Integrating a Closed World Planner with an Open World Robot: A Case Study

Kartik Talamadupula[†] and **J. Benton**[†] and **Paul Schermerhorn**[§] and
Subbarao Kambhampati[†] and **Matthias Scheutz**[§]

[†]**Department of Computer Science**
Arizona State University
Tempe, AZ 85287 USA
{krt,j.benton,rao} @ asu.edu

[§]**Cognitive Science Program**
Indiana University
Bloomington, IN 47406 USA
{pscherme,mscheutz} @ indiana.edu

Abstract

In this paper, we present an integrated planning and robotic architecture that actively directs an agent engaged in an urban search and rescue (USAR) scenario. We describe three salient features that comprise the planning component of this system, namely (1) the ability to plan in a world open with respect to objects, (2) execution monitoring and replanning abilities, and (3) handling soft goals, and detail the interaction of these parts in representing and solving the USAR scenario at hand. We show that though insufficient in an individual capacity, the integration of this trio of features is sufficient to solve the scenario that we present. We test our system with an example problem that involves soft and hard goals, as well as goal deadlines and action costs, and show via an included video that the planner is capable of incorporating sensing actions and execution monitoring in order to produce goal-fulfilling plans that maximize the net benefit accrued.

Introduction

Consider the following problem: a human-robot team is actively engaged in an *urban search and rescue* (USAR) scenario inside a building of interest. The robot is placed inside this building, at the beginning of a long corridor; a sample layout 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. The corridor 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 corridor 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 a new soft goal, to report the location of victims.

It is natural to encode this soft goal as a quantified goal, since it is expected that the planner will report the location of as many victims as it can find given its time and cost constraints. The planner must reason about the 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. This can be modeled as a partial satisfaction planning (PSP) problem, and solved using planners that can handle PSP problems.

Unfortunately, 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 fallibility of human experts in completely specifying information relevant to the given problem and goals up-front, makes it quite likely that information crucial to achieving some soft goals may be specified at some later stage *during* the planning process. In our USAR scenario, for example, the knowledge that victims are in rooms may be relayed to the planner while it is engaged in planning for the executing robot. In order to handle the specification of such statements in the midst of an active planning process, and enable the use of knowledge thus specified, we need to relax two other crucial assumptions that most modern planners rely on. The first is the closed world assumption with respect to the constants (objects) in the problem—the planner can no longer assume that the only objects in the scenario are those that are mentioned in the initial state. The other modification requires that we interleave planning with execution monitoring and, if required, replanning in order to account for the new information.

In this paper, we explore the issues involved in engineering an automated planner to guide a robot towards maximizing net benefit accompanied with goal achievement in such scenarios. The planning problem that we face involves partial satisfaction (in that the robot has to weigh the rewards of the soft goals against the cost of achieving them); it also requires replanning ability (in that the robot has to modify its current plan based on new goals that are added). An additional (perhaps more severe) complication is that the planner needs to handle goals involving objects whose existence is not known in the initial state (e.g., the location of the humans to be rescued in our scenario). To handle these issues, we introduce a new kind of goal known as the *Open World Quantified Goal* (OWQG) that provides for the spec-

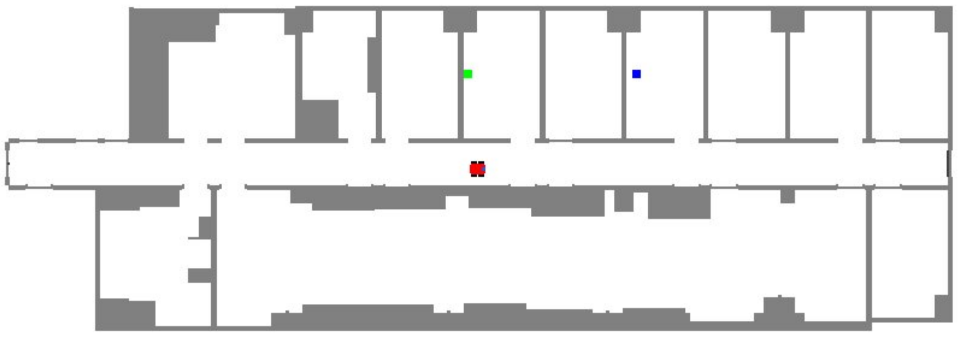

Figure 1: A map of a sample scenario; boxes are stand-ins for humans, where green indicates injured and blue indicates normal.

ification of information and creation of objects required to take advantage of opportunities that are encountered during plan execution. Using OWQGs, we can bias the heuristic's view of the search space towards finding plans that achieve additional reward in an open world.

Architecture

The architecture used to control the robotic agent in the above scenario (shown in Figure 2) is a subset of the *distributed, integrated, affect, reflection and cognition* architecture (DIARC) (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) (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.

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 pre-defined 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 integration 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. The integration is accomplished by running the planner in the context of a newly reated ADE server, as detailed in (Schermerhorn et al. 2009). The planner server implements an interface that allows the goal manager to submit goals and status updates. Plans are returned to the goal manager as new ADE action scripts which can then be executed in the same way as pre-defined scripts.

The planner's ability to exploit opportunities (see below) 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.

In order to support perceptual monitoring for world features that are relevant to the planner, the goal manager allows other servers to specify which types of percepts (such as doorways in the example above) should be monitored and reported back to the server when detected. Hence, the planner server can specify which types of percepts are of interest (i.e., could cause it to update the plan), which effectively fo-

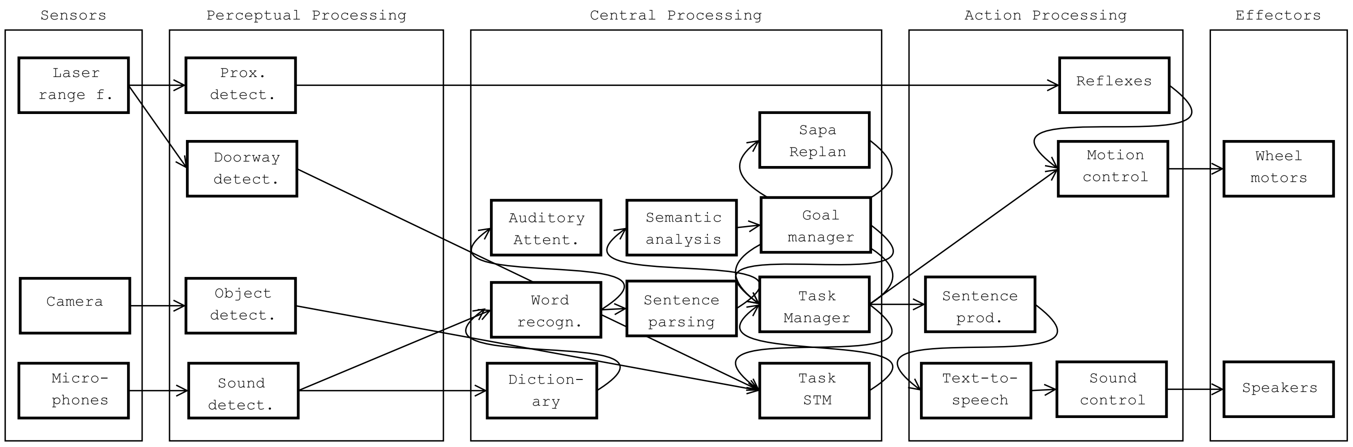

Figure 2: A schematic of the relevant parts of the DIARC architecture used in the experimental evaluation on the robot.

cusses attention on those types by causing the goal manager to instantiate monitoring processes that communicate with other `ADE` servers such as the laser range finder server or the vision server.

Specific to the planner are percept types that could prompt opportunistic replanning via the detection of new objects, as defined in (Schermerhorn et al. 2009). We maintain a list of these percept types that the planner needs to be monitored, known as an *attend* list. When the goal manager detects, or is informed of, the presence of one of the percepts in this attend list, a state update is constructed and sent to the planning system. These updates may trigger the planner to replan in order to take advantage of opportunities thus detected. Similarly, when a plan (i.e., script generated by the planner) completes execution, the goal manager sends an update of the exit status of the plan (i.e., the postconditions achieved, if any). If a percept triggers replanning, the previously executing plan (and script) is discarded and a new plan takes its place.

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 (previously added 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` $\leftrightarrow$ `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` $\leftrightarrow$ `outside-room1` $\leftrightarrow$ `hall-end`). Similarly, the path between the hallway and the newly-detected room is added (`room1` $\leftrightarrow$ `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 below for details). This update to the environment is sent to the planner by the planner server, and if the update causes a plan update, the resultant script is sent to the goal manager for execution.

The integration of the planner with the architecture also provides another very important functionality: it allows DIARC to perform opportunistic planning via the perceptual monitoring process described above. To enable the use of these opportunities, the planner is modified via the relaxation of assumptions relating to (1) the closed nature of the world with respect to object creation; (2) the specification of all goals as hard goals, in order to avail opportunities that enable the achievement of goals previously considered unachievable; and (3) the separation of the planning and execution stages, to allow for execution monitoring and replanning. These relaxations are described in greater detail in the next section.

Planning in an Open World

The `ADE` planning server wraps around *SapaReplan* (Cushing, Benton, and Kambhampati 2008), a forward state-space planner based on *SapaPS* (Do and Kambhampati 2004). *SapaReplan* adds the ability to handle updates to the state through the use of a monitor process. We additionally introduce a novel goal construct called an *open world quantified goal* (OWQG) that combines information about the open world and partial satisfaction aspects of the problem.

Goals in an Open World

Our approach seeks to open the world by allowing statements, called open world quantified goals, that label sections of the domain as open with respect to objects. Using these, the domain expert can furnish details about when new objects may be encountered through sensing and include goals that relate directly to the sensed objects. This can be seen as a complementary approach to handling open world envi-

Figure 3: A Pioneer P3-AT on which the planner integration was verified.

ronments using *local closed world* (LCW) information produced by sensing actions (Etzioni, Golden, and Weld 1997).

An open world quantified goal (OWQG) is defined as a tuple $\mathcal{Q} = \langle F, \mathcal{S}, \mathcal{P}, \mathcal{C}, \mathcal{G} \rangle$. Here, F and $\mathcal{S}$ are typed variables that are part of the problem Π, where 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 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). Note that reward in our case is for each reported injured person. As such, there exists a quantified goal that must be allowed partial satisfaction. In other words, the universal base (Weld 1994), 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 use partial satisfaction planning (PSP) (van den Briel et al. 2004), where the objective is to maximize the difference between the reward given to goals, and the cost of actions. Reward is given for each term $g \in \mathcal{G}$ satisfied, $u(\mathcal{G})$. Additionally each term g is considered soft in that it may be "skipped over" and remain unachieved.

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:

```
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 example above, line 2 denotes F, the typed variable that the goal is quantified over; line 3 contains the typed variable $\mathcal{S}$ about which information is 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[1] and F and S must be non-empty, while the others may be empty. If $\mathcal{G}$ is empty, i.e., 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.

Interleaving Planning and Execution

For most of the sensors on the robot, it is too expensive to sense at every step, so knowing exactly when to engage in perceptual monitoring is of critical importance. Low-level sensing for navigation is handled through action scripts, but for more expensive, high-level operations we use OWQGs. Planning through an open world introduces the possibility of dangerous faults or nonsensical actions. While in some sense, this can be quantified with a risk measure (see (Garland and Lesh 2002), for example), indicating the risk of a plan does nothing to address those risks. A more robust approach in an online scenario involves *planning to sense* in a goal-directed manner. When plans are output to the `ADE` *goal manager*, they include all actions up to and including any action that would result in closure (as specified by the *closure condition*).

Problem Updates and Replanning Regardless of the originating source, the monitor receives updates from the `ADE` *goal manager* 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.

As discussed in (Cushing and Kambhampati 2005), providing for updates to the planning problem allows us to look at unexpected events in the open world as new information

[1]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 re-sense for information that has already been sensed for.

rather than faults to be corrected. In our setup, problem updates cause the monitor process to immediately stop the planner (if it is running) and update its internal problem representation. The planner is then signaled to replan on the new problem. In the presence of reward and action cost[2], the replanning process also allows the planner to exploit new opportunities, potentially finding plans that may achieve better net benefit than previous ones. For example, if a new doorway is discovered, that immediately entails a room and the potential opportunity to achieve more net benefit (by looking for and perhaps finding an injured person). Similarly, if a new *hard* goal arrives with a closely approaching deadline, the planner can generate a new plan that directly achieves it, ignoring soft goals. When a plan is found, it is announced to the `ADE` *goal manager*, which then performs its analysis to find conflicts that may occur in the control mechanisms of the robot.

Implementation

One question that arises when planning in open world scenarios is how to direct actions in the presence of unknown facts. To tackle this problem we use the conjunctive formula from the OWQG in an optimistic manner, biasing the heuristic's model of the world into resolving uncertainty. Since the scenario calls for the planner to work toward achieving higher *net benefit*, the system makes assumptions about facts that can lead to rewarding goals. In other words, the planning system assumes that certain unknown facts are true to achieve greater reward. This conceptual representation of the potential search space is then used to generate plans.

The methodology involves grounding the problem into the closed world using a process similar to Skolemization. More specifically, we generate *runtime objects* from the sensed variable S that explicitly represent the potential existence of an object to be sensed. These objects are represented with a suffixed exclamation mark on the object type, followed by a number (e.g., `human!1`). 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 $\mathcal{G}$. In other words, 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 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 heuristic 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 towards rewarding goals by generating an optimistic view of the true state of the world. [3]

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. 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 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 apprioriate 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, since the idea behind the closure condition is that after it becomes true we can expect closure on certain aspects of the world. Therefore once the condition becomes true, the truth values of the facts in $\mathcal{C}$ are known.

Evaluation

We evaluted the planner's integration with the robotic architecture in the the USAR task scenario introduced earlier: the robot is required to deliver supplies to a location at the end of a corridor, but may also encounter doorways to rooms in which victims might be found. 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 conduct a search through the doorway. The estimated duration $d(traverse\_hallway)$ is 50 seconds, while $d(search) = 35$. The hard goal deadline on delivery, $dl(delivery)$, is varied for purposes of evaluation, while the utility and cost $u(delivery)$ and $c(delivery)$ are fixed at 1000 and 50, respectively.

In the experiments described here, the victims are represented by green boxes; whenever the robot enters a room and identifies a green box, it accrues the utility $u(report\_victim)$ (fixed at 100 for these evaluations). The experimental setup used in these evaluations has a green box (victim) in the first room encountered, a blue box (non-victim) in the second room, and no box in the third room.

[2]In our scenario, the action costs are determined by a combination of the temporal and resource costs incurred by the robot.

[3]This approach is reminiscent of the probablistic planner FF-Replan (Yoon, Fern, and Givan 2007), winner of the International Probablistic Planning Competition in 2004, which solves an optimistic determinization of a given probablistic planning problem at each state it encounters.

Row No.	Cost	Time limit	Green Box Room 1	Green Box Reported	Blue Box Room 2	Blue Box Reported	No Box Room 3	No Box Reported	Status	Net Benefit
1	50	30	-	-	-	-	-	-	Failure	0
2	50	60	Pass	-	Pass	-	Pass	-	Success	950
3	50	90	Enter	Yes	Pass	-	Pass	-	Success	1000
4	50	120	Enter	Yes	Enter	No	Pass	-	Success	950
5	50	160	Enter	Yes	Enter	No	Enter	No	Success	900
6	100	30	-	-	-	-	-	-	Failure	0
7	100	160	Pass	-	Pass	-	Pass	-	Success	950

Table 1: Results of trial runs with various search costs and time limits.

After the third room is passed, the delivery destination referenced in the hard goal is reached. Initially the planner is given information only regarding its initial location (the beginning of the hallway), goal location (the end of the hallway) and connectivity between the two locations. When the robot discovers doorways, it sends an update regarding a new hallway location (i.e., in front of doorway), a new room location and updates connectivity accordingly.

The scenario was tested with search costs $c(search)$ equal to 50 and 100. Table 1 presents the outcomes of the evaluation runs. An update is sent to the planner whenever a doorway is discovered, and the planner subsequently re-plans to determine whether to enter the doorway. Without handling open world quantified goals, the robot would skip entering all doorways—there would be no way for the planner to know that reward may be gained by entering the doorways. However, with these new constructs, the planner is able to reason about entering doors. When there is insufficient time to achieve the delivery goal (i.e, $dl(delivery) = 30$), the planner returns no plan, so the robot does not act. As $c(search) = 50$ is maintained constant and the hard deadline increases, the number of rooms the robot can afford to explore also increases. After each search, the closure condition is satisfied (i.e., $\mathcal{P}$, that we have looked for a box), making searching for boxes at that location superfluous (given the assumption of a closed world in the room via the closure condition).

The results presented in this paper were generated on a Pioneer P3-AT robot (see Figure 3) as it navigated the scenario presented previously. The goal of the planner in guiding the robot was to achieve the hard goal of getting to the end of the corridor, while trying to accrue the maximum net benefit possible from the additional soft goal of reporting the location of injured humans. A video of the robot performing these tasks can be viewed via the following link:
`http://hri.cogs.indiana.edu/videos/USAR.avi`

Discussion

It has been a regrettable reality in planning ranks that as the time required to generate a complete plan has decreased, so too has the ability to encode interesting details about the world. This has led to a situation where state-of-the-art planners can only deal with a subset of the features necessary to encode any domain of interest with a real world perspective. To be sure, there do exist planners that can handle the expressivity required to model some or all of the problems that delineate our USAR scenario from existing planning benchmarks (Penberthy and Weld 1992; Golden, Etzioni, and Weld 1994); unfortunately, none of these planners combine all the features necessary to solve our problem in the real world. Planning technologies and systems have been analyzed previously (Smith 2003) in order to move these techniques closer to being able to step up and perform in the real world (albeit other-worldly) domains used at NASA.

We took a similar approach towards our problem—we first considered the assumptions we would need to relax in order for a state-of-the-art planner to be able to reason about and solve our problem. In doing so, we found that these assumptions—about the closed world, the separation of planning and execution stages, and all goals being hard—had a direct correspondence to the features that we wished to model. We also discovered the lack of a planner that combines solutions to these problems in one integrated system.

The first (and in some ways most important) assumption we had to relax was one that most modern day planners have come to take for granted—the assumption that the world is closed with respect to facts, objects and operators. Since our scenario involved the specification of new knowledge concerning the world at any stage during the robot's progress, we had to allow for the possibility that there may be objects in the world that are not specified to the robot initially. We retained the closed nature of the world with respect to operator templates, since it is only reasonable that the robot is made aware of its capabilities initially and does not gain any additional powers on the way.

In order to deal with objects that the robot may either discover or *attempt* to discover to achieve some reward, it is essential that the planner not close its possibilities with regard to objects in the world. To enable this, we introduced the open world quantified goals defined in the previous section and equipped the planner with a mechanism to parse and use the information specified within these goals.

The issue of planning with an open versus closed world representation has been dealt with before, notably in the work of Etzioni et al. (Etzioni, Golden, and Weld 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.

However, the ability to recognize and represent new objects that appear during execution in the world means nothing if the planner cannot actively use these objects in order to output new plans that improve the quality metric (be it time, cost or net -benefit). To support this requirement, we had to endow our planning system with the capability to interleave planning and execution monitoring, so that changes to the world could be transmitted to the planner in the form of updates and subsequently parsed into the planner's database, as outlined in the previous section. This kind of *online planning* seems to be a case-restrictive yet simple solution to the complex problem of dealing with sensing actions. Specifically, this approach seems to be restricted to problems that contain simple reward models where it is reasonable to take a greedy approach.

It may be argued that XII, the planning system used by Etzioni et al. (Etzioni, Golden, and Weld 1997) handles the twin problems of an open world and interleaving planning and execution monitoring described above. However, these two features alone are not sufficient to model the USAR scenario. As described in earlier sections, we wanted the robot (and the planner that creates plans for it to execute) to look at tasks like reporting the presence of victims in rooms as *opportunities*, rather than as hard goals that *must* be achieved. That is to say, we wanted the robot to do its utmost to satisfy the quantification implied in the open world quantified goal, and to report the location of all victims; however, we did not want the pursuit of this objective to cloud the primary goal, which remained getting to the end of the corridor.

To handle this problem, we needed to consider a third relaxation—one that allowed for some goals to be specified as *soft* goals. We used ideas from the field of Partial Satisfaction Planning (PSP) (Smith 2004) and the planner SapaPS (Benton, Do, and Kambhampati 2009) to include support for reasoning about soft goals and net benefit, as specified in the previous section. Enabling the usage of soft goals mitigates some of the more difficult problems in fully open worlds—in the USAR scenario, when certain rooms are completely undiscoverable, it is infeasible to expect complete satisfaction of certain quantified goals.

Limitations and Future Work

A major limitation of our approach as it currently stands is the lack of support for uncertainty in the world. This uncertainty may be one of two types: it could be (1) uncertainty about the effects of the robot's actions in the world, and (2) uncertainty about whether certain facts hold in the world. We abstract the first kind of uncertainty away by coupling the actions available to the planner with only those scripts that the robotic platform (DIARC) supports, as described in the Architecture section. The second kind of uncertainty, though harder to handle, can actually be beneficial to the synthesis of better quality plans. An immediate line of work for the future is to enhance our system to deal with the probability of additional knowledge or facts about the world holding in a given world state, so that the heuristic may grade the search space to be more conducive to the generation of plans reaching goals whose achievement will garner additional net benefit.

As it stands, the problem of new knowledge about the world is dealt with by sending this information to the planner in the form of state updates. These state updates are picked up by the planning system's execution monitor and incorporated into the planner's database as described in the previous section. However, in a suitably dynamic world, this could potentially lead to the problem of thrashing, sending the planning system into a never-ending loop of update processing and replanning. One possible way to mitigate this problem, while at the same time making use of any probabilistic information that is available in order to produce a heuristic that biases the search space better, is to use anticipatory planning (Hubbe et al. 2008) in order to enable the planner to anticipate the arrival of new information about facts that are relevant to the goals at hand. The availability of distributions that can model the probability of this additional information in the world should significantly improve the performance of such an extension. Additionally, the probabilistic information thus available may be used to implement a hindsight optimization approach similar to the one taken by (Yoon et al. 2008), rather than an optimistic approach (á la FF-Replan).

In the system's current incarnation, we do not handle the problem of mapping the world and localizing the robot within it completely. The planner is able to localize itself within the environment to some degree with the aid of connectivity constraints that are encoded into the domain and problem specification in the form of boolean fluents; thus our system uses an *implicit* map to some degree. The problem of simultaneous localization and mapping (SLAM) (Dissanayake et al. 2001) is not a new one, and has been given extensive coverage in robotics communities. It would be interesting to integrate such approaches with our planner to see if it would improve real world performance while simultaneously decreasing the load on the domain modeler.

A harder limitation to overcome, though, may be the problem of what information from the robot's perception stream the planner should concern itself with. Currently, this problem is handled at the architecture level by specifying an attend list of percepts that may be of interest to the planner. However, a strong case can be made for the planner to use its faculties of reason in deciding which percepts among all those available to it from the world need to be monitored. Such reasoning might involve among other things a process known as *backcasting*, which involves assuming a certain (desirable) future and working backwards to identify a plan that will connect that future to the current state of the world. Identifying the extensions needed to the planner in order to implement such reasoning, as well as recognizing the drawbacks involved, is another promising direction for future research.

Conclusion

In this paper, we presented a novel approach to reconcile a planner's closed world representation with the open world that a robot has to typically execute it. To enable this approach, we presented the integration of techniques that combined are sufficient to represent and solve the scenario described. We showed that we could handle information about new objects in the world using open world quantified goals, and that our replanning and execution monitoring system is able to handle the new information specified by these goals in order to produce plans that achieve a higher net benefit. We also detailed that our system could support soft goals, thus ensuring that opportunities retain their bonus nature, and do not metamorphise into additional hard goals that may constrain existing hard goals. All novel algorithms were implemented and evaluated on a robot. Some lessons from our approach were then presented, along with the limitations of the approach and future work.

Acknowledgements

We thank William Cushing for helpful discussions and the generation of *SapaReplan*. This research is supported in part by the ONR grants N00014-09-1-0017 and N00014-07-1-1049.

References

Benton, J.; Do, M.; and Kambhampati, S. 2009. Anytime heuristic search for partial satisfaction planning. *AIJ* 178(5-6).

Cushing, W., and Kambhampati, S. 2005. Replanning: A new perspective. In *Proceedings of ICAPS*.

Cushing, W.; Benton, J.; and Kambhampati, S. 2008. Replanning as a Deliberative Re-selection of Objectives. *Arizona State University, CSE department*.

Dissanayake, M.; Newman, P.; Clark, S.; Durrant-Whyte, H.; and Csorba, M. 2001. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation* 17(3):229–241.

Do, M., and Kambhampati, S. 2004. Partial satisfaction (over-subscription) planning as heuristic search. *Proceedings of KBCS-04*.

Etzioni, O.; Golden, K.; and Weld, D. S. 1997. Sound and efficient closed-world reasoning for planning. *AIJ* 89(1-2):113–148.

Garland, A., and Lesh, N. 2002. Plan evaluation with incomplete action descriptions. In *Proceedings of the National Conference on Artificial Intelligence*, 461–467. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Golden, K.; Etzioni, O.; and Weld, D. 1994. XII: Planning with universal quantification and incomplete information. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning, KR*, volume 94. Citeseer.

Hubbe, A.; Ruml, W.; Yoon, S.; Benton, J.; and Do, M. 2008. On-line Anticipatory Planning. In *Workshop on a Reality Check for Planning and Scheduling under Uncertainty, ICAPS 2008*.

Kambhampati, S. 2007. Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain theories. *Proceedings of AAAI 2007*.

Penberthy, J., and Weld, D. 1992. UCPOP: A sound, complete, partial order planner for ADL.

Schermerhorn, P.; Benton, J.; Scheutz, M.; Talamadupula, K.; and Kambhampati, S. 2009. Finding and exploiting goal opportunities in real-time during plan execution. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Scheutz, M.; Schermerhorn, P.; Kramer, J.; and Anderson, D. 2007. First steps toward natural human-like HRI. *Autonomous Robots* 22(4):411–423.

Scheutz, M. 2006. ADE - steps towards a distributed development and runtime environment for complex robotic agent architectures. *Applied AI* 20(4-5):275–304.

Smith, D. 2003. The Next Challenges for AI Planning. Lecture given at the Planet International Summer School.

Smith, D. 2004. Choosing objectives in over-subscription planning. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, 393–401.

van den Briel, M.; Sanchez, R.; Do, M.; and Kambhampati, S. 2004. Effective approaches for partial satisfaction (over-subscription) planning. In *Proceedings of the National Conference on Artificial Intelligence*, 562–569. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Weld, D. 1994. An introduction to least commitment planning. *AI magazine* 15(4):27–61.

Yoon, S.; Fern, A.; Givan, R.; and Kambhampati, S. 2008. Probabilistic planning via determinization in hindsight. In *Proceedings of Conference on Artificial Intelligence (AAAI)*.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *17th International Conference on Automated Planning and Scheduling (ICAPS-07)*, 352–359.