A Human-Aware Robot Task Planner

Marcello Cirillo, Lars Karlsson, Alessandro Saffiotti
AASS Research Center
Örebro University, Sweden
{marcello.cirillo, lars.karlsson, alessandro.saffiotti}@aass.oru.se

Abstract

The growing presence of household robots in inhabited environments arises the need for new robot task planning techniques. These techniques should take into consideration not only the actions that the robot can perform or unexpected external events, but also the actions performed by a human sharing the same environment, in order to improve the cohabitation of the two agents, e.g., by avoiding undesired situations for the human.

In this paper, we present a human-aware planner able to address this problem. This planner supports alternative hypotheses of the human plan, temporal duration for the actions of both the robot and the human, constraints on the interaction between robot and human, partial goal achievement and, most importantly, the possibility to use observations of human actions in the policy generated for the robot. The planner has been tested as a standalone component and in conjunction with our framework for human-robot interaction in a real environment.

Introduction

Public interest in home robots is steadily increasing and people are looking at robots as new means to improve the quality of their everyday life. The aging of the population, for instance, could open a wide space for new applications (Tapus, Mataric, and Scassellati 2007). Robots could then become effective workers, precious butlers and, eventually, friendly helpers in our houses. Similar opportunities could also arise in industrial environments.

The identification of suitable interfaces for the interaction between humans and robots is only one of the many challenges that the cohabitation introduces. The presence of humans in the space where robots operate also has a profound influence on how the embodied agents should perform high level reasoning and plan their actions.

As many researchers have already pointed out, a classical AI planning system in which the robot is in control and the state of the world is only affected by the actions of the robot (Nau, M., and Traverso 2004) is not applicable anymore. In order to be effective collaborators in household environments, the robots should include the state of the humans in their control loop (Jenkins, González, and Loper 2007). We believe that, as remarked by Hoffman and Breazeal (2007b; 2007a), the interaction and teamwork of robots and humans can be greatly improved if the former can anticipate the forthcoming actions of the latter. However, our approach goes beyond the one of Hoffman and Breazeal on two points: first, we aim to forecast not only one human action at a time, but a full plan for a wider timespan. Second, we use a planning system to decide which sequence of actions the robot should perform to achieve its goals while respecting a given set of interaction constraints.

The main contribution of this article is a planning technique able to deal with the presence of the human in the environment. This technique is applied to situations in which there is a controllable agent (the robot) whose actions we can plan, and an uncontrollable agent (the human) whose action we can only try to predict. For simplicity, we assume there are only one robot and one human. Multiple alternative human plans are considered, and the domain is partially observable. Therefore the planner generates policies conditional on what the human is observed doing. Actions have duration, but for the purpose of this paper we assume that durations are deterministic.

Although the main focus of this paper is on the planning algorithm, we also describe the inclusion of our planner in a full framework for human aware task planning. This framework includes a plan recognition system for estimating the plans carried out by a person living in a small apartment, the human-aware planner itself, and an execution and monitoring system through which the robot plans are executed.

Related Work

There has been extensive work on planning with external events (as human actions can be considered). An early example is the work of Blythe (1994), which used Bayesian nets to compute the probability of success in the presence of external events. Our approach is also somehow reminiscent of early work on collaborative planning (Grosz and Kraus 1996). In the robotics field, some works have considered human-robot co-habitation. Often these works take a viewpoint which is different from the one adopted here, by focusing on aspects such as safety (e.g., within MORPHA (Graf, Hans, and Schraft 2004)), acceptable motion (e.g., within COGNIRON (Akin Sisbot et al. 2006)) or human-aware manipulation (Sisbot, Marin, and Alami 2007). The problem of task planning in the presence of humans is currently still open, although some researchers have started exploring the issue (Alami et al. 2006; Broz, Nourbakhsh, and Simmons 2008). Montreuil et al. (2007) and Galindo et al. (2008) have addressed the complementary problem of how a robot could generate collaborative plans involving a human. In our work, by contrast, the robot does not plan actions for the human, but instead tries to forecast actions and plans of the human from previous actions. Our approach also diverges from the techniques developed for plan merging (Gravot and Alami 2001), because in our case the human is not controllable and some of his actions can prompt new goals for the robot. Therefore, the actions of the human cannot be re-scheduled, and the plan of the robot cannot be created *a priori* and then rearranged.

An important property of our planning problem is that actions can be executed simultaneously and have durations, and that property has been addressed before in the literature. For instance, Mausam and Weld (2008) present an MDP model (fully observable) based on the concept of interwoven epoch search space, adapted from Haslum and Geffner (2001).

The Planner

The main feature of our human-aware planner is that it is able to consider the plan that the human is currently executing — or, more generally, a set of possible alternative human plans, called agendas — and generate a robot plan that is compliant with these agendas. The possible agendas, together with an associated probability distribution, are assumed to be estimated by a separate plan recognition module. As one doesn't know in advance which of these agendas is the actual one, the planning problem has the property of partial observability. The planner, which is an extension of PTLplan (Karlsson 2001), generates policies which are conditional on observations relating to the human's actions.

In this section we present the human-aware planner. Later in the paper we show how this planner has been incorporated into a full experimental system, including a simple plan recognition module and an execution module.

States, situations and actions

A *state* s is represented in terms of a set of state variables and their values. The set of all states is denoted S.

An *action* a has a precondition $Pr_a : S \to \{T, F\}$, a time duration $t_a \in \mathbb{R}^+$, a cost function $Cost_a : S \to \mathbb{R}^+$ and a transition function $Res_a : S \times S \to [0, 1]$ such that $Res_a(s, s')$ is the probability of going to state s' when action a is performed in state s. We assume in this paper that an action always takes the same amount of time to perform, and cannot be interrupted once started. The end state, however, can both depend on the starting state and be stochastic. Note that actions include both those performed by the human and by the robot. We use HA and RA to denote the sets of human actions and robot actions, respectively.

An *agenda* is a finite list of consecutive human actions: $(a_1, a_2, ..., a_n)$.

A *situation* is a tuple $\langle s, rt, ht, ha\rangle$ where s is a state, $rt \in \mathbb{R}^+$ is the time when the robot latest action ended, $ht \in \mathbb{R}^+$ is the time when the human latest action ended, and ha is the remaining agenda of the human. The set of situations is denoted Σ. Note that the concept of a situation can be generalized in a straight-forward manner to have more robots and more humans, but for the sake of simplicity, we only consider one of each here.

We can now define what happens when the robot performs an action a in a situation $\langle s, rt, ht, ha\rangle$ by extending the function Res_a to transitions between situations:

$$
\begin{aligned}
&Res_a(\langle s, rt, ht, ha\rangle, \langle s', rt', ht', ha'\rangle) = \\
&\begin{cases}
\sum_{s''} Res_{a'}(s, s'') \cdot \\
\quad Res_a(\langle s'', rt, ht'', ha''\rangle, \langle s', rt', ht', ha'\rangle) \\
\quad \text{when } a' = first(ha) \wedge ha'' = rest(ha) \wedge \\
\quad ht'' = ht + t_{a'} \wedge ht'' \leq rt + t_a \\
\\
Res_a(s, s') \text{ when } rt' = rt + t_a \wedge ht' = ht \wedge \\
\quad \forall a'(a' = first(ha) \Rightarrow rt' < ht + t_{a'}) \wedge \\
\quad ha' = ha
\end{cases}
\end{aligned}
$$

Here, the first part is the recursive case when the next human action a' finishes before the robot current action a, and the second part is the case when the robot current action a finishes before the next human action a' provided there is one. Note that the first situation is when the robot action is started, and the second situation is when it ends.

As an example (figure 1), consider three states s, s', s'', one robot action $clean$ such that $t_{clean} = 5$ and $Res_{clean}(s', s'') = 1$, and two human actions $watchTV$ and $eatDinner$ such that $t_{watchTV} = 4$, $Res_{watchTV}(s, s') = 1$, $t_{eatDinner} = 4$ and $Res_{eatDinner}(s'', s''') = 1$. In that case:

$$
\begin{aligned}
&Res_{clean}(\langle s, 5, 3, (watchTV, eatDinner)\rangle, \\
&\quad \langle s'', 10, 7, (EatDinner)\rangle) = \\
&Res_{watchTV}(s, s') \cdot Res_{clean}(\langle s', 5, 7, (eatDinner)\rangle, \\
&\quad \langle s'', 10, 7, (EatDinner)\rangle) = \\
&Res_{watchTV}(s, s') \cdot Res_{clean}(s', s'') \;=\; 1.0
\end{aligned}
$$

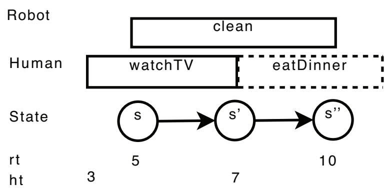


Figure 1: Example of robot and human action transitions.

Partial observability

The framework presented above is sufficient as long as we have full observability, that is we always know the current situation, including the agenda of the human agent. However, that is not always the case. Therefore, we need to introduce partial observability (Kaebling, Littman, and Cassandra 1998), by defining a *belief situation* to be a probability distribution over situations. This implies that the robot can have alternative hypotheses about the current state of the world, the human's agenda, and so on. In addition, partial observability involves *observations* as results of actions. Those are introduced in the standard way by defining a function $Obs_a : S \times O \rightarrow [0,1]$ which specifies the probability of having an observation $o \in O$ when action a is performed resulting in state s. An observation is a set of literals, that can be empty. If a is an action by the robot, then a is assumed to be a sensing action (like testing whether a door is open or closed). If a is an action by the human, on the other hand, then o is still assumed to be an observation by the robot, but an indirect observation of a or some effect of a (if the TV is switched off, the system can observe that the $watchTV$ action is over). Notice that also in the second case, the observation is dependent on the state, so we can specify that a human action only results in an observation under certain conditions.

The observation function Obs_a only concerns states, and not situations. For the latter, it becomes a function $Obs_a : \Sigma \times \Sigma \times \Omega \rightarrow [0,1]$ which specifies the probability of an *observation sequence* $\omega \in \Omega$ (where Ω is the set of all observation sequences) when a transition from one situation to another occurs. This observation sequence consists of the observations resulting from all human actions, that can provide observations, being completed between the two situations, e.g. while a is being executed. The last element of the observation sequence would be the observation resulting from the robot action (that can also be an empty observation, if a is not a sensing action). It must be noticed that some human actions are not observed by the system and give the empty observation.

$$Obs_a(\langle s, rt, ht, ha\rangle, \langle s', rt', ht', ha'\rangle, \omega) =$$
$$\begin{cases} \sum_{s''}(Res_{a'}(s,s'') \cdot Obs_{a'}(s'', first(\omega)) \cdot \\ \quad Obs_a(\langle s'', rt, ht'', ha''\rangle, \langle s', rt', ht', ha'\rangle, rest(\omega))) \\ \quad \text{when } a' = first(ha) \wedge ha'' = rest(ha) \wedge \\ \quad ht'' = ht + t_{a'} \wedge ht'' \leq rt + t_a \\ \\ Obs_a(s', first(\omega)) \text{ when } rt' = rt + t_a \wedge ht' = ht \wedge \\ \quad \forall a'(a' = first(ha) \Rightarrow rt' < ht + t_{a'}) \wedge ha' = ha \end{cases}$$

In analogy with the standard POMDP model (Kaebling, Littman, and Cassandra 1998), different observations result in different belief situations. The probability of a situation σ' in the belief situation b' resulting from an action a performed in a belief situation b with observations ω is computed as follows:

$$P(\sigma'|b') = P(\sigma'|b,\omega) = \frac{\sum_{\sigma} P(\sigma|b) \cdot Res_a(\sigma,\sigma') \cdot Obs_a(\sigma,\sigma',\omega)}{\eta}$$

The denominator η in the equation above is a normalizing factor, and is defined as $P(\omega|b,a)$ below.
The posterior probability for a certain observation sequence ω (and a corresponding resulting belief situation b') when action a is taken in b is:

$$P(\omega|b,a) = \sum_{\sigma}\sum_{\sigma'} P(\sigma|b) \cdot Res_a(\sigma,\sigma') \cdot Obs_a(\sigma,\sigma',\omega)$$

Planning problem and policy

A human-aware planning problem consists of:

1. An initial belief situation b_0, where the different situations can contain both alternative human agendas (as obtained from a plan recognition system), and different states.
2. A set of goals G to achieve in terms of constraints on states (logical formulas), with different values $V(g) \in [0,1]$ such that $\sum_{g \in G} V(g) = 1$. Each $V(g)$ represents the importance value associated to the achievement of the corresponding goal.
3. A set of interaction constraints IC not to violate referring to states (temporal logic formulas).
4. A set of robot actions RA.
5. A set of human actions HA.

A *policy* is a graph $\langle n_0, N, E\rangle$ where the nodes N are marked with the robot actions to perform there (n_0 is the initial node), and the edges E are marked with observation sequences (possibly empty). Some nodes are marked with the special actions *success* and *failure*, and these nodes are terminal.

The policy is executed by performing the action of the current node, and then selecting the edge matched with the observations that occur, following it to the next node, and repeating the process until a terminal node (*success*, *failure*) is reached.

A policy $\langle n_0, N, E\rangle$ is admissible in an initial belief situation b_0 if and only if the following conditions hold. Each node $n_i \in N$ can be assigned a belief situation b_i, in particular with b_0 assigned to n_0, such that: the preconditions of the action in n_i hold in all situations of b_i; and for each possible transition resulting from the action reaching to some other belief situation b_j and producing the observation sequence ω, there is an edge marked with ω from n_i to another node n_j that has been assigned b_j; and there are no other edges. In addition, there should be no edges from terminal nodes.

A policy violates an interaction constraint $ic \in IC$ if and only if there is a node in it with an assigned belief situation in which ic is false.

The value (success degree) of a terminal node n_j in a policy with assigned belief situations is computed as $\Sigma_{g \in G} P(g|b_j) \cdot V(g)$, and the values of other nodes are computed (maximized) according to the Bellman equations, as is done for Markov Decision Processes (Puterman 1994). The cost is computed analogously but has lower priority than the value: if there are several policies with maximal value, the one with lowest cost will be selected. The value and cost of the entire policy are the value and cost of the initial node n_0.

A policy solves a planning problem with a degree of p if it is admissible, only contains actions from RA, has a value of p (or more) for the goals in G and no interaction constraints in IC are violated.

In addition, one can add requirements for the human agendas to be completed, alternatively not completed, in the *success* nodes.

```
Procedure HumanAwarePlanning($b_0, RA, HA, IC, G, V, succ$)
1. Let $B := \{b_0\}$
2. Select a node $b \in B$ with untested actions
3. Select an action $a \in RA$ not tried in $b$
4. If $Pre_a$ hold in $b$
5.    compute the set $\{b' | P(b'|b, a) > 0\}$
      while checking $IC$ and search control
      (the human agendas are progressed here)
6.    add the new $b'$ to B
7.    If there are $b'$ for which some goals $g \in G$ hold,
8.       compute the values of those $b'$ using $V$
9.       perform Bellman update of success and cost for nodes in $B$
10. If no nodes in $B$ with untried actions or value in $b_0 \geq succ$
11.    return best policy
12. goto 2.
```

Figure 2: Algorithm for human aware planning.

The algorithm

We have extended the planner PTLplan (Karlsson 2001; Bouguerra and Karlsson 2005) for probabilistic and partially observable domains to work on our representation for human-aware planning. PTLplan is a progressive planner, starting from the initial belief state, or in the extended version: belief situation, exploring belief states/situations reachable from there until a policy with sufficient value has been found. PTLplan, which itself is an extension of TLplan (Bacchus and Kabanza 2000), uses temporal logic formulas to eliminate belief states/situations from the search space. The algorithm of the human aware planner is detailed in pseudocode in figure 2. As mentioned in the previous sections, our main extensions to the original planner are:

- The use of HA to include human actions, the addition to the belief situations of agendas consisting of such actions, and the fact that the effects of the human actions are taken into consideration when the new belief situations are computed in step 5
- The possibility to generate policies with partial goal achievement and the use of weights $V(g)$ (step 8) to prioritize the accomplishment of a subset of the goals $g \in G$
- The introduction of IC, that are checked in every state encountered while the new belief situation is computed (step 5), to avoid undesirable situations from the human side

We refer to (Karlsson 2001) for further technical details about the original PTLplan.

Performance

To evaluate the performance of the planner as a standalone component, we designed two scenarios. The first one, the vacuum cleaner scenario, is an extension of the real test that we performed with the full framewok. The second one is a purely simulated scenario, in which an autonomous table is required to serve the human when needed. Our intention was to test the performance of the planner with respect to the number of applicable actions, the number of human agendas provided as input and the number of events that compose each of the agendas. We tested each scenario first without, and then with domain knowledge. In this way, we can both see the total difficulty of each problem and what can be achieved using heuristics. All the agendas provided to the planner as an input are assumed to be equally probable.

The vacuum cleaner scenario

A robotic vacuum cleaner is deployed in an apartment. The set of goals G given to the agent specifies that each room of the apartment must be clean and that the agent must be back at its charging station at the end of the plan. The only interaction constraint IC is that the robot and the human should never be located in the same room, to avoid disturbances. The following formula describes this IC:

```
forall r: (not((robot-in=r)and(human-in=r)))
```

The agent is provided with an action set RA of four parametric actions, specified in terms of name, preconditions, effects and time. The effects include a measure of cost, which in this case represents battery consumption. An example of a robot action is:

```
name: robot-clean(r)
precond: room(r) and dirt(r) > 0
results: dirt(r):=(dirt(r) - 1) and cost:=2
time: 10
```

It should be noted that the number of action instances applicable at each step is dependent on the number of rooms of which the apartment is composed.

The initial belief situation for the planner is automatically generated according to the number of rooms in the apartment and to the human agendas that are provided at each run. The elements common to all initial belief situations can be formalized as:

$$\begin{aligned} & rt := 0 \\ & ra := 0 \\ & s := \bigcup_{1 \leq j \leq n} \{dirt(r_j) = i_j\} \wedge \\ & \qquad robot\_in = robotdocking \wedge human\_in = r_1 \end{aligned}$$

Where n is the number of rooms generated and each i_j is set to 1 with probability 0.3, 0 otherwise.

As said before, domain knowledge can be injected into the planner by means of formulas expressed in temporal logic. An example of the formulas used follows:

```
forall r: always (not [[(robot-in=r) and
             (dirt(r)=d) and (d>0)] and
             [next ((dirt(r)=d) and
                   (not (human-in=r)))]])
```

The operator $next$ specifies the check to be performed in the next belief situation relatively to the one in which the formula is first evaluated. This rule defines that it is unacceptable a transition from belief situation b to belief situation b' such that: in b the robot is in a room with a dirt level $d > 0$ and in b' the human is still not in the room and the level of dirt is unchanged. Thanks to this progression formula the robot is forced to clean whenever it is possible.

We analyzed the performance of our planner in three different test setups.

First setup In the first setup, we fixed the number of rooms in the apartment to 3, plus a special location, the robot-docking, where the robot can recharge its batteries and is virtually unreachable by the human. For this setup, we automatically generated 81 problems, grouped in 9 situations, varying the number of human agendas (1, 3, 5) and the number of human actions that compose each of the agendas (1, 3, 5). The problems generated for this experimental setup are fully solvable, that is, a complete search would lead to a robot policy where all the goals have been achieved.

The human actions, also generated automatically, can be of two types: the user can move from one room to another (in such case the duration of the event is fixed to 1 minute) or stay in the current location (the duration of such events is set to a random time value, spanning from 10 to 120 minutes, when the the problem is generated). Moreover, each event can be marked at problem generation time as observable by the system with probability 0.3 and can produce effects in the environment (that is, it can add new dirt on the floor and therefore a new goal to the robot) with probability 0.2. The actions of the human are specified like those of the robot but they do not have preconditions, as we do not need to test their applicability in the planning process. An example of a human action follows:

```
name: a1
results: human-in := r1
  and obs (human-in = r1)
  and dirt(r1):=(dirt(r1)+1)
time: 60
```

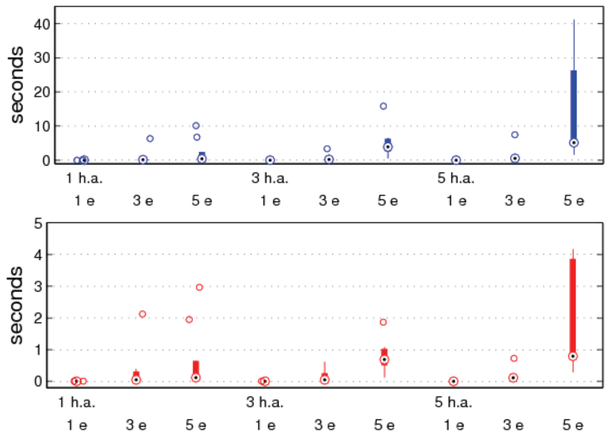


Figure 3: The vacuum cleaner scenario, first setup. Planning time when brute force is employed (top) and when domain knowledge is injected into the planner (bottom). The box plots detail the CPU user time (in seconds) for each set of runs on a PC with an Intel(R) Pentium(R) D 3 GHz processor and 2 GB of RAM. The problems are grouped by the number of human agendas provided as input to the planner (1,3,5) and the number of events composing each of the agendas (1,3,5). The central mark of each boxplot is the median of the represented values, while its edges are the 25th and 75th percentiles. Whiskers contain all the remaining data points not considered outliers, that are plotted individually.

Each problem is solved first exploring the full search space, with exhaustive breadth-first search, limited by the time duration of the human agendas, and then injecting domain knowledge into the planner. Figure 3 shows the results in terms of execution time for this setup. As expected, the run time increases with both the number of human agendas employed and the number of events that compose each agenda. The use of domain knowledge proved to cut by almost 10 times the execution time and the number of nodes explored in the search. The generated policies solved the corresponding problems with the same success degree than the ones generated without using domain knowledge, and had the same costs.

It is worth mentioning that our actual scenarios, in which we use the planner in a real environment as the core of our human aware framework, typically have the same complexity of the test runs of this setup. In our experimental setup, we observed that the plan recognition module typically provides the planner with 2 or 3 human agendas, each composed by a number of events that rarely exceeds 5. Therefore, our real problems are typically solved in a matter of a few seconds, which is more than acceptable considering that the time granularity we employ in this domain is one minute.

Second setup The problems analyzed in this second setup are generated in the same way detailed above. The difference here is the number of rooms — 5 instead of 3 — and therefore the number of applicable actions at each step. Also in this case, we tested the planner on 81 automatically generated, fully solvable problems, grouped in 9 situations.

Figure 4 shows the outcome of this setup in terms of CPU user time. As in the previous setup, domain knowledge speeds up the computation, reducing the required time to less than 20 seconds even for the most difficult problems. The use of domain knowledge did not affect the success rate degree. However, in 5 out of 81 problems a slight increase in the cost has been observed. In the worse case, one of the problems with 5 human agendas each composed by 3 human events, the cost of the plan increased by 4.58 % compared to the one generated by brute force.

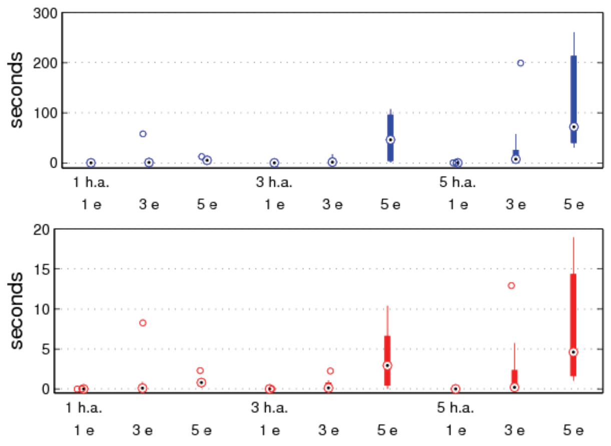


Figure 4: The vacuum cleaner scenario, second setup. Planning time using brute force (top) and using domain knowledge (bottom).

Third setup The third and final setup is meant to extensively test the performance of the planner also under circumstances that would not normally arise in a real environment. In this setup, we generated 450 problems, each with a potential success degree of at least 0.8 (that is, $\sum_{g \in G} V(g) \geq 0.8$) and for these problems we aimed to find the best possible robot policy. The actions and the events are automatically generated as in the previous setups, but now the number of agendas used is increased to (5, 7, 9) and the number of events per agenda to (3, 5, 7). 50 problems have been generated for each of the 9 situations.

As we can see from figure 5, also in this case the progression formulas significantly prune the search space (bottom). However, a little loss in terms of success degree can be observed: the solutions of 5.11 % of the problems using domain knowledge reported a loss in success degree of at most 6 %.

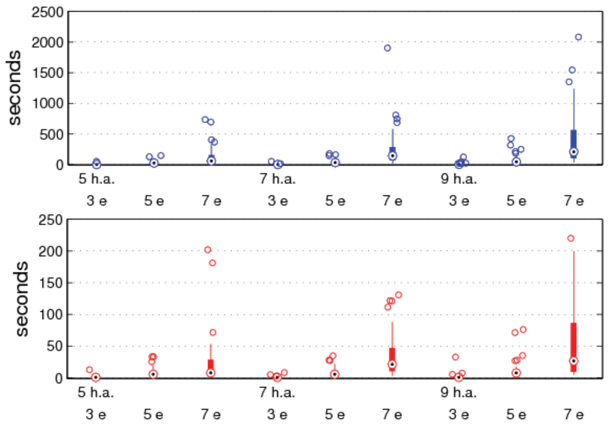


Figure 5: The vacuum cleaner scenario, third setup. Planning time using brute force (top) and using domain knowledge (bottom).

The moving table scenario

In the second scenario, the interaction constraints IC of the planner are designed to provide a service to the user, instead of avoiding interference with him. In this case, we used the planner to generate policies for a moving table that can autonomously bring drinks to the user. The IC of the table is to be ready with a drink within ten minutes from the arise of a request by specific actions of the user:

```
always ((not(human-wants-drink=t)) or
        (time-elapsed-since-request<=10))
```

The goal G of the table is to be back at its docking station when the execution of the policy is over. In this case, the set RA is composed by 5 parametric actions, that are specified as in the previous example. The agent can move from room to room, stay or sleep in a room (the sleep action lasts longer and has a lower cost), get a drink from the fridge and deliver it in the room where the human is located.

We tested this scenario in a single setup, in which the robot must satisfy the user's requests in an apartment composed by 3 rooms plus the special robot-docking location. As in the previous scenario, we automatically generated the initial belief situations, while keeping the initial positions of both robot and human fixed. We generated 300 problems, each with a potential success degree of at least 0.8, grouped into 6 situations according to the number of human agendas (5,7) and the number of human events per agenda (3,5,7).

The human events that compose the agendas have the same structure of the ones detailed in the vacuum cleaner scenario. Each of them, at problem generation time, has a 0.3 probability of being observable by the system, and a 0.5 probability of arising the request for a drink. The duration of each event is also generated as in the previous scenario.

The 300 problems were first solved by brute force, and then by providing domain knowledge to the planner (one of the formulas employed, for instance, specifies that the robot should move only to the robot-docking or in the room where the human is located, in case he requires a drink).

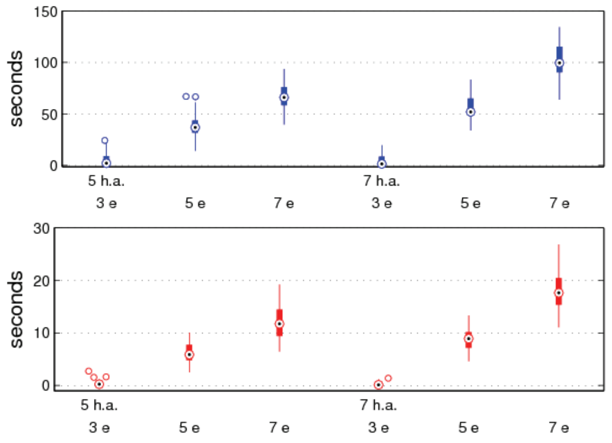


Figure 6: The moving table scenario. Planning time using brute force (top) and using domain knowledge (bottom).

As it can be observed from figure 6, also in this case the use of domain knowledge sped up the computation. The increased performance came at the price of a drop in the success degree in the solution of 7.3 % of the problems of at most 20 %.

A Full System Test Case

We now describe how our planner can be incorporated into a full system for human-aware task planning. The system we describe has been implemented and deployed in a real robotic home environment (Saffiotti et al. 2008), including a variety of sensors and a mobile robot, where a single person is acting (see figure 7). The system is composed of three main parts: a human plan recognition module, the planner, and an execution and monitoring module. A single illustrative run is presented below, since extensive testing of the complete system is not the concern of this paper.

Plan recognition module

The task of the plan recognition module is to produce an estimate of the possible agendas being executed by the human, by taking as input sensor readings and a pre-defined set

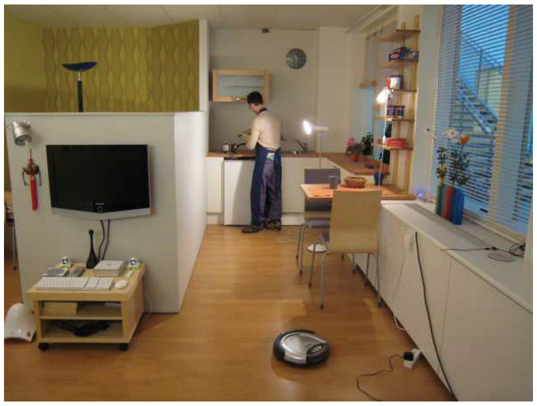


Figure 7: The home environment used in our experiment.

of potential agendas. From the perspective of human-aware planning, the most important aspects of this estimate are the forecast of the future actions that the human will perform and the detection of completion of recognized activities.

The used plan recognition module has a multi-layer structure, and it refines and abstracts the data received as input from the sensors step after step. At the base layer, sensor data are collected and coupled. Using a rule based system, simple instantaneous actions are detected (e.g.: the presence of the user in the kitchen and the fact that the fridge door is open let us infer that the human is using the fridge). In the next layers, we use Hidden Markov Models to identify more complex activities and finally to recognize human plans among a set of pre-defined ones. Similar approaches have already been successfully used (Aoki et al. 2005), although in our case the HMMs are defined beforehand and not learned from training data.

It should be emphasized that this component is not intended to be state-of-the-art, since the research issues related to plan recognition are not the focus of our work. In fact, this module could be easily replaced by another one in future experiments. The module that we used, however, proved to be sufficiently efficient and robust to noise in the sensor data for the purposes of our experiments.

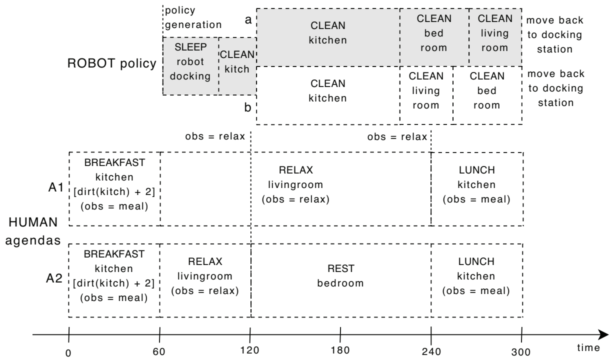
Execution and monitoring module

The executor receives from the planner the sequence of actions that the robot should perform and it sends them, one by one and with the appropriate timing, to the robot itself.

The monitoring module provides continuous support to the execution of the plan. If the plan recognition module updates the human plans identified, then a replanning signal is raised: the execution is suspended, the status of the robot and the effects on the environment of the actions of the robot so far executed are passed to the planner. The planner can thus calculate a new plan, that will be consistent with both the environment status and the new human plans.

A test run

We have tested the system in our home environment where one user is executing daily activities in the morning, from 8 am to 1 pm, following one of six pre-defined agendas, as discussed in the previous section. During the same time span, a robotic vacuum cleaner has the task to clean the floor in all the rooms that need it, minimizing the interference with the user. In our case, this means that the robotic vacuum cleaner must both operate and wait in rooms that the system has predicted as not occupied by the human at that time. In our test run, all rooms were marked as dirty from the beginning, so the robot must clean all of them.



policy generation
a
b
ROBOT policy
SLEEP robot docking
CLEAN kitch
CLEAN kitchen
CLEAN bed room
CLEAN living room
move back to docking station
CLEAN kitchen
CLEAN living room
CLEAN bed room
move back to docking station
obs = relax
obs = relax
HUMAN agendas
A1
BREAKFAST kitchen [dirt(kitch) + 2] (obs = meal)
RELAX livingroom (obs = relax)
LUNCH kitchen (obs = meal)
A2
BREAKFAST kitchen [dirt(kitch) + 2] (obs = meal)
RELAX livingroom (obs = relax)
REST bedroom
LUNCH kitchen (obs = meal)
0
60
120
180
240
300
time

Figure 8: The robot policy generated by the planner (top) with the two agendas (bottom) identified as possible by the plan recognition module. The observation of the completion of the human activity `RELAX` allows the robot to decide which branch must be executed.

The agenda executed by the user in our test run was the one marked **A1** in figure 8. Another agenda (**A2**) among the pre-defined ones is very similar to this one. After a delay, the plan recognition module identifies these two agendas as possible given the received observations, and passes both of them to the planner as suitable candidates for the subsequent actions of the human.

In general, when more than one agenda have been identified as possible, the policy to be executed by the robot will contain branches and the observation of the last completed human activity should be used to discriminate which action to perform next. In our case, the output policy contains one branch: the robot will follow the policy marked **b** in case the system observes that the human has completed the `RELAX` activity at the time of the branching. Otherwise, if there is no such observation, the robot will follow the **a** branch.

In the actual run, the robot executed successfully every action marked in a darker color, and it used the observation variable `RELAX` to discern the correct course of action.

Conclusions and Future Works

We have presented a human aware planner, designed to be part of a larger framework deployed in a real environment. The main contribution of our work is represented by the new techniques we designed to take into account forecasted human actions at planning time. Such actions do not only impose constraints on the robot, e.g., never schedule the cleaning of a room while the human is there. They can also be the source of new goals, e.g., the kitchen must be cleaned after the human has used it for cooking. The possibility of observing the results of some human actions can then be employed by the planner to identify the best policy to achieve the identified goals.

In our future work, we intend to perform extensive tests of the full system in a real environment, possibly deploying different robots with different tasks. Another priority is to relax the constraint that limits us to actions of fixed duration, both for the robot and the human. In this context, it is interesting to note that Mausam and Weld (Mausam and Weld 2008) showed that by first planning with only the expected duration and then improve or re-plan the policy with other possible/actual durations, one can still obtain policies that are quite close to the optimum.

Acknowledgments

This work has been partially supported by CUGS (Swedish national computer science graduate school) and by the Swedish KK foundation.

References

Akin Sisbot, E.; Clodic, A.; Marin, L.; Fontmarty, M.; Brèthes, L.; and Alami, R. 2006. Implementing a human-aware robot system. In *Proc. of the ROMAN Symposium*, 727–732.

Alami, R.; Clodic, A.; Montreuil, V.; Sisbot, E.; and Chatila, R. 2006. Toward human-aware robot task planning. In *AAAI Spring Symp 'To boldly go where no human-robot team has gone before'*, 39–46.

Aoki, S.; Iwai, Y.; Onishi, M.; Kojima, A.; and Fukunaga, K. 2005. Learning and recognizing behavioral patterns using position and posture of human body and its application to detection of irregular states. *Syst. Comput. Japan* 36(13):45–56.

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence* 116(1):123–191.

Blythe, J. 1994. Planning with external events. In *UAI '94: Proc. of the Tenth Annual Conference on Uncertainty in Artificial Intelligence*, 94–101.

Bouguerra, A., and Karlsson, L. 2005. Symbolic probabilistic-conditional plans execution by a mobile robot. In *IJCAI-05 Workshop: Reasoning with Uncertainty in Robotics (RUR-05)*.

Broz, F.; Nourbakhsh, I.; and Simmons, R. 2008. Planning for Human-Robot Interaction Using Time-State Aggregated POMDPs. In *Proc. of the Twenty-Third Conf. on Artificial Intelligence (AAAI)*.

Galindo, C.; Fernández-Madrigal, J.; and González, J. 2008. Multi-hierarchical interactive task planning. application to mobile robotics. *IEEE Transactions on Systems, Man, and Cybernetics part B* 18(3):785–789.

Graf, B.; Hans, M.; and Schraft, R. 2004. Mobile robot assistants: issues for dependable operation in direct cooperation with humans. IEEE *Robotics and Automation Magazine* 11(2):67–77.

Gravot, F., and Alami, R. 2001. An extension of the plan-merging paradigm for multi-robot coordination. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2929–2934.

Grosz, B., and Kraus, S. 1996. Collaborative plans for complex group action. *Artificial Intelligence* 86(2):269–357.

Haslum, P., and Geffner, H. 2001. Heuristic planning with time and resources. In *Proc. 6th European Conference on Planning*.

Hoffman, G., and Breazeal, C. 2007a. Cost-Based Anticipatory Action Selection for Human–Robot Fluency. *IEEE Transactions on Robotics* 23(5):952–961.

Hoffman, G., and Breazeal, C. 2007b. Effects of anticipatory action on human-robot teamwork – efficiency, fluency, and perception of team. In *HRI '07: Proc. of the ACM/IEEE International Conference on Human-Robot Interaction*, 1–8.

Jenkins, O.; González, G.; and Loper, M. 2007. Tracking human motion and actions for interactive robots. In *Proc. of the ACM/IEEE International Conference on Human-Robot Interaction*, 365–372.

Kaebling, L.; Littman, M.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1–2):99–134.

Karlsson, L. 2001. Conditional progressive planning under uncertainty. In *Proc. of the 17th Int. Joint Conference on Artificial Intelligence (IJCAI)*, 431–438.

Mausam, and Weld, D. 2008. Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research* 31:33–28.

Montreuil, V.; Clodic, A.; and Alami, R. 2007. Planning human centered robot activities. In *Proc. of the 2007 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*.

Nau, D.; M., G.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Puterman, M. L. 1994. *Markov Decision Processes: discrete stochastic dynamic programming*. John Wiley & sons.

Saffiotti, A.; Broxvall, M.; Gritti, M.; LeBlanc, K.; Lundh, R.; Rashid, J.; Seo, B.; and Cho, Y. 2008. The PEIS-ecology project: vision and results. In *Proc of the IEEE/RSJ Int Conf on Intelligent Robots and Systems (IROS)*, 2329–2335. Online at http://www.aass.oru.se/~asaffio/.

Sisbot, E.; Marin, L.; and Alami, R. 2007. Spatial reasoning for human robot interaction. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2007*, 2281–2287.

Tapus, A.; Mataric, M.; and Scassellati, B. 2007. The grand challenges in socially assistive robotics. IEEE *Robotics & Automation Magazine* 14(1):35–42.