definition of π, we know that $\forall a \in A, s \notin U, \hat{Q}(s, \pi(s)) \geq \hat{Q}(s, a)$. With lemma 2, $\hat{Q}(s, \pi(s)) \geq Q(s, a)$ and with lemma 1, $\hat{Q}(s, \pi(s)) \geq Q(s, a) \geq \hat{Q}(s, a)$. Then $\hat{Q}(s, \pi(s)) \geq \hat{Q}(s, a)$. We can deduce that the policy of a state not belonging in U remains optimal without updating. □

If there is more than one recommendation, there is only one set U of updated states. This set is the union of all the computed updating sets. Our detection algorithm of updated states is designed to improve efficiency of the value iteration algorithm: instead of sweeping all the updated states without order, we use the order defined by our detection algorithm. This allows us to optimize the number of iterations for a state, because in a first step we compute the states that are highly connected with the other states to update. (Wingate and Seppi 2006) studies this methods for improving efficiency of the value iteration algorithm. We show the influence of this method in the experiments section.

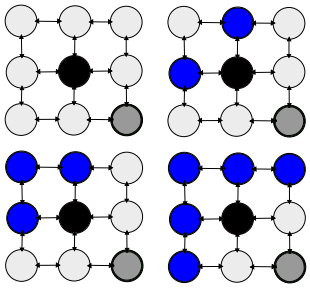

Figure 2: Example of the algorithm for updating states. The goal state is bottom left, the state marked as necessarily available is in middle. We can see the stages of the propagation of states to update with the algorithm described above.

Algorithm 2: Desirable States

Input: f: the recommended states, G: the policy graph
Data: P: a stack of vertex
Data: $Marked$: list of marked vertex
Result: U: the set of states to update
We add to P the vertex marked by the user.
$P \leftarrow P \cup \{s\}$
while P *not empty* **do**
  $s = pop(P)$
  $marked \leftarrow marked \cup s$
  foreach $v \in$*neighbor(s) and* $v \notin marked$ **do**
    $P \leftarrow P \cup \{v\}$
  if $EV(v) < EV(s)$ **then**
    $U \leftarrow U \cup \{v\}$

Updating the set of desirable states

The human recommends a desirable state as a weak constraint: this does not affect all the states, but only a subset of them. In our approach a recommendation of desirable states has been formalized as a goal-oriented decision process For a given Desirable State (DS), we compute the set of states U affected by the DS. First, we don't add in U the states with a smaller value than the one of s (the state of the agent). Those states won't be visited and don't need to be updated. Second, we don't add states with a bigger value than DS because their policy will not change.

Let s be a desirable state defined by the human and R_c the new reward function of the agent. We formalize a recommendation of the human on a state s by changing the reward to reach s with $\forall a \in A, R_c(s,a) = R(goal, a)$ where $goal$ is the state with the highest reward. A limited number of states will be affected by this reward. They will be in the set U and respect the following conditions satisfied:

c1 : Their probabilities to reach s is not null.

c2 : Their expected values are less than the one of s.

To calculate the set U, we search all the states respecting conditions **c1** and **c2** as shown in Figure 2. If the probability of a state s to reach s' is null, then its expected value remains unchanged. Moreover, if a state s' has a higher expected value than the one of s, this means that the agent is closer to a goal in s' than in s. Then, the agent tries to not reach s. Then, the list U computed with algorithm 2 contains all the states which must be updated using these conditions.

We guarantee that the agent does not go to the nearest recommended state and cycles there forever by not adding the state f in U. If there is more than one recommendation on desirable states, we consider them respecting the order of the recommendation. However, the list U of each state has to take into account the states already updated. To do that, we compute the states that respect the previous conditions and that have not been updated with another recommendation. After that we execute algorithm 2 and we don't take into account the state already updated. If the human gives contradictory recommendations, we can compute a bad policy. For example, the human gives two recommendations to an agent leading to two opposite paths.

Experimentation

We developed two types of experiments to present the quality of the solution and the complexity of our two algorithms. For this purpose, we consider an agent moving in an environment and which reaches a goal state. It perceives the world but some walls in the environment are difficult to distinguish. When it acts, it has a low probability that the desired action is not executed as expected. We consider stochastic actions, left, right, up, down and stay.

The quality of the solution

We implemented the problem with the recommendations given by the human. First, the human recommended whether a state is undesirable. Then, we implemented the problem when the human recommends a state as a desirable state.

The undesirable recommendations We developed several experiments depicted in Figure 3. Twenty-one states have been updated because of the human's recommendations. In this example, the policy remains unchanged in only three states among the set of updated states. In theorem 1, we stated that the updated policy with this method gives the optimal policy. Our experiments confirm this statement.

The desirable recommendations Figure 6 shows how states are updated according to the recommendations.

The gains on computation time

We define a complexity parameter which is the distance from an agent to the goal. This distance is the Manhattan distance of a state relative to the objective. It is the average number of states visited before reaching the goal. There is 10% of the state space with a distance less than 5. In this part, we show the benefits of using our method to update the policy according to the human recommendations.

The undesirable recommendations The number of updated states in U depends on two parameters: the number of recommendations and the distance of the recommended state to the goal. When there are multiple recommendations, the set U of each recommendation is merged into one, so the number of recommendations has a minor impact than the distance of the updated states. Figure 4 shows the impact of

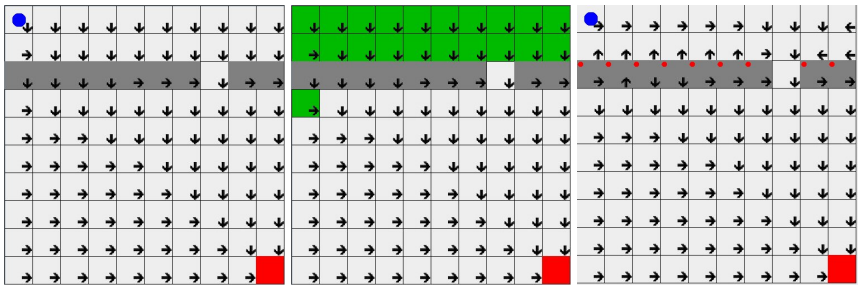

Figure 3: a) Initial policy of an agent. b) The set of updated state. c)The initial policy is updated with human recommendation.

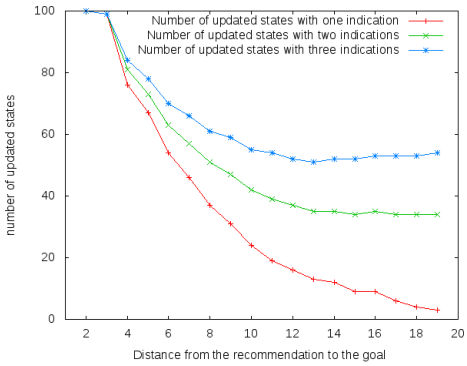

Figure 4: The influence of the distance on the undesirable recommendations

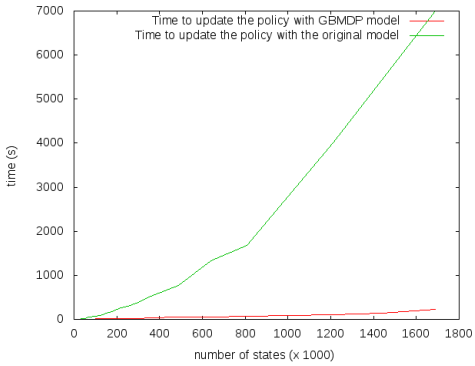

Figure 5: The computation time relative to the states number

the distance on the number of updated states. In this problem, there is a 90 % chance to update less than 60 states. So the average reduction of state space to update is proportional to the distance to the goal. Figure 4 also shows the evolution of the state number where there are several recommendations.

While updating the policy, each iteration will be at least 40 percent faster than computing the overall policy. In the worst case (with a distance of 1 to the goal), the complexity of the algorithm is in $O(\mid S \mid^2)$. The complexity of an iteration pass is $O(\mid A \mid \cdot \mid S \mid^2)$. We develop some experiments to compare our method to the traditional one. For the following experiments, we compute (in the worst case), the number of Bellman backups with 100 iterations, 5 actions and 100 states. The number of backup with the traditional method is 5000000 with those parameters. With our method, we obtain:

Recommendation number	1	2	3
backup number	1371500	2027800	2704100
benefits (%)	72.57	59.44	45.91

Experiments in real states space We developed experiments to show how GBMDP is beneficial compared to the classical approach. To update a policy, the classical approach consists of a value iteration algorithm applied to all states. We show with the previous experiments that the number of updated states for the GBMDP model depends on the position of the state. For this experiment, we choose a forbidden state in the middle of the state space (50% from the top and 45% from the left). We use the previous benchmark in which we extend the size. The differences between our model and the classical one are described in Figure 5.

Here we show that the GBMDP approach is much faster than the classical one. On small problem (200 000 states), the agents updates its policy in 16 seconds with GBMDP while an agent using a classical approach needs more than 3 minutes and 30 seconds. Moreover, on large instances (up to 1 700 000 states), our approach manages to keep the computation time under 4 minutes, while the classical approaches suffer from a combinational explosion (2 hours).

There are two reasons for this difference in computation times. With the classical methods, the number of updated states is bigger than the number obtained with the GBMDP model. Additionally, the iteration number is bigger in the classical method. In fact, our detection method allowing the agent to detect the states to update, find all states in specific order. U is sorted according to the number of states to be reached with nonzero probability. U is sorted in a decreasing order of this number. For example, the human recommendation for 40000 states takes 1 second to update with GBMDP model and 9 seconds for the classical model. For GBMDP model, there are 8460 states updated and 54 iterations, and for the classical model, there are 40000 states updated and 465 iterations. The classical model uses 40 times more Bellman backups than the GBMDP model. Morever, the time needed by GBMDP to detect which states need to

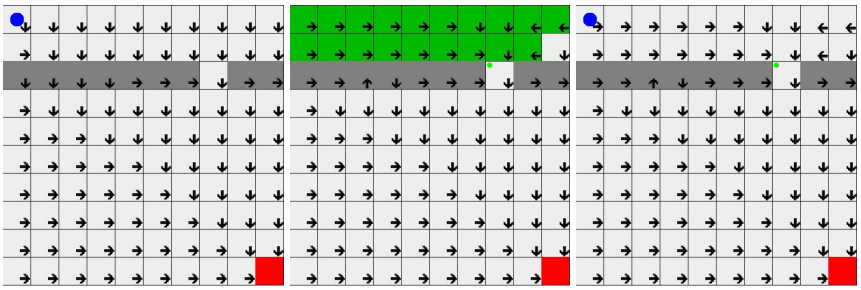

Figure 6: a) Initial policy of an agent. b) The set of updated state. c)The initial policy updated with the human recommendation.

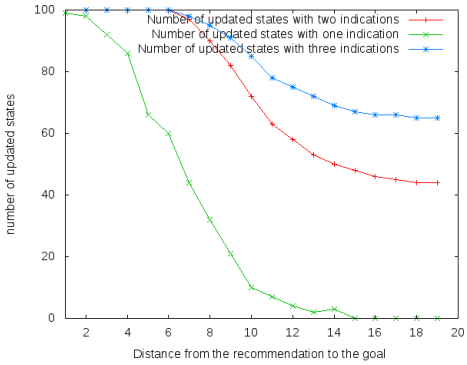

Figure 7: The influence of the distance on the desirable recommendation

be updated is negligible (complexity $O(|S|)$.

The desirable recommendations The number of states to update in U depends on two parameters: recommendations number and the Manhattan distance of a state relative to the objective. Those two parameters have a big influence on the complexity. We developed experiments depicted in Figure 7.

With multiple human's recommendations, the algorithm in Figure 2 is executed one time for each recommendation. For one recommendation, the average number of states updated is smaller than for the undesirable states. In 90 % of our experiments, we have less than 60 states to update. The number of recommendations given by the human increases the size of the list of states to update. When there are several recommendations, in some case, the solution is not good because we have to recompute the entire policy. The complexity of the algorithm is in the worst case, $O(\mid S \mid^2)$

Conclusion

In this paper, we discussed the problem of introducing human recommendations in the policy of an autonomous agent to increase its performances and robustness. Such recommendations are required in situations where the full autonomy is not desirable or not feasible. For this purpose, we present GBMDP where human can give two kinds of recommendations. He can recommend desirable states to the agent (in order to improve its performance), or undesirable states to avoid difficult situations which could lead to a major failure. We presented two algorithms inspired from value iteration where we find the set of states to update and we prove that the policy remains optimal in the other states. Experimental results show the effectiveness of our approach and the benefit of its use in terms of solution quality and computation complexity. Future work will concern a new direction where the human recommendations can be seen as a partially defined policy that the agent should optimally complete.

References

A.Mouaddib, S.Zilberstein, A.Beynier, and L.Jeanpierre. A decision-theoretic approach to cooperative control and adjustable autonomy. ECAI, 2010.

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

G. Cheng and A. Zelinsky. *Supervised autonomy: A framework for human-robot systems development*. Springer, 2001.

J.W. Crandall and M.A. Goodrich. Experiments in adjustable autonomy. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, 2002.

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*, 2005.

Gregory A. Dorais, R. Peter Bonasso, David Kortenkamp, Barney Pell, and Debra Schreckenghost. Adjustable autonomy for human-centered autonomous systems. In *Proc. of IJCAI'99 workshop on adjustable autonomy*, 1999.

V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and SX Zhang. A multi-agent system for intelligent environment control. 1999.

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

D. Fox S. Thrun, W. Burgard. *Probabilistic Robotics*. 2005.

P. Scerri, D. Pynadath, and M. Tambe. Adjustable autonomy in real-world multi-agent environments. 2001.

P. Scerri, D.Pynadath, and M. Tambe. Towards adjustable autonomy for real world. *JAIR*, 2002.

D. Wingate and K.D. Seppi. Prioritization methods for accelerating mdp solvers. *Journal of Machine Learning Research*, 6:851, 2006.