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 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.

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 needed to achieve some soft goals may be specified at some later stage *during* the planning process. This partial knowledge may be unbounded, existing in both the problem dynamics and objectives. 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 some 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 and facts in the scenario are those that are mentioned in the initial state. We must also 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 specification of information and creation of objects required to take advantage of opportunities that are encountered during plan execution. Using OWQGs, we can bias the planner's view of the search space towards finding plans that achieve additional reward in an open world.

System Integration

The planner is integrated into the robotic DIARC architecture (Scheutz et al. 2007) where it interacts directly with the *goal manager*. The DIARC goal manager is a goal-based action selection, management, and execution system

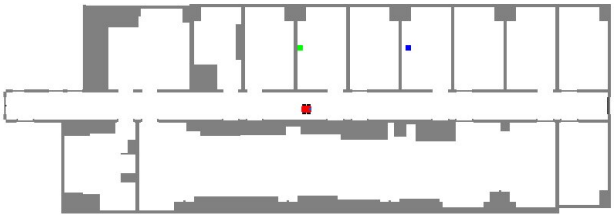

Figure 1: A map of a sample scenario; boxes are stand-ins for humans, where green indicates injured and blue indicates normal.

that allows multiple goals to be pursued concurrently, as long as no resource conflicts arise (e.g., when two goals need the same effector for different actions). Resource conflicts are resolved based on goal priorities with the higher-priority goal obtaining the requested resource. To be able to pursue goals, the goal manager contains "procedural knowledge" in the form of *action scripts* which specify the steps required to achieve a goal. Scripts are constructed of subscripts or action primitives that can be directly carried out by the behavior-based motor control system in DIARC. Since the goal manager has no problem-solving functionality, it cannot achieve goals for which no scripts exist. Hence, the integration of the planning system provides DIARC with basic task problem-solving capabilities of a standard planner in order to synthesize action sequences to achieve goals for which no prior procedural knowledge exists.

On the implementation side, the planner integration into DIARC was accomplished by creating a new planner interface and a new DIARC component implementing this interface as detailed in (Schermerhorn et al. 2009). 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. This can then 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 pre-defined scripts.

Managing sensing: 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 DIARC components to specify which types of percepts (such as doorways in the example above) should be monitored and reported back when detected. Hence, the planner can specify which types of percepts are of interest (i.e., could cause it to update the plan), which effectively focuses attention on those types by causing the goal manager to instantiate monitoring processes that communicate with other DIARC components such as the laser range finder server.

Communicating updates to the planner: 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 have 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.

Planning in an Open World

We apply the planner *SapaReplan* (Cushing, Benton, and Kambhampati 2008) to our planning problem. Like most state-of-the-art planners, it relies on the closed world assumption. There exists an obvious inherent problem with using a planner that assumes a closed world within an open world environment. The planner does not have complete knowledge of all objects (e.g., victims)–because of this, we must consider general, quantified goals while at the same time allowing new object discovery that enables the achievement of the goals. This combination shows the inherent connection between sensing and goal achievement–where goals only exist given particular facts whose truth value remains unknown at the initial state. It further highlights a strict need for sensing in order to ensure high reward for a given plan (in terms of goal achievement). On top of this, we have a set of objects that imply certain facts. That is, a door implies the existence of a room. In this sense, doors imply the potential for goal achievement (i.e., opportunities for reward).

Goals in an Open World

To handle these issues, we introduce a novel goal construct called an *open world quantified goal* (OWQG) that combines 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. This can be seen as a complementary approach to handling open world environments using *local closed world* (LCW) information produced by sensing actions (Etzioni, Golden, and Weld 1997).

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 instance 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. With 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 $\mathcal{O} \in \Pi$, we associate a *closure condition*, $\mathcal{P}$. The value of $\mathcal{P}$ for a given $\langle f, s \rangle$ indicates whether the act of sensing for s has been performed – this knowledge is termed the *sensing closure*. $\mathcal{C} = \bigwedge_i c_i$ 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). 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 (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 associated with 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. In general, of the components that make up an open world quantified goal $\mathcal{Q}$, $\mathcal{P}$ is required 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 expressive, 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 DIARC *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 *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 by Cushing and Kambhampati (2005), 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 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 using a partial satisfaction planning (PSP) representation.

Partial Satisfaction Planning and Replanning: A PSP problem involves actions and (soft) goals with varying costs and rewards. This contrasts with classical planning, which focuses on hard goal achievement. The planning objective is to find plans with high *net benefit* (cumulative goal reward minus plan action cost) by considering which goals should be achieved and which should be ignored due to their high cost or other resource constraints (such as time). The selection process occurs during an A* search. At each search state, the planner's heuristic evaluates the cost for achieving individual goal facts and removes those goals (and supporting actions) that appear too costly to achieve. That is, a goal will not be pursued at a given state if the estimated cost of achievement outweighs the reward.

We can view goal reward as a representation of potential opportunities. The replanning process allows the planner to exploit these as the problem changes over time (i.e., as new updates are sent to the planner). The system aims to handle developments in the problem that remain unknown until execution time, while at the same time providing an ability to exploit opportunities. When a new update arrives, it may enable a path to a potential goal. 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. Hard goals like this can be looked at as commitments that must be achieved.

Implementation

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 towards rewarding goals by generating an optimistic view of the true state of the world.

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` $\leftrightarrow$ `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.

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 was similar to the scenario described in the introduction, with the open world goal being to report victims. Green boxes acted as stand-ins for victims, whereas blue boxes denoted healthy people (whose locations need not be reported). The experimental setup consisted of three rooms, which we represented 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. The respective doorways leading into the three rooms R_1 through R_3 are encountered in order as the robot traverses the hallway.

The aim of these experimental runs was to demonstrate the indispensable nature of each one of the components that make up the 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 total runs of the robot through the scenario. The factors that we varied were: (1) hard goal deadline, (2) presence or absence of action cost, (3) presence or absence of reward on the open world goal, and (4) degree of goal satisfaction on the open world goal (soft or hard). In the table provided, a + symbol stands for the presence of a certain feature, while a - denotes its absence.

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 victims 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 victims 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 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.

When the deadline on reaching the end of the hallway is extended to 200 units, the robot is afforded enough time to

Run	Cost	Reward	Soft	Enter R_1	Report GB	Enter R_2	Report BB	Enter R_3	Hard Goal Deadline
1	+	+	+	Yes	Yes	No	No	No	100
2	+	+	-	Yes	Yes	$\perp$	$\perp$	$\perp$	100
3	+	-	+	No	No	No	No	No	100
4	+	-	-	Yes	Yes	$\perp$	$\perp$	$\perp$	100
5	-	+	+	Yes	Yes	No	No	No	100
6	-	+	-	Yes	Yes	$\perp$	$\perp$	$\perp$	100
7	-	-	+	No	No	No	No	No	100
8	-	-	-	Yes	Yes	$\perp$	$\perp$	$\perp$	100
9	+	+	+	Yes	Yes	Yes	No	Yes	200
10	+	+	-	Yes	Yes	Yes	No	Yes	200
11	+	-	+	No	No	No	No	No	200
12	+	-	-	Yes	Yes	Yes	No	Yes	200
13	-	+	+	Yes	Yes	Yes	No	Yes	200
14	-	+	-	Yes	Yes	Yes	No	Yes	200
15	-	-	+	No	No	No	No	No	200
16	-	-	-	Yes	Yes	Yes	No	Yes	200

Table 1: Results of trial runs. $\perp$ denotes that there is no feasible plan from that point on that fulfills all hard goals.

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 runs, the robot skips all rooms for precisely the same reasons outlined above (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 (non-victims). 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 victims, avoiding the situation encountered in runs 2, 4, 6 and 8 where there is no feasible plan that fulfills all hard goals since the robot has run out of time (denoted $\perp$ in table 1).

These runs thus confirm the indispensable nature of the three main components of the planning system that directs the robot in this USAR scenario – without **replanning**, the system could not 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 fulfillment of new goals. The experimental runs detailed in this section were obtained on a Pioneer P3-AT robot (see figure 2); a video of the robot performing a similar task can be viewed via the following link:
`http://hri.cogs.indiana.edu/videos/USAR.avi`

Figure 2: A Pioneer P3-AT on which the planner integration was verified.

Discussion

Most modern planners are difficult to integrate into real-world systems that must act in scenarios involving unknown facts or objects because they involve the closed world assumption. The issue of planning with an open world versus closed world representation has been dealt with before, notably in the work of Etzioni, Golden, and Weld (1997) via the specification of *local closed world* statements. However, a major difference from that work is that we focus on relaxing our planner's closed world assumption, implementing on top of it a method of admitting possible object existence (using OWQGs). In addition to this, we interleave planning and execution monitoring so new information about the world can be sent to the planner (in the form of state updates). Finally, we provide support for soft goals so that the *opportunistic* nature of new information about the problem is preserved; such goals are not supported by Etzioni et al. due to the fact that these goals were not formally recognized and used by the automated planning community until the early part of this decade.

Other methods involving incomplete initial state information have used conditional planning rather than replanning with execution monitoring for open world scenarios. While performing conditional planning allows for the precomputing of plans, it is also computationally infeasible when the number of contingencies is potentially high. In the USAR scenario under consideration, we would need to contemplate a contingency for every possible door, and sub-

sequently every possible room to search. Even with a limited number of possible doors, this leads to a contingency at each point in the hallway. Instead, we approach the problem by integrating the sensing actions into the architecture, freeing the planner from considering each contingency. The planner finds a new plan whenever the sensing component discovers (and indicates to the planner) information relevant to the domain.

However, just the ability to handle new objects is not enough in this scenario, as we must also consider plan *net benefit*, balancing goal reward and action cost issues. Enabling the specification of soft goals mitigates some difficult problems in handling open world situations. In particular, in the USAR scenario, when certain rooms are completely undiscoverable, it is infeasible to expect complete satisfaction of certain quantified goals. In this respect, we must be allowed to conceptually ignore parts of the problem space in which the robot cannot physically explore. Also, because the scenario involves an explicit deadline (given during plan execution), plans must be allowed to involve passing rooms without searching them to meet the deadline.

With these requirements, it turns out that open world quantified goals can be seen as an approach for solving *conditional goals*. A conditional goal is a goal $A \rightsquigarrow B$ where if the proposition A is true at the initial state, then B must be achieved by a solution plan. Of course, any planner could easily compile conditional goals away with full knowledge of the initial state. However, with partial knowledge a planner must generate plans that sense in order to find whether a goal exists (assuming a single-agent environment). When the conditional goals are soft and assigned a reward, a planner faces the problem of considering both the potential existence of a goal and whether the goal should be achieved. When presented with distributions on goal conditions, we can compute the *expected* net benefit achievable. In this respect, the approach taken for solving OWQGs can be seen as an optimistic determinization of conditional goals, an approach closely connected with the "most-likely outcomes" of the probabilistic planner FF-Replan (Yoon, Fern, and Givan 2007). We are currently exploring the challenges in handling conditional goals, and methods of planning for them.

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 using an integrated system. 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 showed that our system could support soft goals, thus ensuring that opportunities retain their bonus nature, and do not metamorphose into additional hard goals that may constrain existing hard goals. All novel algorithms were implemented and evaluated on a robot. We are currently looking into ways of extending this work to handle open world quantified goals more generally using hindsight optimization and anticipatory planning techniques (Yoon et al. 2008; Hubbe et al. 2008) by looking at them as conditional goals. Methods such as these would likely produce a more robust system capable of better balancing sensing costs with expected reward. We also are considering methods of performing domain analysis to determine what objects should be attended to by the DIARC architecture before plan execution begins.

Acknowledgements

We thank William Cushing for helpful discussions and the development of *SapaReplan*. This research is supported in part by ONR grants N00014-09-1-0017 and N00014-07-1-1049, and the NSF grant IIS-0905672.

References

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. Technical report, Arizona State University, CSE Department.

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 AAAI 2002*, 461–467. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

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*.

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.

van den Briel, M.; Sanchez, R.; Do, M.; and Kambhampati, S. 2004. Effective approaches for partial satisfaction (oversubscription) planning. In *Proceedings of AAAI 2004*, 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 *AAAI*.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-replan: A baseline for probabilistic planning. In *ICAPS*, 352–359.