Planning for Human-Robot Teaming in Open Worlds

KARTIK TALAMADUPULA, J. BENTON, and SUBBARAO KAMBHAMPATI
Arizona State University
and
PAUL SCHERMERHORN and MATTHIAS SCHEUTZ
Indiana University

As the number of applications for human-robot teaming continue to rise, there is an increasing need for planning technologies that can guide robots in such teaming scenarios. In this article, we focus on adapting planning technology to Urban Search And Rescue (USAR) with a human-robot team. We start by showing that several aspects of state-of-the-art planning technology, including temporal planning, partial satisfaction planning, and replanning, can be gainfully adapted to this scenario. We then note that human-robot teaming also throws up an additional critical challenge, namely, enabling existing planners, which work under closed-world assumptions, to cope with the open worlds that are characteristic of teaming problems such as USAR. In response, we discuss the notion of conditional goals, and describe how we represent and handle a specific class of them called open world quantified goals. Finally, we describe how the planner, and its open world extensions, are integrated into a robot control architecture, and provide an empirical evaluation over USAR experimental runs to establish the effectiveness of the planning components.

14

1. INTRODUCTION

As the fields of robotics and Human-Robot Interaction (HRI) have advanced, demand has escalated for applications that require humans and robots to "team" and work together to solve complex problems. While some of these scenarios may be handled through "teleoperation", an increasing number require teaming between humans and *autonomous* robots. A compelling application of this type involves the urban search and rescue scenario, where a human is in remote contact with the robot and provides high-level instructions and goals. Clearly, robots operating in such teaming scenarios require the ability to plan (and revise) a course of action in response to human instructions. Our focus in this article is on understanding the challenges faced by the planner that guides a robot in such teaming scenarios.

Although there has been extensive work in the past on understanding the challenges of human-planner interactions (refer to mixed-initiative planning) and planner-robot interaction (refer to planning and execution), these efforts do not provide a complete solution in human-robot teaming scenarios (see Section 6).

Several parts of the state-of-the-art planning technology that go beyond typical *classical planning* are both required and easily adapted to human-robot teaming scenarios. In particular, the planner should allow for actions with durations to handle goals with deadlines, and partial satisfaction of goals should be possible to allow the planner to "skip" seemingly unreachable goals (e.g., if the goal of exiting a building cannot be currently satisfied, that should not prevent the robot from reporting on injured humans). For partial satisfaction planning, we model soft goals (i.e., goals that may remain unachieved) with a reward and give a cost to each action and the planner seeks to find a plan with maximum *net benefit* (i.e., summed goal reward - summed action cost). Along with these, an important part of any online system is execution monitoring and replanning to allow the planner to receive and react to new information from either the environment (e.g., the discovery of a new area) or from a human commander (e.g., a change in goal deadline). To accept information from a human commander, the robotic architecture parses and processes natural language (i.e., speech) into goals or new facts. If the architecture cannot handle a goal or fact by following a simple script located in its library, it calls the planner to find a method of achieving the goal.

Human-robot teaming tasks present an additional critical challenge not handled by current planning technology: *open worlds*. Most teaming tasks involve open worlds and require the ability to handle both counterfactual knowledge and conditional goals. For example, a human commander might instruct the robot to report on any injured humans that it encounters in a search-and-rescue scenario. In such a scenario, the world is open in that neither the human nor the robot know where injured humans are.

While the state-of-the-art planners are very efficient, they focus mostly on closed worlds. Specifically, they expect full knowledge of the initial state, and expect up-front specification of the goals. Adapting them to handle open worlds presents many thorny challenges. Three tempting yet ultimately flawed

approaches for making closed world planners handle open worlds involve either blindly assuming that the world is indeed closed, deliberately "closing" the world by acquiring all the missing knowledge before planning, or accounting for all contingencies during planning by developing conditional plans. The alternative (assuming a closed world) will not only necessitate frequent replanning during execution, but can also lead to highly suboptimal plans in the presence of conditional goal rewards. Acquiring full knowledge up front would involve the robot doing a sensing sweep to learn everything about its world before commencing the planning. This is clearly infeasible in open worlds where we do not even know how many objects may be there and thus do not know when to stop sensing. After all, a robot cannot be simply commanded to "sense everything," but rather has to be directed to perform specific sensing tasks. Accounting for missing knowledge would involve making conditional plans to handle every type of contingency, and letting the robot follow the branches of the plan that are consistent with the outcomes of its sensing. Such full contingency planning is already known to be impractical in propositional worlds with bounded indeterminacy (refer to Meuleau and Smith [2003]); it is so in open worlds where the number of objects (and their types) are unknown.

What is needed instead is both a framework for specifying conditional knowledge and rewards, and an approach for using it to direct the robot in such a way as to intelligently trade sensing costs and goal rewards. Accordingly, we propose an approach for representing and handling a class of conditional goals called *Open World Quantified Goals* (OWQGs). OWQGs provide a compact way of specifying conditional reward opportunities over an "open" set of objects. Using OWQGs, we can specify (for instance) that for a robot to report an injured human, it must have found an injured human and that finding an injured human involves sensing. We shall see how OWQGs foreground the trade-off between sensing cost and goal reward. We will discuss the issues involved in optimally selecting the conditional rewards to pursue, and describe the approximate "optimistic" method we used in the current work.

The rest of this article is devoted to describing the details of our planner and its open world extensions. We also discuss how the planner is integrated into the robotic architecture to support human-robot teaming in USAR, and present an empirical evaluation establishing the effectiveness of our approach. The article is organized as follows. We start by describing some details of our motivating USAR scenario. In Section 2, we present the automated planner that is used as a part of this system, including a description of the update syntax that enables new information in the world to be relayed to the planner. In Section 3, we discuss the challenges of planning in an open world, culminating in the definition of a new construct for conveying open world information and a description of its implementation in our system. Section 4 describes the architecture used to control the robotic agent and the integration of the planner and this architecture, including a description of how sensing and updates to the world state are handled. In Section 5, we present the results of an empirical evaluation conducted on the robot. Section 6 provides an overview of related work, and Section 7 presents our conclusions.
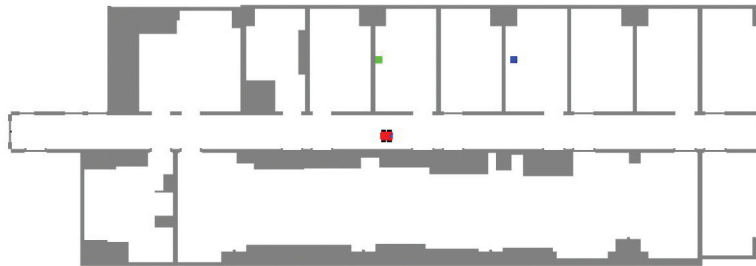
Fig. 1.   A map of a sample scenario; boxes in rooms are stand-ins for humans, where green (at left) indicates injured and blue (at right) indicates normal.

1.1 Overview of USAR Scenario

We consider the problem of a human-robot team engaged in an *Urban Search And Rescue* (USAR) scenario inside a building. The robot is placed at the beginning of a long hallway; a sample layout's map is presented in Figure 1. The human team member has intimate knowledge of the building's layout, but is removed from the scene and can only interact with the robot via on-board wireless audio communication.[1] The hallway in which the robot is located has doors leading off from either side into rooms, a fact known to the robot. However, unknown to the robot (and the human team member) is the possibility that these rooms may contain injured humans (victims). The robot is initially given a hard goal of reaching the end of the hallway by a given deadline based on wall-clock time. As the robot executes a plan to achieve that goal, the team is given the (additional) information regarding victims being in rooms. Also specified with this information is the quantified soft goal of reporting the location of victims.

In this example, the planner must reason about the cost-benefit trade-off (net benefit) of attempting to find a victim, since it is a soft goal and can be ignored if it is not worth the pursuit; it must then direct the robot to sense for the information that it needs in order to determine the presence of a victim in a particular location. The dynamic nature of the domain coupled with the partial observability of the world precludes complete, a priori specification of the domain, and forces the robot and its planner to handle incomplete and evolving domain models [Kambhampati 2007]. This fact, coupled with the inability of human experts to completely specify information relevant to the given problem and goals up front, makes it quite likely that information needed to achieve some goals may become available at some later stage *during* the planning process.

2. BASE PLANNER

The planner that we use, called *SapaReplan*, is an extension of the metric-temporal planner *Sapa* [Do and Kambhampati 2002] that handles partial satisfaction planning [Benton et al. 2009] and replanning [Cushing et al. 2008].

---

[1]Our overall scenario involves supporting natural language communication between the human and the robot; for details on the communications see Cantrell et al. [2010], Dzifcak et al. [2009].

Specifically, the planning problem is defined in terms of the initial state and the set of goals that need to be satisfied. Actions have known (real-valued) costs. Each goal can have a reward and a penalty $\in [0, \infty]$. The reward is accrued when the goal is satisfied in the final state, while the penalty is incurred for not satisfying it. The costs, rewards, and penalties are all assumed to be in the same units. The net benefit of a solution plan is defined as the sum of rewards of the goals it achieves, minus the sum of penalties of the goals it fails to achieve, and minus the sum of costs of the actions used in the plan. The use of reward/penalty model allows our planner to model both opportunities and commitments/constraints in a uniform fashion. A goal with zero penalty is a pure opportunity, while one with zero reward is a pure commitment. A "hard" goal has finite reward but infinite penalty (and thus *must be achieved* by any plan).

The planner consists of three coupled, but distinct parts:

—Search. *SapaReplan* performs a weighted A*, forward search using *net benefit* as the optimization criterion.

—Heuristic. The heuristic used to guide the planner's search is based on well-known relaxed planning graph heuristics where, during search, relaxed solutions are found in polynomial time per state. *Sapa* uses a temporal relaxed planning graph that accounts for the durations of actions when calculating costs and finding relaxed solutions. In the partial satisfaction planning extensions, the heuristic also performs online goal selection. In essence, it solves for all goals (hard and soft) in the relaxed problem and gives a cost for reaching each of them (∞ for unreachable goals). If the cost of reaching a soft goal is greater than its reward, it removes that goal from the heuristic calculation. If the cost of reaching a hard goal is infinity, it marks a state as a dead end. Finally, the difference between the total reward and total cost of the remaining goals is calculated and used as the heuristic value.

—Monitoring / Replanning. The extensions for replanning require the use of an execution monitor, which takes updates from the human-robot team architecture (in this case). Upon receiving an update, the planner updates its knowledge of the "current state" and replans. Replanning itself is posed as a new partial satisfaction planning problem, where the initial and goal states capture the status and commitments of the current plan [Cushing et al. 2008].

To see how our planning system copes with open environment scenarios, it is important to understand the details of its execution monitoring component. This is arguably the most important part of the planning system for the problem at hand, as its focus is on handling unexpected events and gathering new information for the planner. It serves as an interface between the human-robot team architecture (discussed in Section 4.1) and the planning engine.

Problem updates. New sensory information, goals, or facts given by a human commander can be sent to the planner at any time, either during planning or after a plan has been output. Regardless of the originating source, the monitor listens for updates from a single source from the architecture and

correspondingly modifies the planner's representation of the problem. Updates can include new objects, timed events (i.e., an addition or deletion of a fact at a particular time, or a change in a numeric value such as action cost), the addition or modification (on the deadline or reward) of a goal, and a time point to plan from. An example update is given next.

```
(:update
 :objects
    red3 - zone
 :events
    (at 125.0 (not (at red2)))
    (at red3)
    (visited red3)
 :goal (visited red4) [500] - hard
 :now 207.0)
```

All goals are on propositions from the set of boolean fluents in the problem, and there can only be one goal on any given proposition. In the default setting, goals are hard, lack deadlines, and have zero reward.[2] All fields in an update specification, with the exception of ":now" (representing the time we expect to begin executing the plan), may be repeated as many times as required, or left out altogether. The intent of allowing such a flexible representation for updates is to provide for accumulation of changes to the world in one place. In the particular example provided, a new object "red3" of type "zone" is declared. In addition, three new *events* are defined, one of them with a temporal annotation that describes the time at which that event became true. A new hard goal that carries 500 units of reward is also specified, and the update concludes with the specification of the current time.

As discussed by Cushing et al. [2008], allowing for updates to the planning problem provides the ability to look at unexpected events in the open world as new information rather than faults to be corrected. In our setup, problem updates cause the monitor process to restart the planner (if it is running) after updating its internal problem representation.

3. PLANNING IN THE OPEN WORLD

As previously discussed, there exists an obvious problem with using a planner that assumes a closed world within an open world environment. Because the world is open, the robot (as well as the human) do not have full knowledge of all the objects in the world. In the USAR scenario, neither the human nor the robot know where the injured humans might be. Furthermore, it is also possible that the human-robot team does not have a full "map" of the building in which the rescue is taking place.

One immediate ramification of the open world is that the goals are often conditioned on particular facts whose truth value may be unknown at the initial state. For example, the most critical goal in the USAR scenario, namely

---

[2]Since these goals are *hard*, they can be seen as carrying an infinite penalty, that is, failing to achieve even one such goal will result in plan failure.

reporting the locations of injured humans, is conditioned on finding injured humans in the first place.

To see this, consider our Urban Search And Rescue (USAR) scenario, where we have a set of objects that imply certain facts. For instance, when moving through the hallway we can say that sensing a door implies the existence of a room. Subsequently, doors imply the potential for goal achievement (i.e., opportunities for reward). Specifically, the robot's task is to find injured people in a building. While the number of injured individuals remains unknown, the commander becomes aware that people are likely within rooms (and subsequently passes this information on to the robot). This goal is over an open world, in that new objects and facts may be brought to light through either external sources like the mission commander or through action execution.

To be effective in such scenarios, the planner should be opportunistic, generating plans that *enable* goal achievement as against finding the most direct path to the currently known goals (e.g., entering rooms to look for injured individuals). This planning is interleaved with plan execution. Unfortunately, we have several other constraints that may preclude the achievement of goals. The robot must meet a hard deadline and may run out of exploration time; it may also be unable to fully explore the building due to parts of it being inaccessible. Additionally, sensing to resolve the truth of world facts and the existence of objects may often be costly and time consuming. This means that certain aspects of the world may remain open (and therefore unknown) by design.

3.1 Conditional Goals

To formally model the USAR robot's goal of looking for and reporting injured people, it is useful to consider the fact that this goal is certainly not one of simple achievement, since the robot does not need to (and should not) report victims unless they are actually present in the rooms. The uncertainty in this scenario (and other similar real-world problems) stems from the inherently conditional presence of objects (and the truth of facts about them) in the world. Such goals can be looked at as *conditional goals*.

For exposition purposes, we shall start with a discussion of challenges involved in handling *propositional* conditional goals. A propositional conditional goal $P \rightsquigarrow G$ is interpreted as "G needs to be satisfied if P is true initially". Formally, we have the following.

Conditional goal. Given ground predicates A and B, a *(hard) conditional goal* $A \rightsquigarrow B$ is defined as the requirement that if A is true in the initial state I, then any solution plan ρ must make B true in the final state resulting from the application of ρ to I.

From the previous definition of conditional goals, it holds that the set of goals that a plan ρ needs to fulfil in order to be considered a solution to the problem is variable, and that the composition of such a set depends on the state of the antecedents of the conditional goals initially (at I). It also follows that a plan ρ' will not be considered a solution unless it fulfils each and every one of the conditional goals $g_c \in \mathcal{G}_c$.
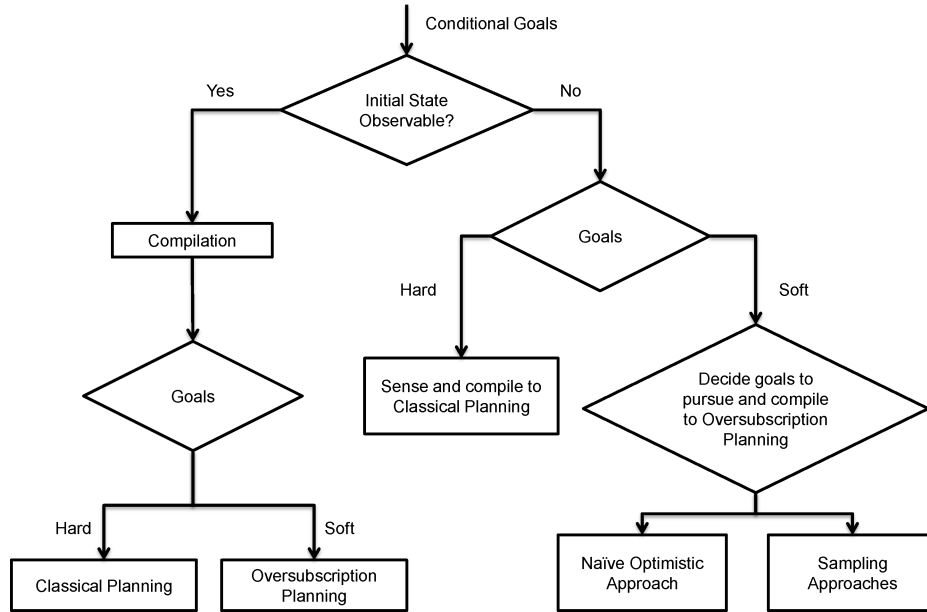
Fig. 2. A schematic outline of methods to deal with conditional goals.

The conditional goal as defined before poses a "hard" constraint: if the antecedent holds, then every solution plan *must* achieve the goal. It is useful to relax this requirement.

Soft conditional goal. A *soft conditional goal* $A \rightsquigarrow B\,[u][p]$ is defined as the provision that if A is true in the initial state I, then the achievement of B in the final state $G' \subseteq \mathcal{G}$ (the set of all goals) will accrue a reward of u units, while the failure to achieve B will incur a penalty of p units.

In general, it is useful consider a *spectrum* of planning methods (as shown in Figure 2) to deal with conditional goals, all of which are contingent on the the observability of the initial state $I \in \pi$. If I is fully observable, the planner knows the values of the antecedents of all the conditional goals $g_c \in \mathcal{G}_c$. With this information, a problem with conditional goals may be compiled into a standard classical planning problem (in case only hard conditional goals are present and a Partial Satisfaction Planning (PSP) problem otherwise).

However, if I is partially observable, the planner is faced with a more complex problem. If all the conditional goals are hard (and hence must be achieved for plan success), the planner has no option but to direct the robot to sense for all the facts that occur in the antecedents of the goals in $\mathcal{G}_c$, culminating in the compilation approach mentioned previously.

If the conditional goals in the scenario are all *soft* instead,[3] the planner is confronted with an interesting problem: it must not only sense in order to establish which of the antecedents are true in the initial state, but must also

---

[3]If there is a mixture of hard and soft conditional goals, they can be split and we can handle the hard conditional goals as described previously.

select a subset of these goals whose achievement will optimize the net benefit achieved given the costs and rewards of achieving the original goals and the costs of sensing for the antecedents (the standard PSP problem).

The most general way of dealing with conditional goals in such a case would be to accept knowledge on the antecedents in the form of *distributions*, and to use a probabilistic planner to compute the set of goals with the best expected net benefit. As an illustration, suppose the planner decided to sense the conditional goals $\mathcal{G}_c^i : \{P_1^i \leadsto G_1^i, P_2^i \leadsto G_2^i, \ldots, P_k^i \leadsto G_k^i\}$. Let us analyze the costs and benefits of this decision. First, let $\mathcal{S}(\mathcal{G}_c^i)$ denote the cost of sensing the status of the conditions $\{P_1^i \cdots P_k^i\}$. Since the results of sensing cannot be predicted during plan synthesis, to decide whether this sensing cost will be offset by the increased net benefit, the planner has to compute the *expected* net benefit achievable. In order to do this, it needs to have (or assume) some prior knowledge on how the truth values of the antecedents $\mathbf{P} : P_i$ of the conditional goals are jointly distributed. Let this distribution be $\Psi(\mathbf{P})$. Further, let $\mathcal{G}_c^i \setminus \mathbf{P}$ be the set of conditional goals that are triggered by a specific valuation of the antecedents. For each such valuation $\mathbf{P}$, the optimal net benefit achievable by the planner is $\mathcal{B}(G_o \cup [\mathcal{G}_c^i \setminus \mathbf{P}])$. The expected net benefit is $\mathbf{E}_{\mathbf{P} \sim \Psi} \mathcal{B}(G_o \cup [\mathcal{G}_c^i \setminus \mathbf{P}])$. Thus the optimal set of conditional goals to be sensed $\hat{\mathcal{G}}_c$ is computed as

$$\hat{\mathcal{G}}_c = \operatorname*{argmax}_{\hat{\mathcal{G}}_c^i \subseteq \mathcal{G}_c} \mathbf{E}_{\mathbf{P} \sim \Psi} \mathcal{B}(G_o \cup [\mathcal{G}_c^i \setminus \mathbf{P}]) - \mathcal{S}(\mathcal{G}_c^i).$$

Focusing sensing this way, while optimal, can be impractical both because of the need for distributional information, and because of the computational cost of computing optimal net benefit plans for each potential goal set. We may be forced to make reasonable *assumptions* on the distribution of the antecedents of these conditional goals, or resort to regret minimization approaches that do not require distributional information.

Conditional Goals in the Open World

While propositional conditional goals give us an understanding of the trade-offs between sensing costs and goal rewards, they are not expressive enough for open world scenarios such as USAR, where the conditional rewards are often quantified over an open set of objects. Accordingly, we will consider quantified conditional goals $\forall_x P(x) \leadsto G(x)$. Because the quantification is over an open set, it cannot be simply expanded into a set of propositional conditional goals. For example, we cannot convert a quantified conditional goal of type $\forall_x person(x) \wedge injured(x) \leadsto reportLocation(x)$ into a finite set of ground conditional goals since we do not know a priori how many persons are there, let alone which of them are injured. This makes direct application of the decision-theoretic goal selection approach described earlier infeasible. Indeed, even the naive approaches of planning for all contingencies or closing the world by sensing for everything up front are infeasible to realize, as the robot does not know how many objects may be there, and thus does not quite know when to stop sensing. Instead, we have to resort to a more incremental expansion of the quantified goal that interleaves planning and execution, and deliberately closes sensing operations

in particular parts of the world. In the following sections, we describe both our representation for quantified conditional goals, called OWQG, and our current method for handling them during planning.

3.2 Open World Quantified Goals

Open World Quantified Goals (OWQG) [Talamadupula et al. 2010] combine information about objects that *may be* discovered during execution with partial satisfaction aspects of the problem. Using an OWQG, the domain expert can furnish details about what new objects may be encountered through sensing and include goals that relate directly to the sensed objects. An Open World Quantified Goal (OWQG) is a tuple $\mathcal{Q} = \langle F, \mathcal{S}, \mathcal{P}, \mathcal{C}, \mathcal{G} \rangle$ where F and $\mathcal{S}$ are typed variables that are part of the planning problem. F belongs to the object type that $\mathcal{Q}$ is quantified over, and $\mathcal{S}$ belongs to the object type about which information is to be sensed. $\mathcal{P}$ is a predicate which ensures sensing closure for every pair $\langle f, s \rangle$ such that f is of type F and s is of type $\mathcal{S}$, and both f and s belong to the set of objects in the problem, $\mathcal{O} \in \Pi$; for this reason, we term $\mathcal{P}$ a *closure condition*. $\mathcal{C} = \bigwedge_i c_i$ is a conjunctive first-order formula where each c_i is a statement about the openness of the world with respect to the variable $\mathcal{S}$. For example, `c = (in ?hu - human ?z - zone)` with `S = ?hu - human` means that c will hold for new objects of the type "human" that are sensed. Finally $\mathcal{G}$ is a quantified goal on $\mathcal{S}$.

Newly discovered objects may enable the achievement of goals, granting the opportunity to pursue reward. For example, detecting a victim in a room will allow the robot to report the location of the victim (where reporting gives reward). Given that reward in our case is for each reported injured person, there exists a quantified goal that must be allowed partial satisfaction. In other words, the universal base [Golden and Weld 1996], or total grounding of the quantified goal on the real world, may remain unsatisfied while its component terms may be satisfied. To handle this, we rely on the partial satisfaction capability of the base planner (Section 2).

As an example, we present an illustration from our scenario: the robot is directed to "report the location of all victims". This goal can be classified as open world, since it references objects that do not exist yet in the planner's object database $\mathcal{O}$; and it is quantified, since the robot's objective is to report *all* victims that it can find. In our syntax, this information is encoded as follows.

```
1 (:open
2   (forall ?z - zone
3     (sense ?hu - human
4       (looked_for ?hu ?z)
5        (and (has_property ?hu injured)
6             (in ?hu ?z))
7   (:goal (reported ?hu injured ?z)
8          [100] - soft))))
```

In the preceding example, line 2 denotes F, the typed variable that the goal is quantified over; line 3 contains the typed variable $\mathcal{S}$, the object to be sensed.

Line 4 is the unground predicate $\mathcal{P}$ known as the closure condition (defined earlier). Lines 5 and 6 together describe the formula $\mathcal{C}$ that will hold for all objects of type $\mathcal{S}$ that are sensed. The quantified goal over $\mathcal{S}$ is defined in line 7, and line 8 indicates that it is a soft goal and has an associated reward of 100 units. Of the components that make up an open world quantified goal $\mathcal{Q}$, $\mathcal{P}$ is required[4] and F and S must be nonempty, while the others may be empty. If $\mathcal{G}$ is empty, that is, there is no new goal to work on, the OWQG $\mathcal{Q}$ can be seen simply as additional knowledge that might help in reasoning about other goals.

3.3 Handling OWQGs in the Planning System

To handle open world quantified goals, the planner grounds the problem into the closed world using a process similar to Skolemization. More specifically, we generate *runtime objects* from the sensed variable $\mathcal{S}$ that explicitly represent the potential existence of an object to be sensed. These objects are marked as system-generated runtime objects. Given an OWQG $\mathcal{Q} = \langle F, \mathcal{S}, \mathcal{P}, \mathcal{C}, \mathcal{G} \rangle$, one can look at $\mathcal{S}$ as a Skolem function of F, and runtime objects as Skolem entities that substitute for the function. Runtime objects are then added to the problem and ground into the closure condition $\mathcal{P}$, the conjunctive formula $\mathcal{C}$, and the open world quantified goal $\mathcal{G}$. Runtime objects substitute for the existence of S dependent upon the variable F. The facts generated by following this process over $\mathcal{C}$ are included in the set of facts in the problem through the problem update process. The goals generated by $\mathcal{G}$ are similarly added. This process is repeated for every new object that F may instantiate.

We treat $\mathcal{P}$ as an *optimistic closure condition*, meaning a particular state of the world is considered closed once the ground closure condition is true. On every update the ground closure conditions are checked and if true the facts in the corresponding ground values from $\mathcal{C}$ and $\mathcal{G}$ are removed from the problem. By planning over this representation, we provide a plan that is executable given the planning system's current representation of the world until new information can be discovered (via a sensing action returning the closure condition). The idea is that the system is interleaving planning and execution in a manner that moves the robot towards rewarding goals by generating an optimistic view of the true state of the world.

As an example, consider the scenario at hand and its open world quantified goal. Given two known zones, `zone1` and `zone2`, the process would generate a runtime object `human!1`. Subsequently, the facts (`has_property human!1 injured`) and (`in human!1 zone1`) and the goal (`report human!1 injured zone1`) (with reward 100) would be generated and added to the problem (where the exclamation mark (!) indicates a runtime object). A closure condition (`looked_for human!1 zone1`) would also be created. Similarly, a runtime object `human!2` would be generated and the facts (`has_property human!2 injured`) and (`in human!2 zone2`) and goal (`report human!2 injured zone2`) added to the problem, and the closure condition (`looked_for human!2 zone2`) would be created. When the planning

---

[4]If $\mathcal{P}$ were allowed to be empty, the planner could not gain closure over the information it is sensing for, which will result in it directing the robot to resense for information that has already been sensed for.

system receives an update including `(looked_for human!1 zone1)`, it will update the problem by deleting the facts `(has_property human!1 zone1)` and `(in human!1 zone1)` and the goal `(report human!1 injured zone1)` at the appropriate time point. Similar actions are taken when `(looked_for human!2 zone2)` is received. The planner must only output a plan up to (and including) an action that will make the closure condition true. Therefore once the condition becomes true, the truth values of the facts in $\mathcal{C}$ are known.

4. OVERALL SYSTEM AND INTEGRATION

The SapaReplan planner is integrated into the robotic architecture as a newly created client server that interacts directly with a *goal manager*, as detailed in Schermerhorn et al. [2009] (see Figure 4). This new server does not manage action execution, as the existing goal manager already has that capability. The planner is viewed by the goal manager, in effect, as an external library that augments its internally maintained store of procedural knowledge. When a new goal is presented, the goal manager determines whether there is a procedure already known to achieve it; if so, then that procedure is executed, otherwise the goal is sent to the planning component, which returns a script representation of a plan to achieve the goal, if one is found. In the following, we describe these parts and the integration of the system in detail.

4.1 DIARC Control Architecture

The architecture used to control the robotic agent in the aforesaid scenario (shown in Figure 3) is a subset of the *Distributed, Integrated, Affect, Reflection, and Cognition* architecture (DIARC)[5] [Scheutz et al. 2007]. DIARC is designed with human-robot interaction in mind, using multiple sensor modalities (e.g., cameras for visual processing, microphones for speech recognition and sound localization, laser range finders for object detection and identification) to recognize and respond appropriately to user requests. DIARC is implemented in the *Agent Development Environment* (ADE)[6] [Scheutz 2006], a framework that allows developers to create modular components and deploy them on multiple hosts. Each functional component is implemented as a *server*. A list of all active ADE servers, along with their functionalities, is maintained in an ADE *registry*. The registry helps in resource location, security policy enforcement, and fault tolerance and error recovery. When an ADE server requires functionality that is implemented by another component, it requests a reference to that component from the registry, which verifies that it has permission to access the component and provides the information needed for the two components to communicate directly.

---

[5]DIARC combines higher-level cognitive tasks, such as natural language understanding, with lower-level tasks, such as navigation, perceptual processing, and speech production [Brick and Scheutz 2007]. DIARC has served as a research platform for several human subject experiments.
[6]ADE combines support for the development of complex agent architectures with the infrastructure of a multiagent system that allows for the distribution of architectural components over multiple computational hosts [Scheutz 2006].
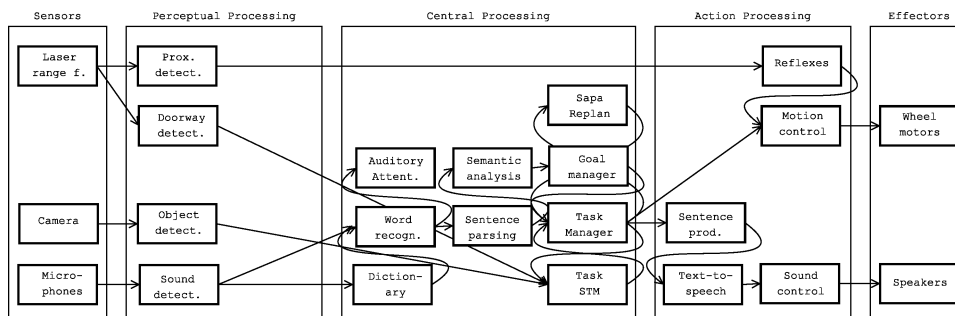
Fig. 3. A schematic of the DIARC architecture used on the robot.

The ADE *goal manager* is a goal-based action selection and management system that allows multiple goals to be pursued concurrently, so long as no resource conflicts arise. When the actions being executed for one goal present a hazard to the achievement of another goal, the goal manager resolves the conflict in favor of the goal with the higher priority, as determined by the net benefit (reward minus cost) of achieving the goals and the time urgency of each (based on the time remaining within which to complete the goals).

The goal manager maintains a "library" of procedural knowledge in the form of (1) *action scripts* which specify the steps required to achieve a goal, and (2) *action primitives* which typically interface with other ADE servers that provide functionality to the architecture (e.g., a motion server could provide an interface to the robot's wheel motors, allowing other ADE servers to drive the robot). Scripts are constructed of calls to other scripts or action primitives. Aside from this predefined procedural knowledge, however, the goal manager has no problem-solving functionality built in. Therefore, if there is no script available that achieves a specified goal, or actions are missing in a complex script, then the action interpreter fails. The addition of the planning system thus provides DIARC with the problem-solving capabilities of a standard planner in order to synthesize action sequences to achieve goals for which no prior procedural knowledge exists.

4.2 Integrating the Planner into DIARC

The integration uses a new interface to the planner to facilitate updates from the goal manager. The modified version of the planner is encapsulated as a new DIARC component that provides access to this interface to other ADE servers (although in practice, the goal manager is the only client of the planning server). The interface specifies how the goal manager can send state updates to the planner, and how the planner, in turn, can send updated or new plans to the goal manager. State updates are sent whenever relevant data of the requested type is received via sensors. In the USAR scenario that we use, for example, information about doors and boxes (which stand in for humans in our experimental runs; see Section 5) would be considered relevant. In this manner, the goal manager *filters* the information that is sent back in the form of problem updates, to avoid overwhelming the planning system. These updates can then
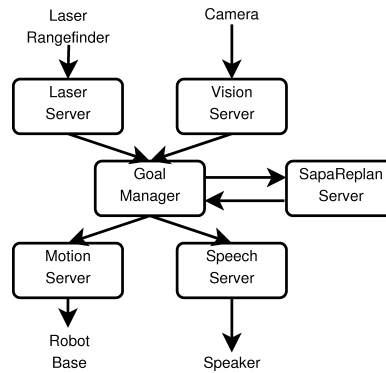
Fig. 4. A schematic showing the interaction of the SapaReplan planner server with the ADE infrastructure.

trigger a replanning process, which returns a plan in the form of action scripts that the goal manager can adopt and execute in the same way as its predefined scripts. Moreover, the new plan can be added to the goal manager's local knowledge base so that future requests can be serviced locally without having to invoke the planner. This *plan reuse* is applicable only when the relevant parts of the world remain unchanged, where relevance is determined by examining the preconditions of the actions in the plan. If there is a change in these facts due to updates to the world, ADE initiates replanning via SapaReplan.

The SapaReplan planner server starts the SapaReplan problem update monitor, specifies the planning domain, and (when applicable) the sensory update types that are of interest to the planner are sent to the goal manager (via the "attend" mechanism, described in Section 4.3 shortly), and the planner server enters its main execution loop. In this loop, it retrieves new plans from the planner (to be forwarded to the goal manager) and sends new percepts and goal status updates (received from the goal manager) to the planner. If a percept triggers replanning, the previously executing plan (and script) is discarded and a new plan takes its place.

A closely related issue that crops up when integrating a planner such as SapaReplan into a robotic architecture is that actions (and consequently plans) take time to execute on a robot and carry temporal annotations denoting the time it takes to execute them (as outlined in Section 2). Since we are executing in an open world, it is entirely possible that an action takes more time to execute than was planned. We circumvent this problem by assigning conservative time estimates to each action available to the robotic agent (and consequently the planner). If there is slack time during the execution, we simply bring forward the execution of the actions that are next in the plan. Though this approach would fail for certain types of concurrency exhibited by actions, the USAR scenario that we seek to solve does not contain any actions that need to be executed concurrently.[7] In case an action takes longer time to execute than

---

[7]Considering the fact that there is only one robotic agent that can effect changes in the world in our scenario, this is not an unreasonable assumption to make.

even the conservative estimate assigned to it (due to a failure of some nature), the planner is called into play in order to provide a new plan.

4.3 Sensing and Updates

The planner's ability to exploit opportunities requires, of course, that it be informed of changes in the environment that signal when an opportunity arises. One major issue for any robotic system operating in the real world is how to determine which small fraction of the features of the environment are of greatest salience to its goals. Resource limitations preclude a "watch out for anything" approach, necessitating some guidance with regard to how sensory processing resources should be allocated. For example, in a search and rescue scenario where victims are likely to be located in rooms, the appearance of a doorway would be of high relevance to the system's goals.

A special "attend" primitive has been defined in the goal manager to allow servers (such as the planner server) to specify which percepts are of interest. This will focus attention on those types of percepts by causing the instantiation in the goal manager of monitoring processes that communicate with other ADE servers (e.g., the vision server to detect targets of interest that are visually perceivable, the laser range finder server to detect doorways, which are detected in the range finder profile, etc.). In the case of the SapaReplan planner server, the percept types of interest are those that could prompt opportunistic replanning (e.g., detection of a new doorway might trigger a new plan to explore the room). A variety of percept types are available from various ADE servers; a subset of those most relevant to the present study are:

(1) `(robot orientation ?heading velocity ?vel location ?loc)` gives the current orientation, velocity, and location of the robot
(2) `(landmark ?name type ?t heading ?dir distance ?dist)` provides information about a perceived landmark with the given label `?name` of type `?t` (e.g., chair, table, etc.), its direction relative to the robot's heading in degrees, and its distance in meters
(3) `(box ?name color ?value heading ?dir distance ?dist)` provides information about a perceived box, including its label, color, direction relative to the robot's heading in degrees, and distance in meters
(4) `(doorway ?name heading ?dir distance ?dist)` provides information about a perceived doorway, including its direction relative to the robot's heading in degrees and distance in meters

When the monitoring process of the goal manager detects a percept in its attend list, it constructs a plan update and sends it to the planner via the planner server's update method. Updates from the goal manager can trigger the planner to replan to take advantage of detected opportunities. Plans are generated in the form of ADE action scripts which are directly executable by an action interpreter in the goal manager. Some examples of ADE scripts (both simple and complex) available to the planner for creating plans are:

(1) `(look-for ?t)` scans the room for percepts of type `?t` while turning 360 degrees
(2) `(move-to ?location)` moves to the location specified (e.g., as indicated by a landmark, for example, `(move-to chair1)`)
(3) `(turn-to ?location)` turns to face the location specified (e.g., `(turn-to box3)`)
(4) `(move-through ?doorway)` moves through the specified doorway
(5) `(report ?object ?c1 ...)` reports the given characteristics `?c1`, etc., (e.g., location, color) of the `?object`

The new plan/script is passed to the goal manager, which oversees its execution. When a plan completes, its postconditions are sent to the planner server as goal status updates. If a newly encountered percept triggers replanning, the previously executing plan is discarded and the new plan takes its place. Hence, the SapaReplan planner server can provide problem-solving capabilities to architectures constructed in the ADE infrastructure.

Example. The following example illustrates the interaction between the goal manager, the planner server, and the planner. In this case, the robot is traversing a hallway from `hall-start` to `hall-end` when it encounters a doorway (having previously added doorways to the attend list). The goal manager sends to the planner server a state update indicating that a new doorway (`door1`) has been detected. The planner server generates an update to the planner that includes the new door, but also updates the planner's representation of the environment; to begin with the planner knows only of the two locations `hall-start` and `hall-end` and the path between them (`hall-start` ↔ `hall-end`), as it has a hard goal of going to the end of the hallway. When the new doorway is detected, a new room (`room1`) is created and a new location `outside-room1` is generated and linked into the path (`hall-start` ↔ `outside-room1` ↔ `hall-end`). Similarly, the path between the hallway and the newly detected room is added (`room1` ↔ `outside-room1`). This allows the planner to generate paths into and out of the room if it determines that it is worth investigating the room (see the following for details). This update to the environment is sent to the planner by the planner server, and if the update causes a change in the currently executing plan, the resultant script is sent to the goal manager for execution.

5. EMPIRICAL EVALUATION

The integration of the robotic architecture with the planner, along with all of its attendant extensions, was evaluated via experimental runs in the USAR task scenario introduced earlier. The task at hand is the following: the robot is required to deliver essential supplies (which it is carrying) to the end of a long hallway; this is a hard goal. The hallway has doorways leading off into rooms on either side, a fact that is unknown to the robot initially. When the robot encounters a doorway, it must weigh (via the planner) the action costs and goal deadline (on the hard delivery goal) in deciding whether to pursue a search through the doorway.

Fig. 5. A Pioneer P3-AT on which the planner integration was verified.

In the specific runs described here, green boxes act as stand-ins for victims, whereas blue boxes denote healthy people (whose locations need not be reported). The experimental setup consisted of three rooms, which we represent as R_1, R_2 and R_3. The room R_1 contained a green box (GB), representing a victim; R_2 contained a blue box (BB), representing a healthy person; and R_3 did not contain a box.[8] The respective doorways leading into the three rooms R_1 through R_3 are encountered in order as the robot traverses from the beginning of the hallway to its end.

The aim of these experimental runs was to demonstrate the importance of each of the planning components that make up this integrated system, and to showcase the tight integration that was achieved in order to control the robot in this scenario. To achieve these goals, we conducted a set of experiments where we varied four parameters, each of which could take on one of two values, thus giving us 16 different experimental conditions through the scenario. The factors that we varied were:

(1) *Hard Goal Deadline*. The hard goal deadline was fixed at 100 time units, resulting in the runs in Table I, and 200 time units to give the runs in Table II.
(2) *Cost*. Presence or absence of action costs to demonstrate the inhibiting effect of costly sensing actions on the robot's search for injured people.
(3) *Reward*. Presence or absence of a reward for reporting injured people in rooms.
(4) *Goal Satisfaction*. Label the goal of reporting injured people as either soft or hard, thus modulating the bonus nature of such goals.

In the tables provided, a + symbol stands for the presence of a certain feature, while a - denotes its absence. For example, run number 5 from Table I denotes

---

[8]Although distinguishing injured humans from healthy ones in noisy environments is an interesting and challenging problem, it is not directly relevant to the core of the work being presented and evaluated.

Table I. Results of Trial Runs with a Deadline of 100 time units

Run	Cost	Reward	Soft	Enter R_1	Report GB	Enter R_2	Report BB	Enter R_3
1	+	+	+	Yes	Yes	No	No	No
2	+	+	-	Yes	Yes	$\perp$	$\perp$	$\perp$
3	+	-	+	No	No	No	No	No
4	+	-	-	Yes	Yes	$\perp$	$\perp$	$\perp$
5	-	+	+	Yes	Yes	No	No	No
6	-	+	-	Yes	Yes	$\perp$	$\perp$	$\perp$
7	-	-	+	No	No	No	No	No
8	-	-	-	Yes	Yes	$\perp$	$\perp$	$\perp$

$\perp$ denotes that there is no feasible plan from that point on that fulfils all hard goals.

an instance where the deadline on the hard goal (going to the end of the hallway) was 100 time units, action costs were absent, the open world goal of reporting people carried reward, and this goal was classified as soft.

The experimental runs detailed in this section were obtained on a Pioneer P3-AT robot (see Figure 5) as it navigated the USAR scenario with the initial hard goal of getting to the end of the hallway, while trying to accrue the maximum net benefit possible from the additional soft goal of reporting the location of injured people. A video of the robot performing these tasks can be viewed via the following link:

`http://hri.cogs.indiana.edu/videos/USAR.avi`

The robot starts at the beginning of the hallway, and initially has a plan for getting to the end in fulfilment of the original hard goal. An update is sent to the planner whenever a doorway is discovered, and the planner subsequently replans to determine whether to enter that doorway. In the first set of runs, with a deadline of 100 units on being at the end of the hallway, the robot has time to enter only the first room, R_1 (before it must rush to the end of the hallway in order to make the deadline on the hard goal).

Even with this restriction, some interesting plans are generated. The planner directs the robot to enter R_1 in all the runs except 3 and 7. This can be attributed to the fact that there is no reward on reporting injured people in those cases, and the reporting goal is soft; hence the planner does not consider it worthwhile to enter the room and simply ignores the goal on reporting. The alert reader may ask why it is not the case that entering R_1 is skipped in runs 4 and 8 as well, since there is no reward on reporting injured people in those cases either; however, it must be noted that this goal is hard in cases 4 and 8, and hence the planner *must* plan to achieve it (even though there may be no injured person in that room, or reward to offset the action cost). This example illustrates the complex interaction between the various facets of this scenario (deadlines, costs, rewards, and goal satisfaction), and shows how the absence of even one of these factors may result in the robot being unable to plan for opportunities that arise during execution, in this case, detecting and reporting injured people.

When the deadline on reaching the end of the hallway is extended to 200 units, the robot is afforded enough time to enter all the rooms. In such a scenario, it is expected that the robot would enter all the rooms to check for victims, and this is indeed what transpires, except in runs 11 and 15. In those

Table II. Results of Trial Runs with a Deadline of 200 time units

Run	Cost	Reward	Soft	Enter R_1	Report GB	Enter R_2	Report BB	Enter R_3
9	+	+	+	Yes	Yes	Yes	No	Yes
10	+	+	-	Yes	Yes	Yes	No	Yes
11	+	-	+	No	No	No	No	No
12	+	-	-	Yes	Yes	Yes	No	Yes
13	-	+	+	Yes	Yes	Yes	No	Yes
14	-	+	-	Yes	Yes	Yes	No	Yes
15	-	-	+	No	No	No	No	No
16	-	-	-	Yes	Yes	Yes	No	Yes

runs, the robot skips all rooms for precisely the same reasons outlined before (for runs 3 and 7): the lack of reward for reporting the goal, combined with the softness of that goal. Indeed, runs 3 and 7 are respectively identical to runs 11 and 15 save the longer deadline on the hard goal.

Another interesting observation is that in all the cases where the robot *does* enter R_2, it refuses to report the blue box (BB), since there is no reward attached to reporting blue boxes (nonvictims). Since the deadline is far enough away for runs 9 through 16, the planner never fails to generate a plan to enter rooms in order to look for injured people, avoiding the situation encountered in runs 2, 4, 6, and 8 where there is no feasible plan that fulfils all hard goals since the robot has run out of time (denoted $\perp$ in Table I).

In terms of computational performance, the planning time taken by the planning system was typically less than one second (on the order of a hundred milliseconds). Our empirical experience thus suggests that the planning process always ends in a specific, predictable time frame in this scenario (an important property when actions have temporal durations and goals have deadlines). Additionally, in order to test the scale-up of the system, we evaluated it on a problem instance with ten doors (and consequently more runtime objects) and found that there was no significant impact on the performance.

These runs thus confirm the importance of the three main components of the planning system that directs the robot in this USAR scenario—without replanning, the system would not be able to take new information about doorways and rooms connected to them into account; without support for soft goals, the planner may fail to return a plan given an overconstrained problem; and without an open world representation, the planner would be unable to reason about new objects (doorways, rooms, injured persons) that result in the fulfilment of new goals.

6. RELATED WORK

In this article, we focused on planning support for robots where human-robot teams work in tandem to solve complex problems in open world scenarios. Although there has not been any prior work that directly addresses planning for human-robot teaming in open worlds, there does exist a rich body of work that is related to various aspects of our overall problem. As shown in Figure 6, this related work can be classified into three parts: human-robot interaction, human-planner interaction, and planner-robot interaction. Specifically:
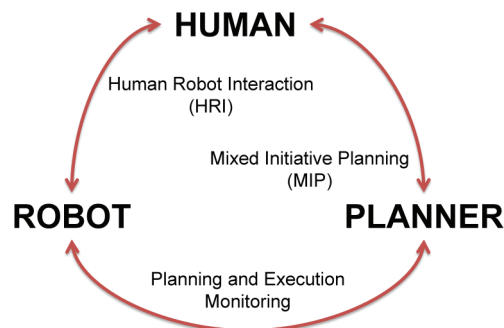
Fig. 6. The various modes of interaction in human-robot scenarios.

—planning and execution monitoring deals with the interactions between a fully autonomous robot and a planner;
—Human-Robot Interaction (HRI) works toward smooth interactions between a human user and a robot;
—mixed initiative planning relates to interactions between humans who are receiving plans and the automated planners that generate them.

Since our focus is on how a planner fits into human-robot teams, we are most interested in work that relates to planning and execution monitoring and mixed initiative planning. There has been significant work in planning and execution monitoring, often in the context of replanning and contingent planning. Contingent planners (refer to Albore et al. [2009] and Meuleau and Smith [2003]) can be viewed as solving for the problem of execution monitoring by assuming full sensing knowledge is available at execution time, so no replanning would ever be necessary. However, as Gat [1992] has pointed out, in designing a planner whose ultimate goal is finding plans for execution, it is difficult (and sometimes impossible) to model for all contingencies, and often it is better to design an execution monitoring system that is capable of recognizing failures (i.e., *cognizant failures* [Firby 1989]). That is, we can relax the problem for the planner by removing uncertainty in the world. Agre and Chapman [1990] also discuss these issues in relationship to planning and execution monitoring and viewing "plans as advice."

A number of systems (refer to [Lemai and Ingrand 2003], [Knight et al. 2001], [Myers 1998]) have worked by performing execution monitoring and subsequent *plan repair* or replanning upon the discovery of an inconsistent execution state. For instance, the CASPER planner [Knight et al. 2001] performs plan repair upon failure. While the IxTeT-eXeC [Lemai and Ingrand 2003] system attempts a similar repair strategy, it replans only if no repair can be found. It handles the arrival of new goals through replanning.

The work that is most closely related to our system, however, seems to be Bagchi et al.'s [1996] system for controlling service robots. The emphasis of their work is on the robotic agent's capability to not only plan and act autonomously, but also to do so in an *interactive* way such that the user's comfort and safety are kept in mind. In order to achieve this, the robot is equipped to comprehend

the user's (changing) goals and advice at different levels of detail. In turn, the planner can refine and modify these goals dynamically and react to unexpected changes in the environment. This system thus includes the human user in the loop via interaction with the robot and a probabilistic planner.

However, one critical area where the action selection mechanism employed by Bagchi et al. will fail is when sensing and/or sensing actions are expensive. Sensing is critical to real-world applications, since most scenarios involve partial knowledge of the world state and the system needs a mechanism to update itself of changes and new information in the world. Our system handles this problem by closing the loop shown in Figure 6: the planner interacts with the robot using the ADE architecture and the SapaReplan execution monitor, while simultaneously providing the human user a way of interacting with it via the specification of new information and goals through OWQGs.

Handling an open environment using a closed world planner has been considered before, notably in the work of Etzioni et al. [1997] via the specification of *Local Closed-World* (LCW) statements. However, there exists at least one major difference between their work and this attempt. We note that the representation used in that work, of closing a world that is open otherwise via the LCW statements, is complementary to our representation. Since our interest in providing support for open world quantified goals is to relax our planner's assumption of a world closed with respect to object creation, we are *opening* parts of a completely closed-world with the aid of OWQGs. This approach provides a method of specifying *conditional goals*, where goal existence hinges upon the truth value of facts.

Semantics of goals involving sensing have received attention in Scherl and Levesque [1993] and Golden and Weld [1996]. The latter work is particularly relevant as they consider representations that leads to tractable planning, and propose three annotations `initially`, `hands-off` and `satisfy` to specify goals involving sensing. A conditional goal $A \rightsquigarrow B$ will translate, in their notation, to $\texttt{initially}(A) \Rightarrow \texttt{satisfy}(B)$. Conditional goals require sensing the antecedent's truth in the initial state to decide whether to pursue the reward offered by the consequent. In this sense, they are inherently temporal (a point noted also by Golden and Weld [1996]). There has been significant work on "temporal goals" [Bacchus and Kabanza 1996; Baral et al. 2001], and "trajectory constraints" [Gerevini et al. 2009].

An important difference, however, is that earlier work focused on problems with completely known initial states (where, as we saw in Section 3.1, these richer goals can be compiled down to goals of achievement). Our interest is on handing conditional goals with incomplete information about initial states (as is the norm in open world scenarios we deal with). Here, conditional goals present interesting trade-offs between goal rewards and sensing costs.

On the "planners interacting with humans" side, there have been some planning systems that work toward accepting input from users. In particular, work by Myers [1996] has dealt specifically with *advisable planning* (i.e., allowing a human to specify partial plans, recommendations of goals and actions, or methods to evaluate plan quality; all in natural language). The *Continuous*

Planning and Execution framework, also developed by Myers [1998], contained such a framework allowing natural language advice. This system provided for plan execution monitoring and initiated plan repairs when necessary (though appears to have never handled fully open world scenarios). Another system that relies on high-level advice from a human is TRAINS-95 [Ferguson et al. 1996]. This system engages the human in a dialog, explicitly eliciting advice from the user and asking for the best way to complete tasks at the high level, while the planner engages in planning using more primitive actions.

7. CONCLUSION

In this article, we focused on the challenges of adapting planning technology to applications involving human-robot teaming. Our motivating problem is an Urban Search And Rescue (USAR) scenario where a human is in remote contact with an autonomous robot and provides high-level instructions to it. We noted that several aspects of state-of-the-art planning technology, such as temporal planning and partial satisfaction planning, can be imported out-of-the-box; in particular we use *SapaReplan* as the base planner. We then showed that the teaming problem also presents a critical challenge, namely the need to handle open worlds. Given that existing planners operate under closed world assumptions, we had to focus on effective ways of enabling them to handle open world requirements. Of particular interest in the USAR scenario is the fact that the most important goals (reporting on wounded people) are "conditional", in that the planner and the robot do not know where the injured people are. To capture this, we investigated the general notion of conditional goals, and showed how they foreground the trade-offs between goal reward and sensing cost. We then developed an approach to handle a specific form of conditional goals called open world quantified goals. We discussed the details of integrating the planner and robot, and presented an empirical evaluation of the effectiveness of our solution.

A fruitful line of extension for this work is to handle open world quantified goals more generally (refer to Section 3.1) without the optimistic sensing assumption. We believe that sampling-based planning techniques such as hindsight optimization [Yoon et al. 2008] and anticipatory planning [Hubbe et al. 2008] would be useful in better balancing sensing costs with expected reward. We are also considering methods of performing domain analysis to determine what objects should be attended to by the DIARC architecture before plan execution begins.

Finally, although we focused only on conditional goals, open worlds also present challenges involving counterfactual knowledge. For example, whereas a statement such as "a door implies a room" can be evaluated and used at "plan time" in classical planning scenarios, in open worlds, where the full map may not be known, the existence of a door can only be sensed during execution. Any use of this knowledge during planning will make the plan "speculative" in that its robustness is subject to the outcomes of sensing during execution. It would thus be worthwhile to develop both representations and planning methods

for handling such counterfactual domain knowledge. OWQGs already provide rudimentary representation support.

Indeed, as we mention at the end of Section 3.2, "*If $\mathcal{G}$ is empty, that is, there is no new goal to work on, the OWQG $\mathcal{Q}$ can be seen simply as additional knowledge that might help in reasoning about other goals.*" Such goalless OWQGs can be seen as sensing-based domain axioms, in that they allow the planner to deduce more world facts based on sensing results. The presence of such knowledge at planning time allows the planner to make counterfactual plans whose success is predicated on specific sensing results. A naive planner might consider such counterfactual plans to be on par with normal ones, and pick the least expensive. The problem with this approach is that the least expensive plan may be "wishful" in that it is predicated on sensing results that are unlikely. Generating robust plans in such scenarios is a challenge that we hope to address in future work.

REFERENCES

AGRE, P. AND CHAPMAN, D. 1990. What are plans for? *Robot. Auton. Syst. 6,* 1-2, 17–34.

ALBORE, A., PALACIOS , H., AND GEFFNER, H. 2009. A translation-based approach to contingent planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 09)*. 1623–1628.

BACCHUS, F. AND KABANZA, F. 1996. Planning for temporally extended goals. In *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 2. 1215–1222.

BAGCHI, S., BISWAS, G., AND KAWAMURA, K. 1996. Interactive task planning under uncertainty and goal changes. *Robot. Auton. Syst. 18*, 1, 157–167.

BARAL, C., KREINOVICH, V., AND TREJO, R. 2001. Computational complexity of planning with temporal goals. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01)*. 509–514.

BENTON, J., DO, M., AND KAMBHAMPATI , S. 2009. Anytime heuristic search for partial satisfaction planning. *Artif. Intell. 173*, 5-6, 562–592.

BRICK, T. AND SCHEUTZ, M. 2007. Incremental natural language processing for HRI. In *Proceedings of the 2nd ACM IEEE International Conference on Human-Robot Interaction.* 263–270.

CANTRELL, R., SCHEUTZ, M., SCHERMERHORN, P., AND WU, X. 2010. Robust spoken instruction understanding for HRI. In *Proceedings of the Human-Robot Interaction Conference*.

CUSHING, W., BENTON, J., AND KAMBHAMPATI, S. 2008. Replanning as deliberative re-selection of objectives. Tech. rep., CSE Department, Arizona State University.

DO, M. AND KAMBHAMPATI, S. 2002. Planning graph-based heuristics for cost-sensitive temporal planning. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems (AIPS'02)*. Vol. 2.

DZIFCAK, J., SCHEUTZ, M., BARAL, C., AND SCHERMERHORN, P. 2009. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proceedings of the International Conference on Robotics and Automation.*

ETZIONI, O., GOLDEN, K., AND WELD, D. S. 1997. Sound and efficient closed-world reasoning for planning. *Artif. Intell. 89*, 1-2, 113–148.

FERGUSON, G., ALLEN, J., AND MILLER, B. 1996. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the 3rd Conference on Artificial Intelligence Planning Systems (AIPS-96)*. 70–77.

FIRBY, R. 1989. Adaptive execution in complex dynamic worlds. Tech. rep., Yale University, New Haven, CT.

GAT, E. 1992. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the National Conference on Artificial Intelligence*. 809–809.

GEREVINI, A., HASLUM, P., LONG, D., SAETTI, A., AND DIMOPOULOS, Y. 2009. Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. *Artif. Intell. 173*, 5-6, 619–668.

GOLDEN, K. AND WELD, D. S. 1996. Representing sensing actions: The middle ground revisited. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*. 174–185.

HUBBE, A., RUML, W., YOON, S., BENTON, J., AND DO, M. 2008. Online anticipatory planning. In *Proceedings of the ICAPS'08 Workshop on a Reality Check for Planning and Scheduling under Uncertainty*.

KAMBHAMPATI, S. 2007. Model-Lite planning for the Web age masses: The challenges of planning with incomplete and evolving domain theories. In *Proceedings of the AAAI '07 Conference on Artificial Intelligence*.

KNIGHT, R., RABIDEAU, G., CHIEN, S., ENGELHARDT, B., AND SHERWOOD, R. 2001. Casper: Space exploration through continuous planning. *IEEE Intell. Syst.,* 70–75.

LEMAI, S. AND INGRAND, F. 2003. Interleaving temporal planning and execution: IxTeT-eXeC. In *Proceedings of the ICAPS Workshop on Plan Execution*.

MEULEAU, N. AND SMITH, D. 2003. Optimal limited contingency planning. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*.

MYERS, K. 1996. Advisable planning systems. *Adv. Plan. Technol.*, 206–209.

MYERS, K. 1998. Towards a framework for continuous planning and execution. In *Proceedings of the AAAI Fall Symposium on Distributed Continual Planning*.

SCHERL, R. B. AND LEVESQUE, H. J. 1993. The frame problem and knowledge-producing actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 689–695.

SCHERMERHORN, P., BENTON, J., SCHEUTZ, M., TALAMADUPULA, K., AND KAMBHAMPATI, S. 2009. Finding and exploiting goal opportunities in real-time during plan execution. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

SCHEUTZ, M. 2006. ADE - Steps towards a distributed development and runtime environment for complex robotic agent architectures. *Appl. Artif. Intell. 20*, 4-5, 275–304.

SCHEUTZ, M., SCHERMERHORN, P., KRAMER, J., AND ANDERSON, D. 2007. First steps toward natural human-like HRI. *Auton. Robot. 22,* 4, 411–423.

TALAMADUPULA, K., BENTON, J., SCHERMERHORN, P., SCHEUTZ, M., AND KAMBHAMPATI, S. 2010. Integrating a closed-world planner with an open-world robot. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

YOON, S., FERN, A., AND GIVAN, R. 2007. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the International Conference on Autonomous Planning and Scheduling (ICAPS'07)*. 352–359.

YOON, S., FERN, A., GIVAN, R., AND KAMBHAMPATI, S. 2008. Probabilistic planning via determinization in hindsight. In *Proceedings of the AAAI Conference on Artificial Intelligence*.