Interactive task planning under uncertainty and goal changes

Sugato Bagchi \textsuperscript{a,}\textsuperscript{*}, Gautam Biswas \textsuperscript{b,}\textsuperscript{1}, Kazuhiko Kawamura \textsuperscript{b,}\textsuperscript{1}

\textsuperscript{a} IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA
\textsuperscript{b} Intelligent Robotics Laboratory, Vanderbilt University, Box 1804 Sta B Nashville, TN 37235, USA

Abstract

This paper describes a task planner designed for service robots. Task planning for service robots differs from traditional planning in the following ways: (i) it requires more extensive interactions between the robot and the human. This interaction may convey changed objectives and goals, and advice or directions to perform tasks in a preferred way, (ii) the presence of multiple agents, such as other robots and humans in the work environment. This makes the environment dynamic, since these agents could make changes even as the robot is executing a predetermined plan, and (iii) lack of structure in the environment (in contrast to most industrial applications). This makes the task of sensing relevant conditions less reliable and less feasible, so the robot often has to plan with incomplete knowledge of the environment. This requires the task planner for service robots to be interactive, reactive, and possess the ability to dynamically evaluate plans based on their reliability in uncertain conditions.

This paper describes a planner which uses a spreading activation mechanism to combine two approaches to planning: (i) a proactive component that generates plans biased toward picking the most reliable actions in a given situation, and (ii) a reactive component that alters action selection based on changes that occur in the dynamic and uncertain environment. A number of examples related to the ISAC system for feeding the handicapped demonstrate the effectiveness of the planning mechanism.

Keywords: Planning under uncertainty; Decision theoretic planning; Service robots

1. Introduction

As robotics and automation are being applied to new areas such as those in the service sector [16], the need for interactive and collaborative decision making is becoming important. Consider for example, the task of feeding a disabled user with a robotic arm. The emphasis here is not on the robot’s abilities to plan and act autonomously, but to do so interactively, keeping the user’s comfort and safety as one of its primary goals, and incorporating the user’s advice into its action selection mechanism. To achieve this extensive interaction before and during task execution, it is crucial that the robot comprehend the user’s goals at different levels of detail, and if necessary, refine and modify these goals dynamically as the situation demands. The planner also needs the ability to react to unexpected changes in the environment. Unexpected changes may be attributed to other agents (human or otherwise) that operate in the same workspace, and the fact that the robot operates in a natural as opposed to a structured environment.\textsuperscript{2} As a result, the robotic

\textsuperscript{*} Corresponding author. E-mail: bagchi@watson.ibm.com.
\textsuperscript{1} E-mail: [biswas, kawamura]@vuse.vanderbilt.edu.

\textsuperscript{2} An example of a structured environment is an industrial setting, where lighting and position of objects can be carefully
agent often has incomplete knowledge and information about the domain and situation it operates in. Task planning for robotic applications has been typically performed non-interactively. Once the current state of the environment and the desired goals are supplied, Artificial Intelligence (AI) planning algorithms search for a sequence of actions without any further interaction[17]. “Anytime” planning algorithms [5,7] can be interrupted from their processing to output a plan at any time, but they cannot be interrupted in order to provide additional input. Such additional input may be necessary if the state of the environment changes unexpectedly or the desired goals are modified. Reactive systems [4,7,9] do react to unexpected changes in the environment, but not to goal changes.

In this paper, we describe a task planner that can be made to alter its plans during the planning and execution process. Planning takes place as a spreading activation process that uses information from:

- the current state of the environment to determine what actions are more likely to succeed in achieving their goals, and
- the desired goals, so that the robot can prioritize its tasks and actions to ensure that it addresses goals based on importance values provided by the user.

As discussed, the current state of the environment may change suddenly, e.g., a second agent may remove a fork from its current location, or the user may change his goals, e.g., a person being fed spaghetti, may want a drink of water and specify that to the robot. Lack of information, because of the actions of other agents, insufficient and unreliable sensors, and incomplete domain knowledge, is modeled probabilistically. Our framework for probabilistic representation of uncertainty is described in Section 2. When changes occur either in the state of the environment or in the desired goals, it is immediately conveyed to the planning process, which propagates this information using spreading activation to determine the best course of actions to follow given this change. Section 3 describes the network representation used by the spreading activation planning process. The plan selection method is described in Section 4. The emphasis of this paper is on illustrative examples taken from the domain of ISAC, a robotic aid system for feeding the disabled (see Fig. 3). A set of feeding tasks under different scenarios is illustrated in Section 5.

2. Probabilistic representation of uncertainty in planning

Why is uncertainty present in an environment? Consider the action of a robot agent using a fork to pick up french fries from a plate. To be completely certain about the success of the action, the robot needs sensors to detect all the relevant conditions, which includes: (i) whether a plate is present, and if so, its position; (ii) whether fries are on the plate and how many there are; (iii) the position and orientation of the individual pieces so that the robot may pick them up correctly; (iv) the hardness of the fries so as to apply the proper force; (v) the sharpness of the fork; and so on. Clearly, it is not possible to sense all the details of the environment. Furthermore, even with all this information, the robot will need to use rather complex theories of the physical world to determine how the different parameters above relate to each other (e.g., the relation between the sharpness of the fork, the hardness of the fries, and the amount of force that needs to be applied to pick up one or more fries). It may not possess such knowledge in an explicit form.

Even if all relevant conditions can be sensed and all required physical theories are available to the robot agent, making complete determinations may be expensive both in terms of the cost for sensing, and the computation time required to derive the required information from sensed conditions. Therefore, dealing with information in such detail is impractical. Details, such as (iii)–(v) in the above example, though relevant to the successful execution of the action are likely to remain unknown. Performing the action without this information results in the uncertainty about its success.\(^3\)

If a condition which is crucial for the success of an action is not sensed, either because the agent does not know about its significance or because the condition is impractical to sense, the action will succeed only

---

\(^3\)These complexities are present even if a static environment is assumed. In reality, environments are dynamic; other agents, whether artificial or natural, may actively alter the conditions in the environment, adding to the uncertainty.
when the condition happens to hold. The probability of success will depend on the probability with which this unsensed condition is valid in the environment.

Sometimes, it is possible to infer the state of one condition from another known condition through a probabilistic relation between the two. For example, the observation that the plate is full of fries allows inferring with high probability that the robot will position the fork on the part of the plate where there are fries and that the fries will be in the right orientation for a successful pick-up. Since this relation is not completely certain, we define plate-full as a soft constraint for the pickup-fries action. It is desirable for this condition to be true to execute this action with success. However, it is not essential: the action may still succeed when the condition is not true.

Even for conditions that can be sensed, there is the possibility of sensor noise. For example, suppose a camera is used to detect a position of the fork on the table. Errors in the position data can occur due to improper segmentation of the fork (perhaps due to bad lighting, shadows, and overlapping objects) and inaccurate transformation from camera coordinates to robot coordinates (perhaps due to a slight movement of the camera, resulting in improper calibration).

3. Representation of actions for planning

A probabilistic framework is adopted for modeling uncertainty in the environment that the robot operates in [1,3]. The domain is represented as a set of propositions \( C = \{c_1, c_2, \ldots, c_N\} \). Each proposition \( c_i \in C \) that depicts a situation of interest (e.g., fork located, plate is full, and spoon in hand) is Boolean, i.e., \( c_i = \text{true} \) or \( c_i = \text{false} \). Propositions also have associated probabilities, indicating the degree of belief that they are true. The robot agent has a set of actions \( \{a_1, a_2, \ldots, a_M\} \) that it can perform. The reliability of achieving the effects of an action is represented as a probability conditioned on the execution of the action.

The representation for actions has been derived from the STRIPS formalism, adapted to include probabilistic information about the domain. Fig. 1 shows the graphical representation of an action and its associated propositions as nodes. Proposition nodes with links pointing to an action node are its set of conjunctive preconditions. The STRIPS notion that the preconditions set of an action must evaluate to true, before an action can be executed is generalized to include preconditions that increase the probability of an action succeeding in achieving its effects. This is denoted by the strength of a link \( w_{ij} \), which represents the correlation between a proposition node \( c_i \) and action node \( a_j \). Proposition nodes with links pointing to them denote the effects of the actions where the links originate. The strength of such a link, \( w_{jk} \) represents the probability with which the action \( a_j \) achieves the effect \( c_k \).

For the link between an action \( a_j \) and one of its preconditions \( c_i \), the correlation is defined as

\[
    w_{ij} = \frac{P(a_j^s \mid c_i = \text{true}) - P(a_j^s \mid c_i = \text{false})}{P(a_j^s \mid c_i = \text{true}) + P(a_j^s \mid c_i = \text{false})}.
\]

where \( a_j^s \) denotes successful execution of \( a_j \). This correlation is \( 1 \) if the precondition is required to be true (false) for the action to execute. These represent the hard constraints for the action that must be satisfied. Intermediate values represent soft constraints, which are preconditions that increase or decrease the probability of an action’s success. If \( w_{ij} \) is positive, the probability of \( a_j \)'s success is higher if \( c_i \) is true. If \( w_{ij} \) is negative, the probability of success increases if \( c_i \) is false. Soft constraints, as discussed earlier, arise due to the probabilistic relation between a sensed proposition \( (c_i) \) and an essential but unobservable precondition of the action. Moreover, soft constraints can also be used to model sensor noise, where the noisy output of the
sensor is probabilistically related to a corresponding real-world proposition.

The link strength $w_{jk}$, between an action $a_j$ and one of its effect propositions $c_k$ is defined as follows

$$w_{jk} = \begin{cases} P(c_k = \text{true} \mid a^k_j) & \text{if } a_j \rightarrow (c_k = \text{true}), \\ -P(c_k = \text{false} \mid a^k_j) & \text{if } a_j \rightarrow (c_k = \text{false}). \end{cases}$$

(2)

where the event $a^k_j$ denotes the execution of $a_j$. These are the prior probabilities of the action $a_j$’s success, unconditioned on the state of the environment. A negative strength denotes that the expected value of the proposition after the action executes is false.

This representation forms the basis for a causal network that includes a set of actions, and their preconditions and effects. The causal network represents the domain knowledge that the agent uses for plan generation and execution. Plans, an ordered sequence of actions, are represented as paths in this network. Actions earlier in the sequence set up preconditions that enable later actions to execute, and this ultimately leads to the achievement of the goal propositions. We have implemented a spreading activation mechanism for determining such paths between the propositions observed in the current state of the environment and the desired propositions in the goal state. This mechanism is described in Section 4.

4. Action selection by spreading activation

Once the domain knowledge network of action and proposition nodes is formed as described above, a spreading activation mechanism selects actions for execution. Although planning is performed before an action is selected, the mechanism does not return a complete plan like classical planners do. As Fig. 2 describes, the spreading activation mechanism operates like a feedback control system, choosing and executing actions only when there is a difference between the current and goal states. This approach addresses the dual objectives of goal achievement and goal maintenance.

Let the current state of the environment be represented by a set of Boolean propositions $S_t$. A goal $G$ is defined as a subset of propositions that must achieve a desired value (true or false). The objective of the system is to reduce the error between $S_t$ and any state $S_t \supseteq G$. This is done by propagating information forward from the current conditions and backward from the goal propositions. In this paper, we describe and justify the propagation process intuitively. A more detailed mathematical description of the spreading activation process is given in [1–3].

Goal propositions are assigned numerical utilities whose magnitude denote how desirable it is for the proposition to achieve the desired value. The sign of the utility denotes whether the desired value of the proposition is true (positive) or false (negative). These utilities are propagated backward to actions which can change the value of the proposition. If any preconditions of these actions are not at their desired values, these propositions accumulate utility, making them possible sub-goals. Back propagation can be performed iteratively from effects to preconditions of actions, thus creating a sequence of sub-goals.

Forward propagation is used to predict possible future states of the environment given its current state. The probability that a proposition will assume a certain value depends on the probabilities of success of all the actions whose executions set the proposition to that value. This in turn affects the probabilities of success of all actions that have this proposition as a precondition. Forward propagation can be performed iteratively from preconditions to effects through a sequence of hypothetical action executions.

When the backward propagation of utility from a desired goal proposition meets the forward propagation of proposition probabilities from the initial state of the environment, the sequence of associated actions

---

4 The word “value” is not to be confused with “utility”. Throughout this paper, “value” of a proposition means its status (true or false). “Utility” denotes the desirability for a proposition to assume a certain value.

5 Opposite to the direction of the causal links.

6 Along the direction of the causal links.
establish a sub-plan for achieving that goal. When alternate plans are found for the same goal proposition, the one with the highest probability of success is chosen. This is done by calculating for each action node, a measure of its expected utility, defined in decision theory [15] as the product of the probability of the action’s success in achieving its effects (obtained from forward propagation) and the utility of its effects (obtained from backward propagation). Actions can also be assigned costs $C(a_j)$, representing the amount of resources (e.g., time) it consumes. If costs are assigned, the expected utility is adjusted by subtracting the cost. Mathematically, the expected utility of an action $a_j$ under the state $S_i$ and goal set $G$ is represented as

$$U(a_j | S_i, G) = P(a_j | S_i) \sum_{c_k \in C} w_{jk} U(c_k | G) - C(a_j).$$

(3)

This value is established by forward and backward propagations through the plan net. The expected utility may change with every step of the forward and backward propagation. After the first propagation step, the action receives utility only if its immediate effects are in the goal set $G$. The probability of its success is based on the current state of its preconditions. As the number of steps in the forward and backward propagations increase, the action will receive utilities from more and more distant goals. Also, the probability of the action’s success will be conditioned upon the execution of other actions.

The action selection mechanism simply selects the action with the highest expected utility. This selection may be done at any time, but the longer one waits (i.e., the more forward and backward propagations one allows) the more informed is the selection function based on the utility measure. Thus, the mechanism has the “anytime” property [5,7]. Increasing the allotted planning time allows a larger number of propagation steps before selecting the action with the highest expected utility. This makes the planner’s behavior change from reactive to deliberative.

Since utility originates in the goal propositions, an action will receive utility only if it is in a “path” of actions and propositions that lead to one or more goal propositions. The utility received by an action represents the product of the probabilities of success of the subsequent actions in the path leading to the goal proposition. So selecting the action with the highest expected utility implicitly selects the plan (i.e., the sequence of actions along a path) with the highest expected utility.

5. Examples of planning behavior

This section presents experiments conducted to test the action selection mechanism described above. These experiments present an empirical confirmation of the properties and capabilities of the planner, such as:

1. planning with hard and soft constraints and uncertainty in domain knowledge represented probabilistically,
2. planning while considering action costs and goal utilities,
3. planning under unexpected changes in the environment, and
4. plan modifications due to changes in goal utilities.

Before describing the performance of the action selection mechanism under the various situations that could arise, we present our application environment, the ISAC system, a robotic aid for feeding the disabled. Performance testing was done using a planning test bed for simulating plan behaviors.

5.1. The ISAC environment

Consider the task of using a robot to pick up some food from a plate for feeding a disabled user. This is an actual task, handled by the ISAC (Intelligent Soft Arm Control) system developed at Vanderbilt University [10,11]. It uses a pneumatically actuated robot manipulator for grasping utensils, picking up food from a plate or bowl, and feeding. A camera is used for recognizing and locating common tableware, including spoons, forks, cups, bowls, and plates on the table. To ensure that the food is placed in front of the user’s mouth, a pair of cameras track the user’s face in three dimensions. An illustration of the ISAC environment appears in Fig. 3.

Uncertainty is introduced due to the unstructured nature of the environment:

1. The object recognition system does not recognize objects with absolute certainty. A probability is associated with each of the recognized objects.
(2) Inaccuracies in camera calibration can lead to grasping errors.
(3) Success in picking up food from a plate or bowl depends on the position of the food within the plate or bowl.
(4) The user is free to change his goals or bias the plan selection process by specifying sub-goals.
(5) Other people are free to modify the environment while the robot is executing a task.

5.2. The planning test bed

The planning test bed uses a description language for expressing all the domain knowledge used by the planner: (a) a list of actions that can be performed by the agent, and (b) relevant conditions in the environment that can be sensed by the agent. In addition the planner also needs the current state of the environment and the goals.

Since the environment is dynamic, proposition values and goal utilities are allowed to change at any time, some means for signaling these changes to the planner are required. This is done through a graphical user interface to the planning test bed. An example of this X window system interface is shown in Fig. 4.

The buttons and other widgets are created from the domain description given to the test bed. The row of buttons on the top represents the value of the propositions in the current state of the environment. The plot windows display the expected utilities of actions over time. A * signifies the execution of the action at that point in time. Goal propositions can be assigned numerical utilities, as shown at the bottom of the window.

To run an experiment, the current state of the environment and the desired goal utilities are set. Then, the Run button is selected. This starts the forward and backward propagations and plots an action's expected utilities until the step limit is reached. The action with the highest expected utility is fired (represented by a * in the action's utility plot). At this point, the change in the environment as a result of performing the action is indicated by highlighting the appropriate proposition buttons. The action selection process can then be continued by selecting the Run button again. The Step button allows one to single step through the propagations for debugging purposes. The Setup button can be used to vary three planner parameters: the link efficiency $\eta$, the sensitivity threshold, and the number of propagation steps between action selections.
The user (or another X application) can set a proposition to true or false even as the planner is running. Similarly, goal utilities too can be changed at any time. These features make it possible to test the reaction of the planner to unexpected changes in the environment and to changes in the goal priorities. Actions can be forced to execute at any time by clicking the corresponding label. This allows the system to learn or modify the supplied domain knowledge.

5.3. Probabilistic planning

When other things, such as action costs and utilities are equal, the planner must choose a plan that has the highest probability of success. Based on information gathered from previous experience, the following characteristics of the domain should influence the robot’s actions:

1. In past feeding activities, the food pickup action was successful 6 times out of 7 when a fork was used and the plate was full. When the plate was not full, the fork was successful in only 3 of 9 attempts.
2. When a spoon was used, the action was successful in only 1 of 5 times when the spoon was small (a teaspoon) and 9 of 13 attempts when the spoon was larger.
3. Grasping the fork was successful 18 times out of 20.
4. Grasping the spoon was successful 15 times out of 17.

The proposition plate-full is a soft constraint for the action use-fork which picks up the food from the plate. The essential precondition is whether there is enough food in proper orientation at the spot on the plate where the fork will be going. This situation cannot be sensed easily so a related condition: whether
the plate is full enough, is sensed. When this is true, it lends credence to the essential condition. The proposition small-spoon is a soft constraint for the action use-spoon. If true, it reduces the chances of success of the pick-up action.

What should the planning behavior be, given different soft constraint values? The planner uses the probabilities of success in picking up the food, as computed by spreading activation to make its decision. Consider the situation where plate-full is true and, therefore, using the fork to feed has a higher reliability than using the spoon. Fig. 5(a) shows the traces of the expected utility for the relevant actions when small-spoon is false. Action executions are denoted by *'s in the plots. After two iterations with the fork, the proposition plate-full becomes false. Due to the decrease in probability of success using the fork, it is released and the spoon is used instead.

Fig. 5(b) shows a different planning behavior when the small-spoon is true. This time, when the plate-full turns false after a couple of iterations, the planner continues to use the fork because the probability of success using a small spoon is even less.

Note that the planner did not have to replan in order to switch plans when the state of the environment changed. The changes in expected utilities as a result of the change in the environment caused an implicit switch in plans. This example also illustrates a situation where an error does not have to occur before plans are changed.

5.4. Considering action costs

So far, we assumed equal costs for all actions. Now consider the situation where in addition to the actions described so far, there is another action fill-plate. There are three plans available when plate-full becomes false while using the fork for feeding: (1) fill up the plate, then use the fork again, (2) release the fork and use the spoon, and (3) continue to use the fork. The choice from these plans depend not only on the relative probabilities of success but also on the cost of the actions. Consider the case where the cost of performing all the actions including fill-plate are the same. Fig. 6(a) shows the planning behavior when plate-full is initially true but subsequently false. Instead of grasping the spoon (as in Fig. 5(a)), the planner selects fill-plate and then goes back to using the fork. When the cost of fill-plate is made larger than the utility it receives, the planner chooses to select the

---

**Fig. 5. Action selection based on the value of soft constraints.**

**Fig. 6. Action selection under different costs of the action fill-plate.**
spoon rather than filling up the plate. This is shown in Fig. 6(b). For both these examples, the state of small-spoon is false.

5.5. Handling unexpected situations

The ability of the planner to react to unexpected situations have been addressed to some extent in the previous sections. Here we focus on this ability, especially in situations where (i) an action fails to have its expected effect, (ii) an unexpected change in the environment causes a switch in plans, and (iii) goal utilities are suddenly changed by the user.

Since the spreading activation mechanism is immediately affected by changes in either the current or desired conditions, the planner is effective in handling these unexpected situations. Consider the unexpected situation that arises when the action does not achieve its desired effect. The planner now has the responsibility of choosing the next action. This action could be the same one that failed, or a new one, representing an alternate plan. This choice does not require the need for an explicit, "conscious" error handling procedure. The sole criterion of choosing the action with the maximum expected utility is applied. Fig. 7(a) shows an example of action selection behavior. When the action use-fork was not successful in picking up food, it was retried because it still had the highest expected utility. However, after a few more failed tries, the probability of its success became low enough to make its expected utility lower than that of the action release-fork. This represents the first action of the alternate plan which uses the spoon to pick-up the food.

Fig. 7(b) shows the reaction to another unexpected change in the environment, this time for the better. Consider the situation where only the spoon is initially on the table. The planner has no alternative but to pick up the spoon. At this point, a fork is unexpectedly located. As described above, the probability of success with the fork is higher when the plate is full. This causes the planner to switch plans, releasing the spoon and using the fork instead.

Another source for an unexpected situation is changing in goals or their utilities. Here again, the planner reacts by switching plans if needed. In the example shown in Fig. 8(a), the original goal was to feed the user. During this process, the user asserted another goal of getting a drink with a higher utility (40, as compared to 20 for feeding). The planner

---

Fig. 7. Handling unexpected situations, such as (a) failure of an action, and (b) an unexpected change for the better.

Fig. 8. Reacting to an unexpected change in goal utilities.
reacts to this change by performing the actions necessary to satisfy the higher utility goal before going back to satisfying the lower utility goal of feeding. Again, the entire process was performed simply by picking the action with the highest expected utility.

In Fig. 8(b), the utility of the new goal is the same as the utility of the existing goal. This is not sufficient to cause a change in plans. The goal of bringing a drink has to wait until the goal of feeding the user has been achieved.

The ability to change the utility of goals during the action selection process is an important feature for mixed-initiative planning where multiple intelligent agents participate in the planning process [8]. When the planner chooses a course of actions that are at odds with some unspecifiable criterion, the user should be able to change the action selection behavior.

6. Where will it fail?

So far, the results of successfully applying the planner to various situations have been described. This section lists some domains and behaviors for which our action selection mechanism is inappropriate.

(1) Where actions do not have immediate effects. For example, the action of heating water for making tea, does not immediately make the water to boil. Currently, the action selection mechanism assumes that the effects of an action can be sensed right after its execution and lack of immediate effects may be treated as the failure of the action. To overcome these problems, an explicit temporal reasoning framework needs to be introduced.

(2) Where actions with numerical arguments are required. The action selection mechanism will have to be modified to operate in continuous-valued domains where actions require a numerical argument. For example, when hammering a nail, the amount of force applied should be proportional to the amount the nail needs to be driven into the block. Determining the right numerical value is traditionally the realm of control systems, but when goals become complicated, planning is required as well.

(3) Where sensing is expensive. The action selection mechanism assumes the existence of sensors that can quickly report the state of the environment. More often than not, sensing, especially image-based sensing, is a compute-intensive process. Action selection should include actions that sense.

7. Conclusions

This paper described a number of characteristics of planning systems that are important for human–robot symbiosis and the field of service robotics.

- The ability to plan, act, and react in uncertain environments. Note that uncertainty arises because of a lack of complete domain knowledge, the presence of other agents (human and robots) in the working area, inaccuracies and noise in the sensors, and the inability to sense all the required conditions in a particular situation.

- The “anytime” planning characteristic. This is achieved by the incremental nature of plan selection based on the spreading activation process. Any changes in the environment are recorded immediately. The user is also allowed to modify goal utilities at will, and the plan generation process reacts to these changes almost immediately.

Feeding of handicapped persons using the ISAC system was used as our test bed for empirically demonstrating a number of properties of our planning mechanism.

From a more theoretical viewpoint, this work shows how a combination of interactive and deliberative planning allows for the development of more robust and effective planners that can operate in realistic environments. The trade-off between the two is achieved by controlling the amount of time allocated for planning. Allowing a few propagation steps between action selections makes the planner fast but possibly sub-optimal. Increasing the number of propagation steps between action selections makes the planner more deliberative and optimal, but at the cost of increased computational complexity.

Classical planners, surveyed in [17], assume that the effects of an action are known with certainty. With this knowledge, they generate a set of actions that will achieve the desired goals. Since the reliability of actions are not modeled, the reliability of implementing the selected plan is not considered. Some planners do monitor for errors as the actions are executed [19]. An error is signaled if the observed state of the world
does not match the expected state after the execution of an action. However, unexpected situations are not always synonymous with execution errors. Some planners have been designed to consider the effects of actions to be probabilistic [6,18,12]. They compare the reliability of competing plans, but do not react to unexpected changes in the environment. Reactive planners [9,13,14] are designed to select and execute actions as a response to the current state of the world, thus interleaving planning and execution. This makes them more responsive to unexpected changes. However, with a few exceptions [5,7], they do not use probabilistic information to bias the selection of actions to those that were more reliable in the past. Therefore, while they may be good at handling the unexpected, they do not attempt to alleviate uncertainty by choosing more reliable actions.

The planning behavior demonstrated in this paper makes use of an existing model of the domain with three components: (i) a list of actions the agent can perform, (ii) a list of propositions the agent can sense or infer, and (iii) probabilistic relations between the actions and propositions. In this paper we have assumed that all three components of the domain knowledge are already available during planning. However, it may not be possible to have a priori information about the probabilistic relations between actions and propositions. In other work, we have developed methods by which the system can learn these relations from observations of action executions and by experimentation [1]. In future, we wish to look at more sophisticated learning techniques by which the robotic agent learns to interact more effectively with a human. Other dimensions that we will explore to improve human–machine cooperation include: (a) more robust and expressive interaction mechanisms between the human and robot (e.g., better language processing abilities), and (b) the ability to develop multi-agent systems that use the spreading activation methodology for plan generation and execution for cooperative problem solving of complex problems.

References