Goal Reasoning with Informative Expectations

Benjamin Johnson and **Mark Roberts**
NRC Posdoctoral Fellow
Naval Research Laboratory
Washington, DC
firstname.lastname.ctr@nrl.navy.mil

Thomas Apker and **David W. Aha**
Navy Center for Applied Research in AI
Naval Research Laboratory (Code 5514)
Washington, DC
firstname.lastname@nrl.navy.mil

Abstract

In complex and dynamic scenarios, autonomous vehicles often need to intelligently adapt their behavior to unexpected changes in the their environment. Goal Reasoning provides a methodology for autonomous agents to deliberate and adapt their goals to more intelligently react to changing conditions. This paper implements a Goal Reasoning system based on the Goal Lifecycle, and grounds the implementation in the information measures and expectations used by the vehicles to asses their performance. The implemented system, termed Goal Reasoning with Information Measures (GRIM), is demonstrated using a disaster relief scenario in which a small team of vehicles is tasked with surveying a pre-defined set of geographical regions. This demonstration shows how area search goals can be progressively refined, and how they can be adapted to resolve problems encountered by the vehicles during execution.

1 Introduction

Complex applications of robotics often require one or more autonomous vehicles to react intelligently to changes in the operational scenario. A change in the observed state of the environment (e.g., the robot senses an unexpected event) or the internal state of the vehicle (e.g., the vehicle is consuming fuel faster than expected) may necessitate a change in the robot's behavior. This change can manifest as a change to the vehicle's plans, tasks, or even the underlying goals that it is trying to achieve. Intelligent adaptation to unexpected changes is vital to the design of autonomous systems for complex applications.

Goal Reasoning (GR) research aims to develop autonomous agents that can deliberate on, and change, their own goals (Vattam et al. 2013). Such agents would be able to adapt to new and unexpected observations about their environment by creating new goals to pursue. Similarly, they would be able to modify their existing goals to account for unplanned changes by reprioritizing and reordering their goals. Such capabilities become even more valuable in applications where multiple robots must act collaboratively to accomplish their goals; GR would allow the robots to adjust their goals to align with those of the other agents, or to leverage the assistance of other robots.

An example application that would benefit from GR is the control of autonomous vehicles in the Foreign Disaster Relief (FDR) domain (U.S. Department of Defense 2011). The FDR domain focuses on providing humanitarian aid in the wake of natural disasters, and is an area that could greatly benefit from the deployment of autonomous vehicles. In such situations, autonomous vehicles could be used to provide rapid surveys of the disaster area, identifying important locations and traversable routes for the responders. Additionally, the vehicles could be used to enhance the reliability and range of communications, by serving as mobile communication relay points.

To apply GR techniques to applications like FDR operations, it is important for them to be grounded in the information and capabilities that regulate the behavior of the vehicles. That is, deliberation about the goals of an autonomous agent should be performed based on the metrics that define and govern those goals. This paper investigates that by grounding the systems described in Roberts et al. (2015a)[1] and Apker, Johnson, and Humphrey (2016). This grounding frames the goal refinement process in terms of information gathered during the execution of the goal, and is implemented in a system called Goal Reasoning with Information Measures (GRIM). The GRIM system also includes a set of strategies to resolve problems that arise during execution. *The work here presents early steps in creating a full GR system for a team of robots assisting with FDR operations with multiple, possibly conflicting, goals for the system. The paper demonstrates a multi-vehicle system performing GR with respect to a set of area-survey goals.*

The paper is structured as follows. Section 2 describes a motivating example. Section 3 provides a more in-depth description of GR and the instantiation that is

[1] *ActorSim, an implementation of the Goal Lifecycle, is available online at* `http://makro.ink/actorsim/`

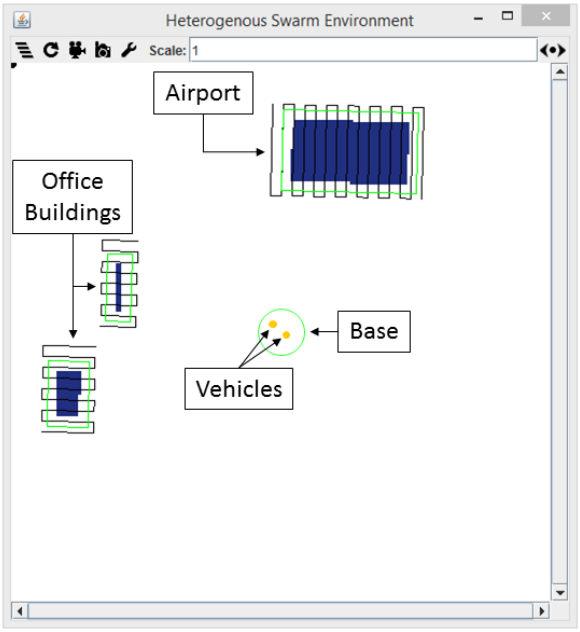


Figure 1: Map of the example scenario. Two vehicles begin in a base region, and are tasked with surveying three different regions of interest: the Airport and the two Office Buildings.

extended in this work. Section 4 demonstrates the GRIM system with respect to the motivating example. Section 5 discusses this work in the scope of other, related work, as well as the future expansion of the system. Finally, Section 6 concludes the paper.

2 Motivating Example

Consider, as a motivating example, an unmanned FDR mission where a team of Unmanned Air Vehicles (UAVs) must survey several pre-defined regions to identify and locate an important official. Figure 1 shows a map of such a scenario, in which a team of UAVs must survey three different regions, each with different characteristics. The Airport is the largest of the regions, composed primarily of flat, open space. Each of the two Office Buildings, on the other hand, are significantly smaller and have considerably more complicated terrain. There is also a Base region, where the UAVs begin the scenario.

The system's first task is to search the regions to locate the official. Once the location of the official is known, the system must then establish and maintain a communications relay for that official. The establishing of a relay is made more difficult in the Office Buildings (in comparison to the Airport) due to the more complicated and cluttered terrain.

These goals are further complicated by other restrictions and factors involved in the mission, such as:

- The need for the UAVs to refuel at the nearby base station.
- Changes to the set of resources (e.g., vehicles) that are available to the system.
- The presence of uncontrolled or adversarial environmental factors (e.g., wind or road blockages).
- Additional goals, with varying or dynamic priorities/importance (e.g., the discovery of a medical emergency that must be immediately addressed).

The system controls a team of two UAVs and can assign them to any of the search areas. Furthermore, due to uncontrolled factors, it is assumed that the vehicles will under-perform their expectations during execution, resulting in slower-than-expected searches of the areas.

3 Goal Reasoning and the Goal Lifecycle

GR focuses on developing agents that can deliberate on and modify their goals during execution within a dynamic environment. The work presented here leverages and adapts the *Goal Lifecycle* of Roberts et al. (2014), Roberts et al. (2015a), and Roberts et al. (2015b), an adaptation of which is shown in Figure 2. This provides a framework for the refinement of goals and the resolution of problems that arise during execution.

The set of goals ("goal nodes" in (Roberts et al. 2015b)) G are stored in a data structure called the Goal Memory. GR is performed by transitioning each goal $g \in G$ through the *modes* (represented by boxes) of the Goal Lifecycle via *strategies* (the arcs). Progression through the modes in the Goal Lifecycle represents increasing refinement in the goal detail. *The step-like structure of the goal modes enforces the concept that each mode builds on the previous mode: each transition strategy can only occur from specific modes in the Lifecycle.*

The remainder of this section briefly summarizes the key strategies of the Goal Lifecycle; specific details are given in Section 4, as the strategies relate to the goals for the motivating scenario described in Section 2. For the remainder of the paper, transition *strategies* are denoted with small caps (e.g., FORMULATE) and the resulting goal *modes* are denoted by monospace small caps (e.g., `FORMULATED`).

The FORMULATE strategy determines when a new goal g is created and enters the Goal Lifecycle *from an external source (e.g., user input, or a triggering event)*. This strategy takes an abstract goal as an input, and transitions it to a `FORMULATED` mode, by defining the initial constraints, the measures defining success or failure, and its prerequisites. The result of FORMULATE is that a new goal is entered into the Goal Memory with the information (i.e., constraints , measures, and prerequisites) required for further refinement.

The SELECT strategy takes a `FORMULATED` goal g, and determines whether the system activates it. A goal

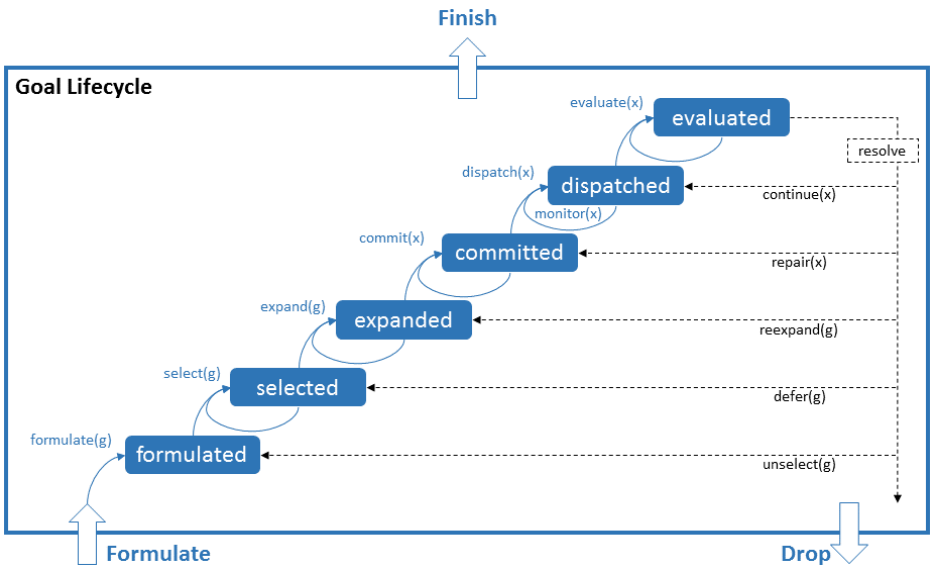


Figure 2: An adapted version of the Goal Lifecycle, from Roberts et al. (2015a). Goals transition through the modes (boxes) via the strategies (arcs), where higher-level modes represent a higher level of goal refinement.

transitions to SELECTED (meaning that it will be actively pursued by the system) only if its prerequisites are satisfied and the system has the available resources to pursue, both of which are defined by the FORMULATE strategy. As such, some goals may not be SELECTED and will instead remain in the FORMULATED mode until their prerequisites are met and the required resources become available.

The EXPAND strategy takes a SELECTED goal g and generates one or more expansions (i.e., plans) $x \in X$ to achieve the goal. An EXPANDED goal defines how the system can satisfy the constraints that were created during the FORMULATE strategy, and generates expectations for how each expansion will perform when executed. If one or more feasible plans are created, the goal transitions to an EXPANDED mode. Otherwise, the EXPAND strategy fails and the goal remains in the SELECTED mode.

Once a goal g has been successfully EXPANDED the COMMIT strategy chooses one of the feasible expansions x for execution. Such a choice involves assessing the costs of each of the expansions, as well as the likelihood that they will successfully execute the goal (per the FORMULATED constraints). A COMMITTED expansion defines how the system will satisfy g, and provides the set of expectations for the performance of the plan's execution.

The DISPATCH strategy sends the COMMITTED expansion x to the executive to run. This process amounts to allocating resources and generating metrics for plan execution. A successfully DISPATCHED expansion defines the criteria by which a goal is evaluated to ensure that it detects and reacts to discrepancies in the expected performance of the expansion.

During execution, two strategies manage updates that impact the mode of the goal g. First, EVALUATE is a passive strategy that is called whenever new information impacts the goal. It can be called by an external process or by the goal itself. In contrast, the MONITOR strategy, when enabled, proactively tracks the execution of the DISPATCHED expansion x to ensure that its expected performance will still result in successful completion of the goal. Additionally, MONITOR ensures that the prerequisite conditions for the goal remain met and that the allocated resources remain available. If MONITOR detects a problem, it triggers EVALUATE directly, and x progresses to an EVALUATED mode, indicating that there is some discrepancy in the performance of the expansion that should be addressed.

When a goal transitions to EVALUATED, the RESOLVE strategy assesses any discrepancies that were detected, and determines how the system should resolve the discrepancy (i.e., which mode the goal should transition to). If the EVALUATED goal g is determined to have met all of the constraints and success-conditions that were generated during the goal formulation, it is resolved with FINISH and marked as completed. If g violates the formulated constraints, DROP marks it as unsuccessful (it can then be reformulated with new constraints). Both FINISH and DROP results in the removal of g from Goal Memory.

Otherwise, if g still meets its formulated constraints but does not meet its success conditions, the RESOLVE strategy transitions the goal to one of the earlier modes

in the Goal Lifecycle. If EVALUATE determines that the DISPATCHED plan x is still feasible, the goal is resolved back to the DISPATCHED mode (referred to as CONTINUE). If the COMMITTED plan can be fixed without major changes by reallocating system resources, it resolves back to the COMMITTED mode (REPAIR). If the committed plan is infeasible, but another feasible plan $\bar{x} \in X$ exists, the goal g is resolved back to the EXPANDED mode and commits to a different, feasible plan $\bar{x}$ (REEXPAND). If no feasible expansion exists given the currently available resources, g is resolved back to the SELECTED mode, where it can be expanded once the necessary resources become available (DEFER). Finally, if the goal no longer meets its prerequisites for selection, but it still satisfies its constraints, it is resolved back to a FORMULATED mode until the prerequisites for selection are met once again (UNSELECT).

4 Goal Refinement with Information Metrics

Goal Refinement for unmanned FDR missions can be framed in terms of refining a set of constraints and expectations for a set of measurable information measures. This section describes a GR system, called Goal Reasoning with Information Measures (GRIM), and demonstrates this system via simulation. The GRIM system instantiates the Goal Lifecycle (discussed in Section 3), and provides centralized control for a small team of 2 UAVs. Returning to the motivating example, described in Section 2, the goal refinement strategies of the Goal Lifecycle are defined here for the area search goals, while the relay goal (which will be the focus of future research) is only included as an abstract goal (i.e., it is not specified below). The information measures for an area search goal, defined in this section, describe the degree to which the defined region has been "searched".

The metric used to evaluate the uncertainty in an area search will differ based on the sensors and algorithms used to conduct the search. For simplicity, the vehicles conduct searches in this example by following a lawnmower waypoint pattern, and the information measure used is the length of that search pattern that has yet to be traversed, though this metric could easily be adjusted to a more accurate measure of uncertainty. This measure was chosen as a simple approximation for the information gathered during the survey task, and future work will explore more accurate measures of the uncertainty in an area survey.

Formulate

The FORMULATE strategy, in the case of the area survey, defines three parameters that describe the constraints under which each can be considered as successful or failed. These parameters are:

1. *maximum uncertainty*: the upper bound on the uncertainty in the search area (i.e., the uncertainty of the area before any information has been gathered),
2. *acceptable uncertainty*: the level of uncertainty at which the goal is considered complete, and
3. *deadline*: the time by which the search must be complete.

The maximum uncertainty specifies amount of uncertainty in the search area prior to any search, and represents the total information that can be gathered about an area during a survey. The acceptable uncertainty parameter defines the level of uncertainty at which the area can be reliably deemed to be empty of an official (i.e., a finishing criteria for the search). Finally, the deadline is set by estimating the time required to arrange follow-on interactions with a located official, as a function of the area's type and terrain. It defines the point in time at which the search must be finished, in order for it to be considered successful. For both (1) and (2), the constraints are defined by the metric used for uncertainty: the length of the search path (in meters) that has yet to be traversed by the vehicles. For (3), the constraint is defined in terms of mission time (seconds).

Figure 3 displays the constraints on each of the formulated goals as a function of the area uncertainty (in meters of untraversed search path) and the execution time. Each of the dashed lines in this figure represents the allowable area of uncertainty for a survey area (i.e., the Airport and the Office Buildings) at a given time. Because of the more complicated interactions of the Office Buildings, the deadline (i.e., the time at which the area uncertainty must be within the defined acceptable level of uncertainty) for each of these is earlier than the deadline for the Airport. At all times up until the deadline, no constraint is placed on the allowable area of uncertainty, so it is set as the full length of the search path for each of the search areas (28,267 meters for the Airport, and 10,303 and 7,537 meters for the smaller Office Buildings). At the deadline, the area of uncertainty is required to not exceed the defined acceptable level of uncertainty. This acceptable level of uncertainty was defined as 500 meters for each of the search areas, but is omitted from Figure 3 for clarity.

The result is a FORMULATED goal $g \in G$, which defines the constraints on the execution of the goal (maximum uncertainty and deadline), as well as the criteria for successful completion of the goal (acceptable uncertainty).

Select

The next strategy in the Goal Lifecycle is for GRIM to SELECT one or more goals to pursue. The SELECT strategy requires comparing high-level estimates of expected performance and value (cost/reward) for each

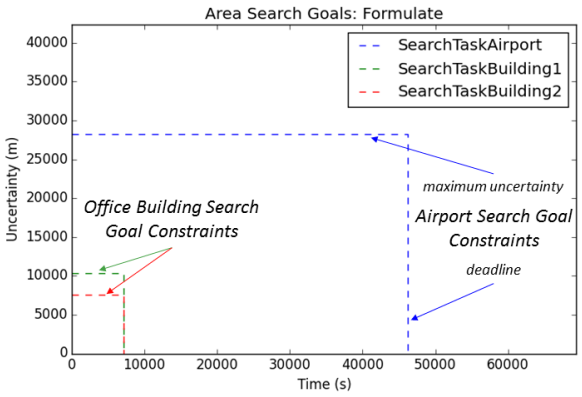


Figure 3: Maximum uncertainty and deadline constraints for the Airport and Office Building search goals. The performance of each goal must remain within the pictured constraints during execution, by reaching the acceptable uncertainty level before the deadline.

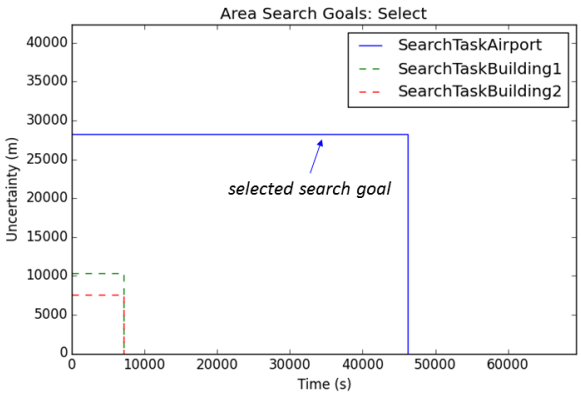


Figure 4: GRIM selects the Airport search goal g, while the Office Building search goals remain unselected. The selected search goal progresses progresses to the EXPAND strategy.

goal, and assessing the available resources for the system. This strategy determines which goals are operational (i.e., those that are selected). For the motivating example presented here, only a single goal is allowed to be SELECTED at a given time.

Figure 4 shows the results of this process, where GRIM elects to pursue the Airport search goal because it is deemed most likely to be successfully searched within the formulated constraints. For this example, the SELECT strategy choses the goal with the smallest ratio of $\frac{max\ uncertainty}{deadline}$. Due to the significantly longer deadline, *the ratio for the Airport is (despite the larger uncertainty) smaller than the ratio for the Office Buildings, and* GRIM selects the Airport goal.

In short, a SELECTED goal g is one that is being pursued by the GRIM system via later strategies in the Goal Lifecycle, while a goal that is *not* SELECTED remains paused in the FORMULATED mode.

Expand

After selecting a particular goal g, EXPAND generates a set of plans[2] X to accomplish it. For each plan $x \in X$ that is generated, a set of expectations is also generated that describe its expected performance with respect to the metrics used in the formulation strategy. A successful EXPAND strategy generates at least one feasible plan (i.e., it is expected to satisfy the formulated constraints). In the example used here, a feasible plan is one where the expected value of the uncertainty at the deadline is no greater than the defined acceptable uncertainty.

[2]The original Goal Lifecycle (Roberts et al. 2015a) used the term *expansion* to refer to the possible plans that could be applied to a goal; the remainder of this paper uses the term *plan* interchangeably with the term *expansion*.

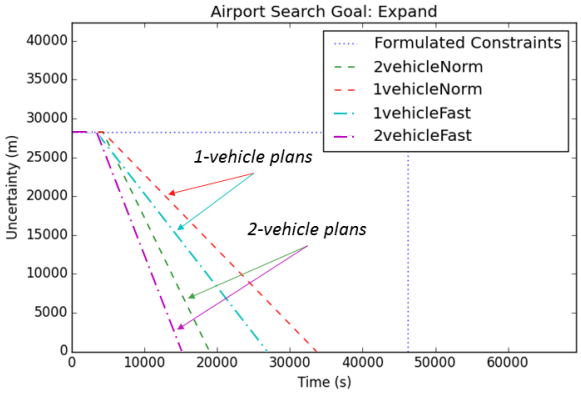


Figure 5: Plots of expected survey performance for each expansion x of the Airport search goal. The "fast" plans increase the vehicle speed to result in a quicker expected completion time, at a higher fuel cost.

Figure 5 displays the expectations of four feasible plans that are generated during the EXPAND strategy, for the selected goal g. The expectations are shown as the expected change in the uncertainty of g (i.e., the length of the search pattern that has been traversed) over time, and each expanded plan uses the same set of waypoints. The resulting plans are for a single vehicle moving at normal speed ("1vehicleNorm"), a single vehicle moving at a faster speed ("1vehicleFast"), two vehicles moving at normal speed ("2vehicleNorm"), and two vehicles moving at a faster speed ("2vehicleFast"). It is assumed that a faster vehicle speed improves the search rate at the cost of higher fuel consumption and that, for each plan, the bulk of the UAVs effort will be expended determining where the official is not located (i.e., a nearly complete search of the area will be required).

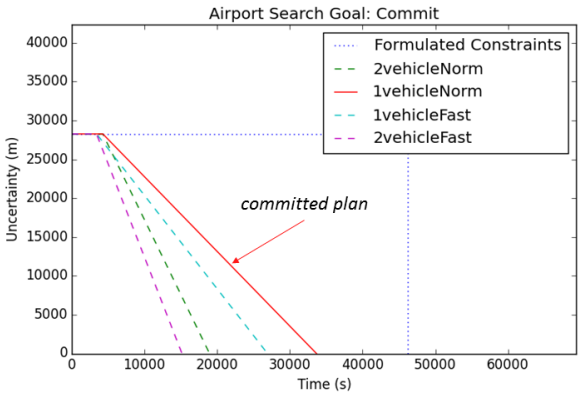


Figure 6: GRIM commits to the 1-vehicle, normal-speed expansion. This plan was selected to conserve resources while still resulting in successful completion.

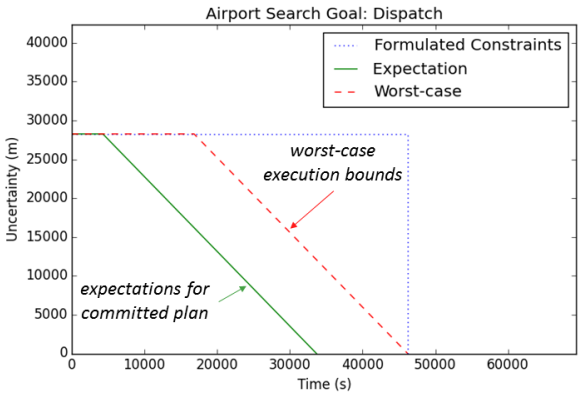


Figure 7: GRIM dispatches the committed expansion x (with expectations) to the applicable vehicle. As part of the plan dispatch, the system generates worst-case bounds on the successful performance of the plan.

The result is an EXPANDED goal g, which has one or more feasible plans $x \in X$, each with a set of performance expectations that satisfy the completion criteria generated by the FORMULATE strategy.

Commit

Once the goal g has been EXPANDED into a set of feasible plans, GRIM must COMMIT to a single plan. To do so, it assesses the costs (i.e., the expended resources, including time) of each feasible plan, and commits to the least costly plan.

Figure 6 highlights the COMMITTED plan ("1vehicleNorm"), which was chosen because the expectations lie well within the formulated constraints while conserving the most resources. By using only a single vehicle, GRIM leaves the second vehicle available in reserve, and by committing to the plan that moves the vehicle at the normal speed the system preserves fuel for other tasks, such as searching another area or providing a relay for a discovered official.

In short, the COMMITTED expansion $x \in X$ is the plan that GRIM chooses to enact in order to pursue the goal g.

Dispatch

Once GRIM has a COMMITTED plan, it must then DISPATCH that plan to the appropriate vehicles. To do this, it must also determine the expected performance bounds for successful plan execution. These bounds represent the worst-case scenario from which the plan can still be expected to satisfy the formulated constraints. To generate these bounds, GRIM uses the expected performance of the plan adjusted such that it is expected to just barely satisfy the constraints; that is, the worst-case bounds represent the execution for which the expectations reach the acceptable level of uncertainty at the deadline. If the performance exceeds these bounds during execution, GRIM will need to adapt the goal g, by applying the RESOLVE strategies described in Section 3. Figure 7 displays the expectations and worst-case execution bounds of the dispatched plan.

In GRIM, a plan is dispatched by scheduling predefined commands for execution by vehicles. When the vehicles receive these commands, they are passed to a synthesized Finite State Automaton (FSA) that is running on the vehicle, and executed according to the rules that were used to synthesize that FSA. A more thorough description can be found in Apker, Johnson, and Humphrey (2016). In the case of the dispatched "1vehicleNorm" plan, the command to search the Airport region is sent to a single vehicle, and the vehicle's speed is left at its default, more fuel-efficient value. The other vehicle is allowed to determine its own behavior (it proceeds to search a different area).

The DISPATCHED plan x is the one that is being enacted by GRIM, and defines both the expected performance of the plan and bounds on when that expected performance will fail to satisfy the constraints and completion criteria of the formulated goal g.

Monitor

During execution, GRIM will actively MONITOR the progress of the DISPATCHED expansion to ensure that it will satisfy the constraints of the selected goal. Figure 8 shows the system's estimate of the search area uncertainty over time, as well as the constraints, expectations, and worst-case bound. In Figure 8a the execution is proceeding slower than expected, but still remains within the worst-case bound; if the execution were to proceed from this point forward at the *expected rate*, it would satisfy the formulated constraints on the goal. Figure 8b shows the execution at a later point in time,

where it first exceeds the worst-case bound; if it were to continue to execute at the expected rate, it would not complete the search within the formulated constraints. As such, MONITOR triggers the EVALUATE strategy.

In summary, the MONITOR strategy actively tracks the execution performance of the dispatched plan x, and triggers the EVALUATE strategy when the performance violates any of the goal's constraints or the plan's worst-case bounds.

Evaluate and Resolve

Once the monitor detects that an execution has violated some constraint or bound, it passes the goal to the EVALUATE strategy. If the execution had satisfied the acceptable level of uncertainty constraint that was generated during formulation, the goal would be passed to the FINISH strategy, where it would be marked as successfully completed. If it violated the other formulated constraints (i.e., it passed the deadline without reaching the acceptable level of uncertainty), the DROP strategy would mark it as failed. In this example, the execution violates the worst-case bound generated during DISPATCH and the goal g is passed to the RESOLVE strategy, where GRIM attempts to change the execution such that it may still satisfy the constraints on g.

The RESOLVE strategy attempts to fix g by working through previous modes in the Goal Lifecycle. In the system described here, the RESOLVE strategy set is limited to the REPAIR, REEXPAND, and UNSELECT strategies.

First, GRIM attempts to REPAIR the expansion by adjusting the expansion chosen in the COMMIT strategy. This involves changing the vehicle speed by committing to a different instance of the 1-vehicle plan from the original expansion: "1vehicleFast". The new instance of the expansion is `COMMITTED`, and it is dispatched and monitored as before. Figure 9 shows the expectation and worst-case bound for the newly repaired plan, which now requires the vehicle to move more quickly and expend more fuel. However, as the execution continues, the repaired plan also fails to meet expectations, and at time 39,975 the system execution crosses the new bound and triggers the EVALUATE strategy, again activating the RESOLVE strategies.

This time, when GRIM attempts to RESOLVE the failing goal, it finds that neither instance of the originally expanded plan can be expected to satisfy the formulated constraints. Thus, it cannot REPAIR the expansion, and it instead attempts to REEXPAND goal g. Doing so allows GRIM to attempt to generate new plans that might be feasible by incorporating resources that were not used by the current expansion. In the example shown here, the second vehicle (which was not used in the originally committed expansion) is available for the newly expanded plans. As a result, the re-expansion strategy finds two new feasible plans (the single-vehicle plans are deemed infeasible): using both vehicles at their default ("2vehicleNorm") and fast ("2vehicleFast") speeds. With these new plans, the goal g returns to the `EXPANDED` mode, and progresses through the Goal Lifecycle again, committing and dispatching the "2vehicleNorm" plan.

Figure 10 shows the expectation and worst-case bounds for the newly re-expanded and `DISPATCHED` plan p, which now assigns both vehicles to search the area. Because the 2nd UAV must first traverse to the search region, it does not arrive in time to assist the search before the deadline, and the new plan also fails to meet expectations. At time 43,670 the execution violates the bounds of the new expansion, and GRIM uses the REPAIR strategy to increase the speed of both vehicles. At time 44,380 MONITOR again triggers the EVALUATE and RESOLVE strategies, but both the REPAIR and REEXPAND strategies fail. GRIM then proceeds to UNSELECT the Airport search goal and consider other goals to pursue. In this case, which was designed specifically to fail (in order to demonstrate the RESOLVE strategies), both of the other search goals have already passed their deadlines, and are dropped as they are deemed to have failed. Once the execution time passes the deadline for the Airport search goal, it will also be dropped.

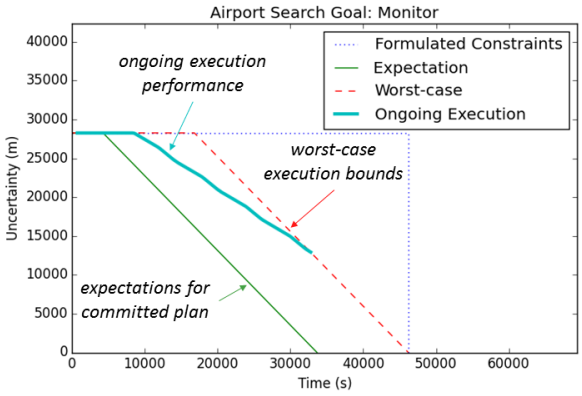
The EVALUATE strategy assesses the performance of the goal and determines which RESOLVE strategy to activate; the RESOLVE strategy will FINISH a completed goal, DROP a failed goal, or REPAIR, REEXPAND, or UNSELECT a goal with an infeasible expansion.

5 Discussion

This paper describes initial efforts towards grounding a GR system, termed GRIM, in the information measures used by the controlled vehicles during execution. In particular, this work adapts the Goal Lifecycle introduced in Roberts et al. (2014; 2015a) and instantiates it in the GRIM system: a centralized GR system that provides commands to independent vehicles. Each vehicle interprets the commands via a play-calling architecture that leverages a formally synthesized FSA and executes the required behaviors via an application of artificial physics, termed *physicomimetics*; more details can be found in Apker, Johnson, and Humphrey (2016) for the play-calling architecture, Kress-Gazit, Fainekos, and Pappas (2009) for the controller synthesis process, and Apker and Martinson (2014) for physicomimetic vehicle control.

Other implementations of GR systems have been developed. Vattam et al. (2013) describes a GR agent as an autonomous agent that is "aware of its own goals and [can] deliberate upon them," and provides a useful survey of related GR research. Autonomous agents that deliberate on their goals are not an isolated concept, and significant research has been conducted towards those ends (Norman and Long 1996; Altmann and Trafton 2002; Cox 2007; Molineaux, Klenk, and Aha 2010; Thangarajah et al. 2010; Harland et al. 2014).

(a) Execution monitor at time 15,000.



(b) Execution monitor at time 32,824.

Figure 8: GRIM monitors the performance of the dispatched plan during execution. Violation of the goal constraints or plan performance bounds will trigger the EVALUATE strategy, causing the system to react to the change in the goal.

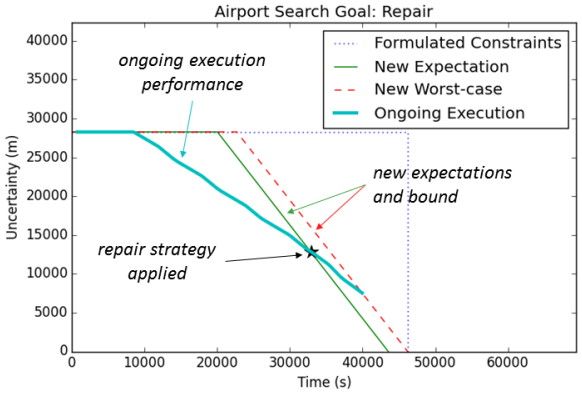


Figure 9: After a violation of the original plan's bounds, the plan is repaired to increase the UAV speed), and new expectations and bounds are generated for the repaired plan. GRIM continues to monitor the execution of the goal until time 39,975.

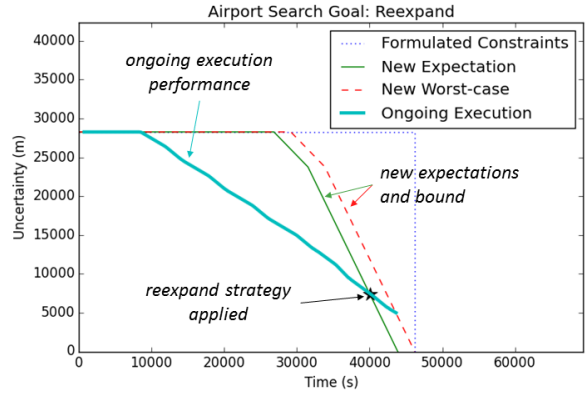


Figure 10: After a violation of the repaired plan's bounds, the goal is reexpanded and the 2-vehicle plan is selected. New expectations and bounds are generated for the reexpanded plan, and GRIM continues to monitor the execution of the goal until time 43,670.

The individual strategies used in the Goal Lifecycle are, themselves, important research topics, and each can be accomplished in a variety of ways. Goal formulation, for example, may occur externally to the system (i.e., a user may provide a goal), or may be conducted autonomously during execution. For example, Klenk, Molineaux, and Aha (2013) present a GR system in which the autonomous agent automatically detects and explains discrepancies during execution, which then facilitates the generation of new goals for the agent. Alternatively, new goals can also be learned or guided by human input through methods such as case-based reasoning (Weber, Mateas, and Jhala 2012; Jaidee, Muoz-Avila, and Aha 2013). As with goal formulation, the specific method for goal selection can vary widely, from domain-specific rule-based selection (Shapiro et al. 2012; Thangarajah et al. 2010) to the evaluation of domain-independent heuristics (Wilson, Molineaux, and Aha 2013), or goal priorities (Young and Hawes 2012).

Similarly, the plan generation strategy can vary among applications or systems, and may involve trajectory generation (Yilmaz et al. 2008; LaValle and Kuffner 2001) or occur at a more abstract level (Blythe 1999; Kress-Gazit, Fainekos, and Pappas 2009). In many cases, plan generation will also generate expectations for the plan's execution performance, though in some cases it may be necessary to generate expectations separately, as in Auslander et al. (2015).

The Goal Lifecycle provides a formal structure for these strategies, such that the resulting system can deliberate on and adapt its goals to dynamic and unpredictable events. This paper extends the Goal Lifecycle within the FDR domain by grounding its implementation using the vehicle's information measures, and by implementing and demonstrating the RESOLVE strategies. This work focused specifically on area survey goals within a disaster relief scenario, though other related goals exist that must also be characterized in a similar fashion. For example, once an official is located by the vehicles conducting the area search it may be necessary to provide a continuous communications relay for that official, which involves formulating a goal of a new type (relay) in GRIM. Likewise, other potential goals (e.g., medical evacuation or logistics supply delivery) may arise during execution of the FDR scenario. Each goal should be defined in the Goal Lifecycle and grounded in the metrics used to evaluate its performance. These are topics for future extensions of the GRIM system.

Future extensions will also investigate the use of more complex algorithms and metrics in the implementation of the Goal Lifecycle. A more accurate measurement of the uncertainty remaining in an area survey will allow GRIM to improve its performance estimates and react accordingly. Additionally, more complex strategies would improve the system's capabilities. For example, a planner or scheduler could be used to SELECT goals *while accounting for the likelihood of discovering an official in each region*, thus enabling GRIM to more intelligently choose which goals to pursue. Likewise, adapting the plan expectations (i.e., recognizing that the vehicles are not completing the survey at the expected rate, and changing the expectations accordingly) would enable GRIM to more quickly identify and evaluate problems, and thus improve the likelihood that it could RESOLVE any discrepancies.

6 Conclusion

This paper demonstrated, via simulation, how a GR system can FORMULATE, SELECT, EXPAND, COMMIT to, and DISPATCH area search goals to a team of autonomous vehicles using the team's information measures and expectations. The system, GRIM, will then MONITOR the performance of these vehicles with respect to their goals, and trigger the EVALUATE strategy when a problem is detected in the execution. When the execution performance violates the pre-determined bounds for the plan, GRIM automatically attempts to RESOLVE the problems by repairing the plan or re-expanding the goal into a new set of plans.

This demonstration showed how a GR system can be useful for autonomous systems operating in dynamic environments, and how to ground it to the information measures used by the system to evaluate its performance. Future work on this subject will extend the system to process additional goal types, and use additional and more complex strategies. This will enable a more thorough evaluation of the benefits of such a system via an experiment with randomly generated scenarios.

References

Altmann, E. M., and Trafton, J. G. 2002. Memory for Goals: An Acctivation-Based Model. *Cognitive Science* 26(1):39–83.

Apker, T., and Martinson, E. 2014. *Distributed Autonomous Robotic Systems: The 11th International Symposium.* Berlin, Heidelberg: Springer Berlin Heidelberg. chapter A Physics Inspired Finite State Machine Controller for Mobile Acoustic Arrays, 47–58.

Apker, T. B.; Johnson, B.; and Humphrey, L. 2016. LTL Templates for Play-Calling Supervisory Control. In *Proceedings of AIAA Science and Technology Exposition*.

Auslander, B.; Floyd, M. W.; Apker, T.; Johnson, B.; Roberts, M.; and Aha, D. W. 2015. Learning to Estimate : A Case-Based Approach to Task Execution Prediction. In *Proceedings of the 23rd International Conference on Case-Based Reasoning*, 15–29.

Blythe, J. 1999. Desicion-Theoretic Planning. *AI Magazine* 20(2):37–54.

Cox, M. T. 2007. Perpetual Self-Aware Cognitive Agents. *AI Magazine* 28(1):32–46.

Harland, J.; Morley, D. N.; Thangarajah, J.; and Yorke-Smith, N. 2014. An Operational Semantics for the Goal Life-Cycle in BDI Agents. *Autonomous Agents and Multi-Agent Systems* 28(4):682–719.

Jaidee, U.; Muoz-Avila, H.; and Aha, D. W. 2013. Case-Based Goal-Driven Coordination of Multiple Learning Agents. In *Proceedings of the 21st International Conference on Case-Based Reasoning*, 164–178.

Klenk, M.; Molineaux, M.; and Aha, D. W. 2013. Goal-Driven Autonomy for Responding to Unexpected Events in Strategy Simulations. *Computational Intelligence* 29(2):187–206.

Kress-Gazit, H.; Fainekos, G. E.; and Pappas, G. J. 2009. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Transactions on Robotics* 25(6):1370–1381.

LaValle, S. M., and Kuffner, J. J. 2001. Rapidly-Exploring Random Trees: Progress and Prospects. In Donald, B.; Lynch, K.; and Rus, D., eds., *Algorithmic and Computational Robotics: New Directions*, 293–308.

Molineaux, M.; Klenk, M.; and Aha, D. W. 2010. Goal-Driven Autonomy in a Navy Strategy Simulation. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. Atlanta, GA: AAAI Press.

Norman, T., and Long, D. 1996. Alarms: An implementation of motivated agency. *Intelligent Agents II Agent Theories, Architectures, and Languages* 219–234.

Roberts, M.; Vattam, S.; Alford, R.; Auslander, B.; Karneeb, J.; Molineaux, M.; Apker, T.; Wilson, M.; McMahon, J.; and Aha, D. W. 2014. Iterative Goal Refinement for Robotics. In *Working Notes of the Planning and Robotics Workshop at ICAPS*. Portsmouth, NH: AAAI.

Roberts, M.; Apker, T.; Johnson, B.; Auslander, B.; Wellman, B.; and Aha, D. W. 2015a. Coordinating Robot Teams for Disaster Relief. In *Proceedings of the 28th International FLAIRS Conference*.

Roberts, M.; Vattam, S.; Alford, R.; Auslander, B.; Apker, T.; Johnson, B.; and Aha, D. W. 2015b. Goal Reasoning to Coordinate Robotic Teams for Disaster Relief. In *Planning and Robotics: Papers from the ICAPS Workshop*. Jerusalem, Israel: AAAI.

Shapiro, S.; Sardina, S.; Thangarajah, J.; Cavedon, L.; and Padgham, L. 2012. Revising Conflicting Intention Sets in BDI Agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 1081–1088.

Thangarajah, J.; Harland, J.; Morley, D.; and Yorke-Smith, N. 2010. Operational Behaviour for Executing, Suspending, and Aborting Goals in BDI Agent Systems. In Omicini, A.; Sardina, S.; and Vasconcelos, W., eds., *Declarative Agent Languages and Technologies VIII: Papers from the AAMAS Workshop*, 1–21.

U.S. Department of Defense. 2011. Department of Defense Support to Foreign Disaster Relief (Handbook for JTF Commanders and Below). *GTA 90-01-030*.

Vattam, S.; Klenk, M.; Molineaux, M.; and Aha, D. W. 2013. Breadth of Approaches to Goal Reasoning : A Research Survey. In Aha, D. W.; Cox, M. T.; and Muñoz-Avila, H., eds., *Goal Reasoning: Papers from the ACS Workshop (Techinical Report CS-TR-5029)*, 111. College Park, MD: University of Maryland, Department of Comnputer Science.

Weber, B. G.; Mateas, M.; and Jhala, A. 2012. Learning from Demonstration for Goal-Driven Autonomy. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 1176–1182.

Wilson, M.; Molineaux, M.; and Aha, D. W. 2013. Domain-Independent Heuristics for Goal Formulation. In *Proceedings of the 26th International FLAIRS Conference*, 160–165.

Yilmaz, N. K.; Evangelinos, C.; Lermusiaux, P. F.; and Patrikalakis, N. M. 2008. Path Planning of Autonomous Underwater Vehicles for Adaptive Sampling Using Mixed integer Linear Programming. *IEEE Journal of Oceanic Engineering* 33(4):522–537.

Young, J., and Hawes, N. 2012. Evolutionary Learning of Goal Priorities in a Real-Time Strategy Game. In *Proceedings of the 8th Annual International Conference on Artificial Intelligence and Interactive Digital Entertainment*, 87–92.