Integrating Human Recommendations in the Decision Process of Autonomous Agents

Nicolas Côté, Maroua Bouzid, Abdel-Illah Mouaddib

Integrating Human Recommendations in the Decision Process of Autonomous Agents

Nicolas Côté - Maroua Bouzid - Abdel-Illah Mouaddib
GREYC - CNRS (UMR0672), Université de Caen, Basse-Normandie, ENSICAEN
{nicolas.cote, maroua.bouzid, abdel-illah.mouaddib}@unicaen.fr

Abstract

In order to help agents in a difficult situation, we integrate the human in the agent's decision process. We develop a new model called Human Help Provider in a Markov Decision Process (HHP-MDP). We define HHP-MDP as a middle ground between an autonomous agent and a teleoperated agent called adjustable autonomy. The global approach of HHP-MDP is based on (1) a recommendation is translated into a partial policy, (2)then, this partial policy is translated into transition and reward functions associated to the agent Markov Decision Process. This new model is quickly integrated into the agent's decision process.

1. Introduction

This paper is motivated by the problems where robots deployed in complex environments meet many difficulties coming from their limited ability to perceive or to act in these environments. Because of these limits and in some sensitive applications, a full autonomy is not desirable and sometimes computationally not feasible. In order to extend their ability, robots should take advantage from an external entity such as humans which can play a role of an operator (sending orders) or a companion sharing the mission. With external helps, robots can overcome many difficulties such as the environment uncertainty or the human preference changes.

In this paper, we propose a new model which allows the agents to overcome their limited capacity by sending them human's recommendations to act when the quality of their behavior is degraded. The agents evolve autonomously in a stochastic environment. To formalize the problem of deriving a behavior policy, we use a Markovian Decision Process (MDP) which assignes for each state an action to execute. We assume, in this paper, that the human observes the agents evolving in the world, and detects that an agent meet difficulties in some situations. In order to help him, we propose that the human sends recommendations.

Moreover, we focus on two situations where the human sends recommendations: he gives recommendations (1) to help the agent to overcome difficult situations due to the lack of information or (2) to reach some states respecting particular preferences. For these two cases, the cost is the human's time spent for helping one agent and depends on the agent's situation. This cost is less than that due to the situation where the agent the human teleoperates the agent, but higher than that where the agent is autonomous. In both situations, the cost of help is very low and thus she is able to send recommendations to more than one agent without considering their models nor their current states.

In this paper, we address the problem where the agent cannot react to unexpected events and can not take into account modification into its models. From its point of view, the agent has an optimal behavior, but human has more information about the situation, and from his point of view, the agent's behavior is not optimal in this situation. The purpose of this paper is to allow a human to share his perception model with the agent, in order to improve its behavior.

The difficulty of the human to help the agent comes from the fact that some unexpected events can require to modify the agent's policy at a large number of state. We introduce our approach *HHP-MDP* that allows a human to give recommendations to the agent to react properly to the unexpected events without specifying the policy at all concerned states. There is several assets of this approach:

- We give a framework which formalize human-agent recommendation and can be adapted to many types of recommendations.
- We reduce the agent's autonomy when its behavior is not desirable.
- We decentralize the recommendations integration in order to be executed in parallel on several processors.

In the following, we explain how human recommendations are integrated in the agent's decision process. To this aim, our approach is decomposed into three steps: (i)From *Human's recommendations* to partial policy: we present what is a recommendation and how a recommendation can be defined as a partial policy. In order to illustrate our explanation, we present three recommendations according to our *UAV-an* problem.(ii) From partial policy to a new model: We explain how the partial policy is translated into a new agent's model (a new transition and rewards function in a subset of states). (iii) Integrating a new model: we explain how the agent integrates a new model or a partial policy in its decision process. Before presenting our contributions, we present several related works in the next section.

2. Related Works

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 different approaches related to our model.

2.1. Human Robot Interaction

Human-Robot Interaction was widely studied in the literature. Our model is based on adjustable autonomy, as described in Bradshaw *et al.* [2003], which allows us to dynamically change the agent's autonomy. In their work, the adjustable autonomy allows the agent to change its autonomy from teleoperation to full autonomy.

Brooks introduced some other kinds of autonomy in Brooks [1986] (studied further in Goodrich *et al.* [2001]) and demonstrated in Dorais *et al.* [1999] the advantages of adjustable autonomy compared to traditional approaches.

When an agent reaches a difficult situation, the human teleoperates it. In the same way, Lee *et al.* [2010]; Wang *et al.* [2009] describe a problem involving multi-humans and multi-robots teams where humans observe agents and assist them during the difficult situations.

2.2. HRI and MDP

In this subsection, we present several works where the human can interact with the agent in order to improve its strategy to accomplish its goal, where the agent models its environment with MDP or POMDP (Partially Observable MDP). In Mouaddib *et al.* [2010], an MDP based approach is described and which deals with agents asking for teleoperation to the humans. However, this model suffers from several limitations (for example, the latency of the model is high and it deals with only one robot with a permanently available human).In Armstrong-Crews and Veloso [2007], a model OPOMDP was introduced to plan human help using partially observable MDPs, but this model makes the assumption that the human is permanently available and only plays the role of an observation provider. All these few works are proposed by the MDP community using some HRI methods. In our approach, we use the MDP properties to introduce a new Human Robot Interaction approach. In the next section, we present the human's recommendations that our approach can take into account in the next section.

3. Human's recommendations

In this paper, we consider that the agent has a partial observability on the environnement while the human has more information on the situation. To augment its observability, the human shares his information with the agent by sending his recommandations. These recommandations should help to improve the current policy of the agent. In the following, we present what is a recommendation, and we introduce three kinds of recommendations useful for sharing human's help to the agent in *UAV-an* problem.

3.1. What is a recommendation?

A robot has a policy π_a considered optimal given its incomplete model M_a. This policy could be not optimal from the human point of view because he has more information and richer model M_h. To improve the policy π_a, the human sends recommandations leading them to derive a new policy π'_a to overcome the difficulty. Different recommandation types can be considered:

- **Primitive Recommandations** *PR* : these recommandations assign an action a to a state s such that $PR = (s,a)$. For example: turn left at the next crossing.

- **General Recommandation** *GR* : these recommandations are a partial policy π_p defined by specific states in the subspace. Such recommandations can be seen as a set of primitive recommandation such as $GR = (S, recommandation) = (s_1,a_1),(s_2,a_2),...,(s_n,a_n)$. For example : follow the wall so $GR = (wall, follow) = \pi_{follow}$ such that $s \in adjacent(wall)$.

3.2. Transforming general recommandation into primitive recommandations

3.2.1. Giving a subgoal recommendation. We define a subgoal as a set of target states. The agent must reach one of its subgoal before its initial goal. In our *UAV-an* problem, the human subgoal ensures that the agent takes the right direction by reaching the subgoal first. For example, in Figure 1 the agent (in black circle) has to reach its goal (in black rectangle). In the environment, the agent evolves in grid by executing four actions: left, right, up and down. It evolves in the white square with 0.1 probability that its action fails, and it evolves on the snow in blue square with a probability of 0.8 of deviating from its trajectory. The agent has computed its optimal policy which takes only the white square into account. The sensor of the agent are unable to detect snow. In order, to improve its policy, the human gives two subgoals (in gray rectangle) to the agent in order to avoid the snow. The agent changes its initial policy which takes it through the snow by reaching the two subgoal given by the human. When the human gives a subgoal, this recommendation has to be translated into a primitive recommandation, in order to be integrated in the agent's decision process.

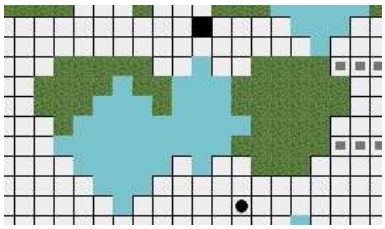


Figure 1. Giving a subgoal

Definition 1 *A subgoal is a recommendation where the agent has to reach at least one state of the set of states associated to subgoal before its goal.*

More formally: the human gives a subgoal s_g when the agent is in a state s_i. We define π^{s_g} which allows the agent to reach its subgoal. To this aim, we change the reward function where $\forall s \in s_g, R'(s_p) = R(g)$ and $\forall s \in S \backslash \{s_g\}, R'(s) = R(s)$. Moreover, the primitive recommandation associated to s_g contains a state s iff $\forall s_p \in s_g, \exists s \in S$ where $P(s, \pi^{s_g}, s_p) > 0$.

When we want to integrates primitives recommendation into the agent decision process, the traditionnal approach is simple: we execute the value iteration algorithm without changing the given partial policy. This computation shows a lack of performance, and this approach executes some unnecessary operations:

- We compute the expected value for each recommended states, even if we know the agent's policy.
- We update the set of all states, even if these states are not modified.

We study these two problems in the following, and we present an approach to solve them, and improve significantly the naive approach.

4. Efficient Algorithm for Human Integration

In this section, we present the algorithm translating the recommended partial policy into a new model of transition and reward functions. These modifications avoid the expected value computation for some unecessary states. This model is more easily integrated into the agent's decision process than the partial policy integration, because unlike naive approach, IHR-MDP does not compute the expected value for each state where the policy was given, but only in the necessary subspace. Unnecessary computation are then avoided. Thus our new algorithm is based on two main steps: computation of the new model (T' and R') and the integration of this new model in the agent's policy. The rest of this section is focused on the first step and section 5 is dedicated to the integration step.

To the end of the first steps, we define a set U as the human's recommended states where for each state in U, we know the given human policy π_h where $\forall s \in U, \exists a \in A, \pi_h(s) = a$.

The intuition of our algorithm is: for each state where the policy is given (U), we compute the new reward function R' and a new transition function T' according to the human policy π_h. In order to integrate the human's recommendation, the agent has to integrate these functions R' and T' to computes a new π_G. To this purpose, we define a set of entry states $ES(r)$ of a recommended state $r \in U$: $s' \in ES(r)$ iff $\exists s' \in S$, $s' \neq r$, $T(s', \pi_a(s'), r) > 0$. In the same way, we define a set of output states $OS(r)$ of a recommended state r: $s' \in OS(r)$ iff $\exists s' \in S$, $s' \neq r$, $T(r, \pi_h(r), s') > 0$. Note that ES is defined by π_a and OS is defined by π_h.

After the computation of sets OS and ES for a state r, we compute a transition T' and a reward R' function for each state in $ES(r)$ to reach a state in $OS(r)$.

The transition or reward function is easy to compute when no cycle exists, but some cycles can occur in the given human's partial policy. A recommanded state leads to a cycle when the following conditions holds : if $\exists o \in OS(r), \exists e \in ES(r)$ where $T(r,a,r) > 0$ or $T(o, \pi_a(o), r) > 0$ or $T(r, \pi_h(r), e) > 0$ or $T(o, \pi_a(o), e)$ means that there is a cycle. Although this policy may

contain small cycle number, the algorithm resolution has to integrate an iteration function to give the ε close result.

Our aim is to compute the probability for an agent to reach out states when it reaches entry state. To this end, we compute the new transition $T'_r(i,e)$ and reward $R'_r(i,e)$ functions for each state in $r \in U$, the state $i \in \{r\} \cup OS(r)$ and $e \in ES(r)$ with the following equation:

$$T'_r(i,e) = \gamma \cdot \sum_{\forall s \in OS(r)} T'_r(s,e) \cdot T(r, \pi_h(r), s) \quad (1)$$

this equation is computed until $T'_r(i,e) - \gamma \cdot \sum_{\forall s \in OS(r)} T'_r(s,e) \cdot T(r, \pi_h(r), s) < \varepsilon$. In the same way, we define $R'(s)$ the reward function as follows:

$$R'_r(i,e) = R(i) + \gamma \cdot \sum_{\forall s \in OS(r)} T(r, \pi_h(r), s) \cdot R'_r(s,e) \quad (2)$$

with $0 < \gamma < 1$ the discounted factor and we initialize $T' = T$ and $R' = R$. This equation is computed until $R'_r(i,e) - [R'(r) + \gamma \cdot \sum_{\forall s \in OS(r)} T(r, \pi(r)_h, s) R'_r(i,e)] < \varepsilon$.

Finally, we compute $T_r(i,e)$ which describes the probability to reach the state i with the recommendation r for an entry state e and takes into account the possible cycles. In order to compute the probability to reach a state in OS from a state in ES, we replace the initial transition T and reward R function for each entry states of the recommended states: $\forall e \in ES(r), o \in OS(r), T(e, \pi_a(e), o) = T(e, \pi_a(e), r) \cdot T'_r(o,e)$ and $T(e, \pi_a(e), r) = 0$.

In order to improve the performance of our algorithm, we aggregate some states in a partition G. With this approach, we reduce the new transition and reward computation by computing only for the partition G. The integration process is also improved.

We define a partition $G = \langle G_1, ..., G_n \rangle$ of S. G_i is a set of states. A state $s \in G_i$ iff $s \in U$ and $\exists g' \in G_i$ where $T(s, \pi_h(s), g') > 0$ or $\exists g' \in G_i$, $T(g', \pi_h(g'), s) > 0$. We define ES and OS over G_i rather than s as described previously as follows: $s' \in ES(G_i)$ iff $\exists s' \in S, s' \notin G_i$,$\exists s \in G_i$, $T(s', \pi_a, s) > 0$. In the same way, we define $s' \in OS(G_i)$ iff $\exists s \in S, s \notin G_i$,$\exists s' \in G_i$, $T(s', \pi_a, s) > 0$.

When we know the agent's policy, we can compute the probability and the reward for an agent in a group G_i to leave this group. In our approach, our goal is to reduce the state space for the agent by using given human's partial policy. For this end, we compute for each entry state in a partition G_i, the reward and the probability to reach a state out of G_i.

We use Equation 1 and 2 in order to compute $T_i(r)$ which describes the probability to reach the state $i \in G_i \cup OS(r)$ with the recommendation r and takes into account the possible cycles. Then, we compute the probability $\forall e \in ES(G_i), \forall o \in OS(G_i) T'(e,o)$ and the reward $R'(e)$. Then, we replace the initial transition T and reward R function for each entry state of the recommended states: $\forall e \in ES(G_i), o \in OS(G_i), T(e, \pi_a(e), o) = T(e, \pi_a(e), r) \cdot T'_r(o,e)$ and $T(e, \pi_a(e), r) = 0$.

In Figure 2, we present a small example of partial policy given by the human. The policy graph is represented where a vertex is a state and an edge is the transition probability. The states 1, 2 ,3 and 4 are in the set $ES(g_i)$. The states o_1, o_2, o_3, o_4, o_5 and o_6 are in the set $OS(g_i)$. We apply our algorithm to compute the transition function for the set g_i of the partition G. The table of Figure 2 shows the results for each state in $ES(g_i)$ and for each state in $OS(g_i)$. For example, when the agent reaches the state 1, it has a probability of 0.6 to reach o_1, 0.2 to reach o_2 and 0.2 to reach o_3 when it follows the human's recommendations. To improve our algorithm efficiency, we can avoid the computation of the states in $OS(g_i)$. In this aim, we take into account the states reachable by the states in $OS(g_i)$ when the policy π_h is applicable. With our approach, the agent avoids the expected value computation of 11 states for this small policy. This number depends on the model transition function and the human's partial policy.

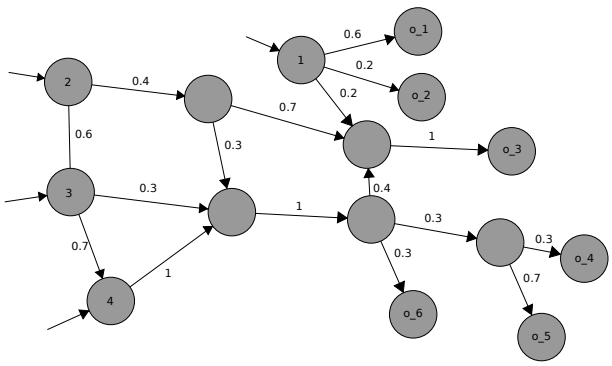
In our approach, we have presented how the human gives recommendations and how we translate these recommendations into a partial policy. Then a new agent's model is extracted from the human's recommendations in order to improve the integration time. Finally, in the next section, we present how the agents integrates a new model in its decision process. Another comment of this approach is that in general, the number of iteration is low. In order to be a good recommendation, the human's partial policy may avoid the cycle.

5. Human policy integration

In this section, we present how the agent integrates a new model computed with the algorithm described previously. We explain what are the conditions under which the agent's has to change its previously computed policy and how the agent updates its policy by taking the human's recommendations into account. We denote two steps in order to integrate the human's recommandation: how to compute the set of updating states affected by the recommandation and how to update the set of updating states.

5.1. The updated states definition

In this subsection, we present how to search the set D of states influenced by the human's recommendations. The states in D will needs to be updated when



$ES(g_i)$	$OS(g_i)$	T
1	o_1	0.60
1	o_2	0.2
1	o_3	0.2
1	o_4	0
1	o_5	0
1	o_6	0
1	w	0

$ES(g_i)$	$OS(g_i)$	T
2	o_1	0
2	o_2	0
2	o_3	0.568
2	o_4	0.0648
2	o_5	0.1512
2	o_6	0.216
2	w	0

$ES(g_i)$	$OS(g_i)$	T
3	o_1	0
3	o_2	0
3	o_3	0.4
3	o_4	0.09
3	o_5	0.21
3	o_6	0.3
3	w	0

$ES(g_i)$	$OS(g_i)$	T
4	o_1	0
4	o_2	0
4	o_3	0.4
4	o_4	0.0900
4	o_5	0.21
4	o_6	0.6
4	w	0

Figure 2. Example of Space State Reduction in order to improve the agent's integration time

the agent integrates the human recommendations.

The updated algorithm is different because, the agent can use its policy and its expected value previously computed. For the agent point of view, its computed policy is the optimal policy. When the human gives recommendations on several states, she expresses that the agent has not all informations to compute the optimal policy. The expected value of the new global policy π^G will always be lower than the expected value of π^A from the agent point of view.

Our aim is to update the agent policy by updating only the minimal subset of appropriate states (states which are affected with the recommendations). For this purpose, 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 probability to transit from s_1 to s_2 by following the policy is not null. The following Proposition concerns the choice of updated states. We define $Parent(s,\pi(s))$ as a function in which $Parent(s,\pi(s)) = \{s' \in S \mid T(s',\pi(s),s) > 0\}$. Moreover, we define a rational policy π_h when $\pi^h(s) = \text{argmax}_{a\in A, s\in U} Q(s,a)$.

Proposition 1 *When π^H is good, π^G is optimal.*

Proof. π^G is optimal when $\forall s \in S$: $\pi^G(s) = \text{argmax}\, a \in A, Q(s,a)$. • $\forall s \notin D, \pi^G(s) = \underset{a\in A}{\text{argmax}}\, Q(s,a)$. That means π^G is optimal.

• $\forall s \in D$, $\pi^G(s) = \pi^H(s)$. π^H is rational means that π^H is consistent and $\pi^H(s) = \text{argmax}_{a\in A, s\in U} Q(s,a)$, thus $\forall s \in D, \pi^G(s) = \underset{a\in A}{\text{argmax}}\, Q(s,a)$. Consequently $\forall s \in S : \pi^G(s) = \text{argmax}_{a\in A} Q(s,a)$. □

In order to find all the updated states in the set D, we make a Breadth First Search from each states recommended in U in the policy graph of π^A. We add a state in D if another action is not better than the optimal action with the new model.

More formally, we denote previously π^A as the optimal policy. Let π^{A^n} the n-best policy where $\pi^{A^n}(s) = \text{argmax}_{a\in A\setminus\{\pi^{A^{n-1}}(s)\}} Q(s,a)(\pi_n < \pi_{n-1} < ... < \pi_*)$. We deduce that $\pi^A = \pi^{A^1}$. Let $Q'(s,a)$ the Q-value using the new transition and reward function where $Q'(s,a) = R'(s,a) + \sum_{s'\in OS(s)} T'(s,a,s')V(s') + \sum_{s'\notin OS(s)} T(s,a,s')V(s')$

In order to reduce the state space of reduced states, we present the following proposition.

Proposition 2 *A policy $\pi^A(s)$ is optimal in s with $s \notin U$ if and only if $1 < n <\mid A \mid$:*

$$\sum_{s'\in S} T(s,\pi^A,s')[V(s') - V'(s)] > Q(s,\pi^A(s)) - Q(s,\pi^{A^n}(s))$$

Proof. We know that π^A is optimal if and only if $Q'(s,\pi^A(s)) > Q'(s,\pi^{A^n}(s))$

$$\begin{aligned}
&\Leftrightarrow && Q'(s,\pi^A(s)) > Q(s,\pi^{A^n}(s)) \\
&\Leftrightarrow && Q(s,\pi^A(s)) - \textstyle\sum_{s\in S} T(s,\pi^A(s),s')V'(s) \\
& && -Q(s,\pi^{A^n}(s)) > 0 \\
&\Leftrightarrow && \textstyle\sum_{s\in S} T(s,\pi^A(s),s')[V(s) - V'(s)] > Q(s,\pi^A(s)) \\
& && -Q(s,\pi^{A^n}(s))
\end{aligned}$$

□

Discussion: The purpose of proposition 2 is to avoid as early as possible the useless states to update. We define a function called *possibleChange*(s) that gives a boolean if the policy change at states s. A state can added to D and the optimal policy doesn't change, but we detect this in the integration phase. We know that $Q'(s,\pi^{A^n}(s)) < Q(s,\pi^{A^n}(s))$, so we make the approximation $Q'(s,\pi^A(s)) > Q(s,\pi^{A^n}(s))$ in order to reduce the computation time: the agent has already computed $Q(s,\pi^{A^n}(s))$. For each action, the agent evaluates only $\sum_{s\in S} T(s,\pi^A(s),s')V'(s) - Q(s,\pi^{A^n}(s))$. The transition often concerns a small subset of S (in our experimental test, this subset size is less than 4). We define $T_{reachable}$ the maximum number of states reachable with

any transition. The computation of $possibleChange(s)$ has a complexity of $|A| \cdot T_{reachable}$. We give the following definition based on proposition 1 and 2:

Definition 2 *A state $s \in D$ if and only if:*

- $s \in Parent(s', \pi(s'))$ *where* $s' \in D$
- $\sum_{s' \in S} T(s, \pi^A, s')[V(s') - V'(s)] > Q(s, \pi^A(s)) - Q(s, \pi^{A^n}(s))$

To compute the set of updated states D, we use an algorithm which finds all the states respecting the presented constraints.

5.2. Integrates the human's recommendation

In this section, we present how the agent update its policy already computed with the given human's recommendations. We present two different algorithms, one designed to integrate a partial policy, and another designed to integrates a model composed by a new transition and reward function. We present two different algorithms based on the updating state process using two steps : (i)defining a set of states D to update, (ii)updating the states in D and the state where the human gives her policy (U). The partial policy integration algorithm is mainly founded on this updating process while the new model integration algorithm improves the IHPP by combining the updatnig states process with the new model computation and integration.

The drawback of the IHPP algorithm is that the agent has to compute the expected value for the states where its policy is defined by human's recommendations (states in U). This computation is important in order to compute the expected value of the changing states (states in D). In this purpose, the computation of the new transition and reward function avoids the computation of the expected value of the states in U.

In the Integration Human's Transition and Reward function algorithm (IHTR), we can reduce the size of the updating states by taking advantage of the new transition and reward function. In the *Efficient Algorithm for Human Integration* section, we show how the agent computes a new transition and reward function. We take advantage of the new function previously computed in order to avoid the agent's policy computation. Moreover, we use this function to reduce the states to update: only the subset D is updated.

The advantage of the IHTR algorithm is that the agent integrates more quickly the new transition and reward function because the expected value of the set of state U is not computed. This difference is crucial, because the iteration number (n in our algorithm) is smaller in the IHTR than in IHPP algorithm.

6. Conclusion

In this paper, we have presented *HHP-MDP*: a new approach in order to integrate quickly human's help in the agent decision process. To this aim, we introduce the definition of recommendation which is the human's help given to the agent. We distinguish between these recommendations by three useful recommendations types for the *UAV-an* problem. Also, we present algorithms and methods which allow the agent to integrate quickly the human's recommendations. Our future work is to extends *HHP-MDP* with fuzzy recommendations and apply the recommendations into a complex model in order to solve problem of helping the agent in difficult situations. Another future works is to evaluate the human's recommandation and detect the inconsistancies.

References

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

J.M. Bradshaw, M. Sierhuis, A. Acquisti, P. Feltovich, R. Hoffman, R. Jeffers, D. Prescott, N. Suri, A. Uszok, and R. Van Hoof. Adjustable autonomy and human-agent teamwork in practice: An interim report on space applications. *Agent Autonomy*, 2003.

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

G. Dorais, R.P. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost. Adjustable autonomy for human-centered autonomous systems. pages 16–35, 1999.

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

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.

A.-I. Mouaddib, S. Zilberstein, A. Beynier, and Laurent J.P. A decision-theoretic approach to cooperative control and adjustable autonomy. *ECAI*, pages 971–972, 2010.

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