Journal of Experimental & Theoretical Artificial Intelligence

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/teta20

Expectations for agents with goal-driven autonomy

Dustin Dannenhauer , Héctor Muñoz-Avila & Michael T. Cox

To cite this article: Dustin Dannenhauer , Héctor Muñoz-Avila & Michael T. Cox (2020): Expectations for agents with goal-driven autonomy, Journal of Experimental & Theoretical Artificial Intelligence, DOI: 10.1080/0952813X.2020.1789755

To link to this article: https://doi.org/10.1080/0952813X.2020.1789755

Published online: 14 Jul 2020.

JOURNAL OF EXPERIMENTAL & THEORETICAL ARTIFICIAL INTELLIGENCE
https://doi.org/10.1080/0952813X.2020.1789755

Taylor & Francis
Taylor & Francis Group

ARTICLE

Expectations for agents with goal-driven autonomy

Dustin Dannenhauer[a], Héctor Muñoz-Avila[b] and Michael T. Cox[c]

[a]Navatek LLC, Arlington, VA, USA; [b]Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, USA; [c]Wright State Research Institute, Wright State University, Dayton, OH, USA

ABSTRACT

Goal-driven autonomy is an agent model for managing a dynamic environment by reasoning about current and potential goals while planning and acting. Since unexpected events and conditions may cause an agent's goals and plans to become invalid or infeasible, an agent with goal-driven autonomy should monitor the environment against its expectations. Designed for dynamic, open, and partially observable environments, such an agent can create new goals or change its existing goals as needed. We present a formalisation of expectations for agents operating in these kinds of environments. Our formalisation includes situations where agents have the capability to sense the environment with some associated costs. We examine agent choices and behaviour in these domains and evaluate multiple approaches for selecting a subset of the agent's sensing actions to execute. The contributions of this work are (1) a specification of different approaches to generating expectations; (2) a formalisation of the autonomy problem that minimises sensing costs; (3) a complexity analysis of the problem; (4) new algorithms for deciding which sensing actions to perform; and (5) empirical results demonstrating the benefit and cost of these approaches.

ARTICLE HISTORY
Received 13 July 2019
Accepted 18 June 2020

KEYWORDS
Goal-driven autonomy; agent's expectations; goal reasoning

Introduction

Coined in Molineaux et al. (2010), *goal-driven autonomy (GDA)* (Muñoz-Avila, Aha et al., 2010; Weber et al., 2012) is a model of goal reasoning that changes the focus of the agent's attention by dynamically changing its goals based on discrepancies between the agent's expectations and the observations made in the environment. Discrepancies arise when the agent's expectations do not match the agent's observations. This happens when acting in complex environments (i.e., changes occur for reasons other than the agent's individual actions). When discrepancies occur, a GDA agent will generate alternative goals or modify current ones. An example, adapted from Molineaux et al. (2010), involves an agent performing Navy operations. A naval convoy is en route to deliver some equipment and along the way an escort vessel identifies an unknown contact. At this point the agent could pursue one of multiple alternative goals including (1) abort the mission and route the vessels back to the departing port or (2) hold the convoy and send escort vessels to identify the contact.

The defining characteristic of GDA agents is that they adhere to a four-step cycle:

Step 1: ***Discrepancy Detection*** observes the environment for anomalies and when found, generate corresponding discrepancies d.

Step 2: ***Explanation*** generates explanations χ for discrepancies d.

Step 3: ***Goal Formulation*** may generate new goals $\hat{G}$ taking into account the discrepancies d and explanations χ.

CONTACT Dustin Dannenhauer ddannenhauer@navatekllc.com Navatek LLC, Arlington, VA, USA

Step 4: *Goal Selection* selects which goals $g \subseteq \hat{G}$ the agent will pursue next. Goal selection finishes the GDA process until it is triggered again during discrepancy detection.

In this article, we focus on the first step, i.e., discrepancy detection, in GDA agents. Significant previous work has been carried out developing the remaining Steps 2, 3, and 4 (see related work in Section 2). Discrepancies are violations of expectations observed when an agent finds itself in an anomalous or unexpected situation. Discrepancy detection uses expectations to find discrepancies. Expectations are knowledge artefacts that enable agents to check if they are operating as intended. When encountering unknown situations, GDA agents start with what is expected behaviour and use that to derive unexpected behaviour.

In partially observable and dynamic environments, it may not be obvious what an agent needs to know about the current state to achieve its goals and what is irrelevant. This becomes especially true for GDA agents, which may change the goals they pursue over time. The actions an agent takes depend on its goals. In environments that are partially observable, an agent may not be able to directly observe all information in the state. Since some of this information is relevant for deciding which goal to pursue, we consider agents endowed with sensing capabilities. Sensing often comes at a cost, which must be considered when checking expectations. The motivation for assessing sensing costs has been a focus of prior research; researchers have long observed that acquiring knowledge about the state of the world can be expensive in terms of running time to complete the tasks and in resource consumption (e.g., (Knoblock, 1995)). Physical agents may use sensors that require power, time, and potentially other resources (Mei et al., 2005).

The contributions of this article are as follows:

- A formalisation of different approaches to generating expectations for GDA agents (Section 2.2).
- A formalisation of the problem of sensing expectations in GDA (Section 3.1).
- A lower bound formulation for the GDA problem minimising sensing costs, showing it to have a complexity that is PSPACE-hard (Section 3.2).
- New approaches for computing expectations that vary the frequency at which sensing occurs (Section 2.3).
- An empirical evaluation of these new approaches against previous approaches to sensing in partially observable domains (Section 4).

This article is organised as follows. In Section 2 discusses related work. In Section 3 we present our model of autonomy starting with preliminaries that situate this work within the planning and intelligent agent communities. We formalise definitions of expectations and introduce the concept of a frequency with which to check expectations. Section 4 describes an algorithm for computing these expectations. We formulate the problem of GDA under sensing costs, and we give a lower bound complexity analysis for its execution with sensing costs. Section 5 presents experimental evaluations, and Section 6 concludes with a discussion of future work.

Related work

Goal-driven autonomy is inspired in part by work on introspective agents (Cox, 2007; Fox & Leake, 1995; Murdock & Goel, 2008). Cox (2007) in particular introduces notions such as the central roles played by expectations, explanation and goal formulation.

Using the classification scheme of Vattam et al. (2013), Table 1 describes four approaches to discrepancy detection. First, *plan monitoring* is using information from the plan to detect anomalies. Discrepancies can be detected by checking the preconditions and effects before and after actions are executed. Often the agent's planner returns both a plan and corresponding expectations to be checked at each step of the plan. There has also been work done to monitor

Table 1. Different approaches to discrepancy detection.

Discrepancy detection approach	Source of knowledge	Cited works
Plan Monitoring	Planner; preconditions, effects of actions	(Ayan et al., 2007), (Benson & Nilsson, 1993), (Fox et al., 2006a), (Sugandh et al., 2008)
Periodic Monitoring	Whole environment at intervals	(Rao & Georgeff, 1995)
Expectation Monitoring	Agent's specific knowledge base	(Bouguerra et al., 2007), (Veloso et al., 1998), (Kurup et al., 2012), (Cox, 2007)
Domain-based Monitoring	Model of the environment	(Coddington et al., 2005), (Hawes et al., 2011)

the optimality of plans compared to alternative plans; if the current plan is deemed suboptimal or predicted to fail, an alternative plan can be chosen.

Second, *periodic monitoring* is defined as an agent that monitors the environment and uses that as a knowledge source for managing goals. Essentially the agent checks the environment at designated intervals. Periodic monitoring is frequently used in real-time systems. Third, *expectation monitoring* is using knowledge about past experiences to create expectations and using them to monitor parts of the environment as in Veloso et al. (1998). Agents also use models of the environment (can be learned) to identify normal vs. anomalous behaviour. Fourth, *domain-specific monitoring* is monitoring specific variables in the state, and testing those values during plan execution. This approach is similar to *expectation monitoring*, except *expectation monitoring* is generally specific to a plan whereas domain-specific monitoring can be used to trigger new goals to be formed at any time (Coddington et al., 2005). The work we present here is most related to *plan monitoring* and *expectation monitoring* including four specific techniques shown in Table 2 and corresponding related works that use said technique. We define these techniques formally in Section 3.2.

More recent work by Wilson et al. (2014) deals with expectations for agents operating in continuous environments. Their system PHOBOS creates bounded expectations as opposed to precise predictions for continuous state variables (e.g., speed) allowing an autonomous underwater vehicle to avoid false discrepancies when using precise predictions over complex motion. By generating lower and upper bounds, acceptably small variations in continuous state variables do not trigger goal reasoning processes unnecessarily. Their work is similar in the theme of generating expectations that reduce false positives of discrepancy detection; our work is different in that we are concerned more with sensing costs and partially observable environments.

The guiding sensing problem was originally formulated in Dannenhauer et al. (2016). In addition to the input (Σ, s_0, G, c), which we define in Section 4.1, the original guiding sensing problem included one more element: a heuristic function $\phi_g : S \rightarrow \mathcal{A}$ that provides the agent with the next action a to perform when it finds itself in partial state s and pursuing goal g. Since explicitly requiring a control heuristic is unnecessary for the problem definition and complexity analysis we removed it

Table 2. Plan and goal-based discrepancy detection techniques.

Approach	Description	Cited works
Individual Action Expectations	Before executing an action, check that the preconditions of the action hold; After executing an action, check the effects are true in the environment; During action execution check to see if alive conditions hold (optional)	(Sugandh et al., 2008) (Ayan et al., 2007) (Bouguerra et al., 2007)
Informed Expectations	Build up cumulative effects from all previous actions thus far	(Dannenhauer & Muñoz-Avila, 2015; Dannenhauer et al., 2016)
Goal Regressed Expectations	Regress over all preconditions and effects for each step in the plan leading up to the goal	(Fritz & McIlraith, 2007)
State Based Expectations	Use whole states, stored during the planning process, and compare at execution time against the perceived state	(Cox et al., 2012) (Klenk et al., 2013) (Fox et al., 2006a)

from the input to the guiding sensing problem. We feel this is warranted given that similar problem definitions do not require how a solution is generated; for example, in defining the STRIPS planning problem we do not need to make any commitment of how plans are generated. Additionally, in our new definition (see Section 4.1) we add Condition 4 requiring that the effects of the action executed are checked with sensing actions. Condition 4 is needed because the original definition did not make any commitments about when the sensing actions would be executed.

The problem of planning in dynamic environments spawned contingency planning methods (Dearden et al., 2003), in which agents plan for plausible events and conditions that may occur during plan execution. Conformant planning methods (Goldman & Boddy, 1996) generate plans that are guaranteed to succeed given some strong assumptions such as the a priori identification of all possible contingencies. Plan repair methods instead adapt a plan's remaining actions whenever the state conditions required to execute the plan's next action are not satisfied (Fox et al., 2006b). These agents cannot change their goals, whereas GDA agents dynamically reason about which goals they should achieve or modify (see also Cox, 2013; Cox & Dannenhauer 2016).

Deterministic (STRIPS) planning assumes that actions have a predetermined outcome (Fikes & Nilsson, 1971). The result of planning is a sequence of actions that enable the agent to achieve its goals. A Markov Decision Process (MDP) is a frequently studied planning paradigm whereby actions have multiple outcomes (Howard, 1960). In MDPs, solutions are found by iterating over the possible outcomes until a policy is generated which indicates for every state that the agent might encounter, what action to take that will enable the agent to achieve its goals. A Partially Observable Markov Decision Process (POMDP) is an extension of MDP for planning when the states are partially observable (Kaelbling et al., 1998). In POMDPs, solutions are found by iterating over the possible states that the agent believes itself to be in and the possible outcomes of the actions taken on those states until a policy is found. The GDA framework is general allowing a variety of planning paradigms. GDA research has used both planning (Molineaux et al., 2010) and MDP-based planning (Jaidee et al., 2012). Also in regard to POMDPs, the goal-sensing problem doesn't assume that the dynamics of the environment are known by the agent.

The general topic of combining planning and execution has, of course, a long history (Goldman et al., 1996). For example, Sage will aim to plan as far as possible with the known information and perform sensing when needed to advance the plan further (Knoblock, 1995). There is a recurrent interest on planning and execution as exemplified by the recent call for the actor view of planning (Ghallab et al., 2014). Brenner and Nebel coined the term continual planning to refer to the integration of planning, execution and monitoring (Brenner & Nebel, 2006). In their work sensing actions are defined by using variables that are allowed to be uninstantiated. So, for example, the result of a sensing action changes the status of a variable from undefined to a particular constant. An algorithm for asynchronous planning and execution monitoring is presented. Bonet and Geffner (2014) study the problem of contingent planning (i.e., generation of tree plan that accounts for all contingencies that might occur during execution) and conformant planning (i.e., plans that are guaranteed to succeed regardless of the uncertainty in the environment) in belief states (i.e., the collection of all states that are consistent with the current set of observations). Conformant (Goldman & Boddy, 1996; Grastien & Scala, 2020) and (Pryor & Collins, 1996) planning are particularly useful in situations when the probability distributions are not known and hence fall outside of the POMDP framework. Bonet and Geffner (2014) formalism use multi-valued variables and conditional effects to model uncertainty in the environment. Their results show that belief tracking (i.e., planning with belief states) is Turing-complete and propose an approximation algorithm using factored representations.

An alternative to contingent and conformant planning in dynamic environments is replanning (Cashmore et al., 2019; Shani & Brafman, 2011). A plan is generated and when an execution failure is encountered, a new plan is generated from the state where the failure occurred. This has been extended for planning in belief states (Shani & Brafman, 2011). The GDA framework

could adopt any of these planning paradigms for the planning phase. In our work, our planning phase is reminiscent of replanning. The crucial characteristic of GDA is that the agent can change its goals over time.

The representation of goals in partially observable environments that require exploration may need to be different than in fully observable environments. Talamadupula et al. (2017) introduce the concept of Open World Quantifiable Goals used in an urban search and rescue setting (see also Talamadupula et al. (2010)). In this work, human-robot teams search for survivors in damaged buildings, and because the number of survivors is not known ahead of time, they use goal structures that award an agent a higher score for rescuing more survivors while balancing a goal to survey an area in a limited amount of time. For future work, we would like to explore the compatibility of Open World Quantifiable Goals into these kinds of goal reasoning agents.

The goal selection operation can be performed in several ways. For example, in the ICARUS cognitive architecture (Choi & Langley, 2018) goal selection is based on the priority values assigned to the goals, the values assigned are in the range zero to ten, zero signifies that the goal has the least possible priority and ten indicates the highest priority (Choi, 2011). In the MIDCA architecture (Cox et al., 2016), goals are selected using a criterion based on the ratio between the expected benefit of the goal and the expected cost in terms of resources (Kondrakunta, 2017; Kondrakunta & Cox, 2017). In our work, we use a heuristic that chooses the goal we are closest to achieving and do not consider priorities (as the focus of this work is on expectations).

Research in GDA has resulted in techniques to learn GDA knowledge automatically; this includes research to learn goals and goal formulation knowledge (Jaidee et al., 2011) and learn explanations (Weber, 2012). Researchers have also explored applying GDA for playing computer games (Dannenhauer & Muñoz-Avila, 2013; Weber et al., 2010), for conducting naval operations, and for controlling robots (Roberts et al., 2014) among others. Thus far, GDA work has not considered explicit models of the cost of sensing actions; examining the state is assumed to have no cost for the agent. Our work is the first to use GDA with an explicit model of partial observability that accounts for the cost of sensing actions. Furthermore, current GDA research assumes that enough information in the state is observable to plan ahead a sequence of grounded actions to achieve the goals. In our work, we drop this assumption presenting a model that accounts for situations when such planning is not always possible (while at the same time not precluding this possibility).

Goal-driven autonomy (GDA)

Our model of goal-driven autonomy is derived from the formalisms and representations of the automated planning and autonomous agents communities (Ghallab et al., 2004). In this context, autonomy is seen as an agent's capability to independently formulate new goals and to change them when necessary (Cox, 2013). To place our work within this context, we start with some preliminary definitions for agents, goals and behaviour. We then examine the role expectations play in this model.

Preliminaries

A planning model of a given domain is represented by a state transition system $\Sigma = (S, A, \gamma)$, where S is the set of all possible states (a state is defined as a set of grounded atoms) and A is the set of all possible actions. The model of action execution $\gamma : S \times A \rightarrow S$ is a successor function that, given state s_{i-1} and action a_i, returns the subsequent state s_i. A planning problem P is defined as $P = (\Sigma, s_i, g_j)$, s_i the current state and g_j the current goal to be reached. A goal g_j is a set of atoms and is satisfied in the state $s \in S$ if $s \models g_j$. A solution to a planning problem P is a plan π, i.e., a sequence of actions

$\langle a_1, a_2, \ldots a_n \rangle$ such that each action applied sequentially from s_i will result in the state $s_n \models g_j$ (i.e., the state of the world after executing a_n in π).

An action $a \in A$ consists of the usual triple, $(pre(a), a^+, a^-)$ indicating the preconditions, positive effects and negative effects of a. The set $pre(a)$ is the preconditions of a; a is applicable in state s if $s \models pre(a)$. Applying a_i in s_{i-1}, results in a state $\gamma(s_{i-1}, a_i) = (s_{i-1} \backslash a_i^-) \cup a_i^+$.

The goal-driven autonomy model includes mechanisms to create goals, select goals, and change goals (Cox et al., 2017; Cox & Veloso, 1998). The impetus for such reasoning is often a response to anomalous events or behaviour in the environment.

A major underlying motivation for GDA is that, when encountering an anomaly, it may be better for an agent to change its goal than simply to change its plan or to replan; although replanning is subsumed in the GDA framework because a GDA agent may decide to continue with its current goal. A model of our GDA agent is shown in Figure 1.

Here the four-step GDA process is located within the Controller component. Because a GDA agent can plan and execute its plans, it has other components (e.g., perceive, act, and plan). Figure 1 shows multiple components outside of the GDA process. These are often found in GDA agents because they enable planning and execution.

The state transition system Σ is where actions are executed. After an action is executed the agent will perceive the subsequent state. GDA agents start in some initial state s_0 with some initial goal g_0. The planner Π (shown at the top) receives a problem, P, that includes the current state s_i and goal g_j along with the state transition system, Σ. Note that the initial problem will be (Σ, s_0, g_0). The planner then returns a plan π with corresponding expectations X.

As the GDA agent executes actions, the discrepancy detector checks the expectations X against the current state s. If a discrepancy d is found, it is sent to the GDA explanation step. The explanation generator generates an explanation χ, which the goal formulator will use to generate one or more goal(s) $\hat{G}$ to pursue. Then, the goal selector chooses a current goal from among $\hat{G}$. At this point, the current goal g will be sent to Π, and the current plan will be updated with or replaced by any new plan produced by Π. If the discrepancy detector does not detect any anomalies, the GDA processes of explanation generation, goal formulation, and goal selector will not run.

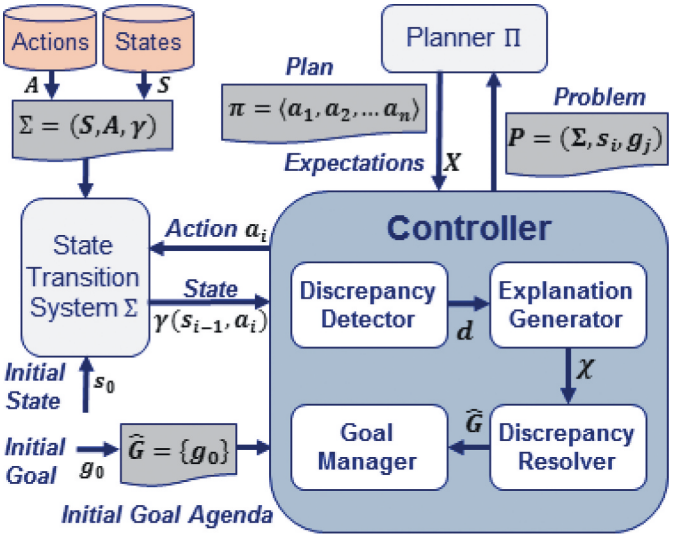


Figure 1. A conceptual model of a goal-driven autonomy agent based on (Molineaux et al., 2010; Muñoz-Avila, Jaidee et al., 2010).

GDA and expectations

We now elaborate on the steps that the agent with goal-driven autonomy repeatedly performs.[1] (1) **Discrepancy detection**: After executing an action a, the agent compares the *observed state* o after sensing and the agent's expectation X (i.e., it tests whether any constraints are violated, corresponding to unexpected observations). In a particular domain, an expectation may be the atom *activated (beacon5)* and a discrepancy may be the observed contradicting atom *deactivated(beacon5)*, for example. If a discrepancy d is found (e.g., $d = x \backslash o$ is not empty), then the agent executes the following step. (2) **Explanation generation**: Given an observed state o and a discrepancy d, this step hypothesises an explanation χ causing d. (3) **Goal formulation**: This step generates a goal $g \subseteq G$ and adds it to the set of existing or pending goals $\hat{G}$ in response to d, χ, and o. (4) **Goal selection**: Given a set of existing/pending goals $\hat{G} \subseteq G$, choose a (possibly) new goal $g \in \hat{G}$ as the current goal.

The GDA cycle is triggered when discrepancies occur. In turn, discrepancies hinge on the notion of an agent's expectations. Expectations are generated from the agent's internal world state: all of the facts an agent has asserted, but may no longer be accurate. In other words, the agent's internal world state is a partial state that may have some, facts invalidated over time, and this internal world state is updated whenever sensing occurs. We explore five kinds of expectations from the literature (Dannenhauer & Muñoz-Avila, 2015; Mitchell et al., 1986; Muñoz-Avila, Aha et al., 2010; Muñoz-Avila, Jaidee et al., 2010), adapted to consider the problem of choosing sensing actions online, which we refer to as the guiding sensing problem. We use the following conventions: $\pi_{prefix} = \langle a_1, \ldots a_n \rangle$ is the sequence of actions executed so far; $\langle s_0, \ldots s_n \rangle$ is the sequence of partial states the agent believes it has visited so far; and $a_{n+1} \in A$ is the next action to be executed.

We need to extend the collection of actions to include sensing actions as well as actions that change the state of the environment. For this purpose, we redefine $A = A_{plan} \cup A_{sense}$, where A_{plan} are typical planning actions (e.g., move the agent to a neighbouring location from its current location), and A_{sense} consists of sensing actions. A sensing action $a_\tau \in A_{sense}$ exists for every condition τ that may be satisfied in the environment. Hence, the application of $\gamma(s, a_\tau)$ returns $\{true, false\}$ depending if τ is a satisfied condition in the environment s (e.g., checks if a specific beacon is activated). In particular, for every effect e of an action, there is a unique condition τ_e for that effect. For example, if e = *activated(beacon55)*, then $\gamma(s, a_{\tau_e})$ is the application of the sensing action to check if beacon 55 is indeed activated. When a sensing action is performed, the current partial state is updated accordingly; the environment itself is not changed.

We now define the five kinds of expectations used in this work:

(1) *No expectations*: The agent performs sensing actions that are only related to the preconditions of a_{n+1} where $a_{n+1} \in A_{plan}$. For every precondition $\tau \in pre(a_{n+1})$, the agent performs a sensing action $a_\tau \in A_{sense}$.
(2) *Immediate expectations*: The agent performs sensing actions for both the preconditions and effects of a_{n+1} where $a_{n+1} \in A_{plan}$.
(3) *Eager expectations*: The agent checks if its cumulative internal state s_n is consistent with the current observed state and if s_{n+1} is consistent with the state observed after executing $a_{n+1} \in A_{plan}$.
(4) *Informed expectations*: $Inf(\pi_{prefix}, s_0)$ move forward all valid conditions computed so far in π_{prefix}. Informed expectations are formally defined as follows: $Inf(\pi_{prefix}, s_0) = Inf(\pi_{prefix}, s_0, \emptyset)$
   - $Inf(\langle\rangle, s, cc) = cc$.
   - $Inf(\langle a \rangle, s, cc) = cc'$ if $a \in A_{plan}$ and is applicable in s, then $cc' = (cc \backslash a^-) \cup a^+$, where a^- are the negative effects from a and a^+ are the positive effects from a.
   - $Inf(\langle a_k, a_{k+1}, \ldots a_n \rangle, s_k, cc) = Inf(\langle a_{k+1}, \ldots a_n \rangle, s_{k+1}, Inf(\langle a_k \rangle, s_k, cc))$.
(5) *Goal-regression expectations*: Given a plan suffix $\pi_{suffix} = \langle a_{k+1} \ldots a_m \rangle$ achieving a set of goals g from some state, goal regression expectations $Regress(\pi_{suffix}, g)$ is formally defined as follows:

- $Regress(\langle\rangle, cc) = cc$.
- $Regress(\langle a\rangle, cc) = (cc \backslash a^{+}) \cup pre(a))$, where $pre(a)$ are the preconditions of a.
- $Regress(\langle a_{k+1} \ldots a_m\rangle, cc) = Regress(\langle a_{k+1} \ldots a_{m-1}\rangle, Regress(\langle a_m\rangle, cc))$.

Checking for no expectations is typical of systems performing deliberative planning, where the agent's actions are not executed in the environment and therefore cannot fail. In a dynamic environment, actions may become invalid and hence an agent using no expectations is prone to fail to achieve its goals. Immediate expectations are an improvement in that the agent checks if the conditions for the next action to be applied hold in the observed state. Immediate expectations are often the default choice by plan execution systems since they verify an action will only be executed when applicable. But if earlier conditions are no longer valid, then the agent will be unaware (e.g., a beacon needed to achieve a goal condition has become deactivated). Hence, immediate expectations are also prone to fail to fulfil its goals.

Eager expectations check that all conditions in the agent's internal world state are satisfied at every iteration. Hence, they are an improvement over the previous two kinds of expectations in that checking for eager expectations guarantees that if these conditions are valid, the plan is still valid (e.g., it will continue checking if a previously activated beacon, needed to achieve a goal, is still active). A drawback is that it may incur high sensing costs; it will also check for conditions that are not relevant for the current plan (e.g., any atom in the state, even if irrelevant to the current goal, will be checked and the agent will incur the corresponding sensing costs). In contrast, goal regression expectations are an improvement in that they guarantee that the expectations computed, $Regress(\pi_{suffix}, g)$, are minimal. That is, if any condition in $Regress(\pi_{suffix}, g)$ is removed then some precondition in the suffix plan π_{suffix} is no longer applicable and therefore g cannot be fulfilled. The main drawback is that some of these conditions might become irrelevant if the agent needs to replan which is prone to occur in dynamic environments. Informed expectations addresses this limitation in that they move forward all conditions validated by the plan, π_{prefix}, executed so far. We state the following property implying advantages and disadvantages of informed expectations over goal regression expectations: Let s_0 be a state, g a set of goals and a plan $\pi = \pi_{prefix} \bullet \pi_{suffix}$ achieving g from s_0 (where $\bullet$ is the concatenation of the two plans). Under these conditions, if $Inf(\pi_{preffix}, s_0)$ is applicable in a state s_k, then $Regress(\pi_{suffix}, g)$ is also applicable in s_k but not the other way around.

This follows from the fact that $Regress(\pi_{suffix}, g)$ computes the minimal conditions and our assumption that $\pi_{prefix} \bullet \pi_{suffix}$ achieves g from a state s_0. This result means that an agent checking for $Inf(\pi_{prefix}, s_0)$ will check for unnecessary conditions assuming that there is no need to replan after executing π_{prefix}. On the other hand, if there is a need to replan to achieve the same goals g, then the informed expectations, $Inf(\pi_{prefix}, s_0)$, will compute the needed conditions regardless of how the plan is completed. In contrast, $Regress(\pi_{suffix}, g)$ conditions might no longer be valid.

Checking expectations periodically

We now introduce periodicity to compute informed expectations. A primary benefit of informed expectations is that it can be used for policy planners in which an agent decides what action to take based on the current state (as opposed to generating a single sequential non-branching plan beforehand). An underlying assumption of informed expectations is that each action the agent takes is relevant to later actions or the agent's goal.

Under partial observability, the agent may believe that it has achieved the goals when in fact this is not the case. For instance, the agent might douse a fire at time earlier but since then the fire has re-started again. Informed expectations guarantee that this will not happen: when the agent believes that the goals have been achieved, they have indeed been achieved. For agents performing other

forms of expectations, this might not be the case. So they explicitly check if the goals are achieved when doing sensing. In particular, now that sensing will be done every few steps, these agents will also check if the goals are achieved.

While prior experiments from Dannenhauer et al. (2016) regarding informed expectations measured goal achievement, in these new approaches presented here, we enable agents to perform additional sensing in situations where the agent incorrectly believes it has achieved its goal (which can happen due to a discrepancy between the true state and the agent's state). This allows agents to verify the conditions of their goal upon believing they have reached their goal. Given new observations, if the goal is not achieved, the agent continues acting. Such agents can vary the frequency at which they perform sensing since they will continue acting following an incorrect assumption that they have reached their goal. The choice of which facts of the state should be verified through sensing remain the same as those computed by the original informed expectations.

Frequency refers to how often sensing should be performed whereas expectations refer to what should be sensed. When the number of expectations to check are numerous (such as informed and eager expectations), it is unlikely most of these expectations will be violated at once. Therefore, sensing may be able to occur less frequently, without significant hindrance on performance. In this work, a frequency $f = 1$ signifies the agent will perform sensing of informed expectations following each plan action. A frequency $f = 2$ signifies the agent will perform sensing of informed expectations every 2 actions, and so forth for $f = 2, 5, 10, 20$. Whenever $f > 1$ holds, for every step that the agent is not performing sensing of the informed expectations, it will still check immediate expectations to ensure each action is executed successfully.[2]

Algorithm 1 shows how an agent with goal-driven autonomy uses these expectations. The algorithm includes a function $X(s, \pi)$, (Line 17), for computing the five kinds of expectations defined in the previous section. We add the following five implementations that determine specific frequencies.

I_2: informed expectations, frequency = every 2 actions
I_5: informed expectations, frequency = every 5 actions
I_{10}: informed expectations, frequency = every 10 actions
I_{20}: informed expectations, frequency = every 20 actions
I_∞: informed expectations only to be checked at time of believed goal achievement

Whenever the system changes to pursue a goal g, if g has not been tried before, the agent will reset the informed expectations. The agent will start accumulating expectations from the first action achieving g. If g has been tried before and a sequence of actions $\pi_g = \langle a_1, a_2, \ldots, a_n \rangle$ had been executed when pursuing goal g, informed expectations will be computed over π_g. For example, suppose the agent begins pursuing some goal g_1 and takes actions $\langle a_1, a_2, a_3 \rangle$ then switches to some other goal g_2 and takes actions $\langle a_4, a_5 \rangle$. The expectations while the agent is pursuing goal g_1 will be computed from $\langle a_1, a_2, a_3 \rangle$ while the expectations for pursuing goal g_2 will be computed from $\langle a_4, a_5 \rangle$. If the agent switches back to g_1 and executes action a_6, then informed expectations are computed over $\langle a_1, a_2, a_3, a_6 \rangle$.

A goal-driven autonomy agent

Algorithm 1 shows the pseudo-code for our agent that is operating in a partially observable and dynamic environment. The main algorithm GDA starts on Line 6. The algorithm uses the following variables, initialised in Line 3: a plan π (initially empty), a sequence of states $\hat{S}$ (initially consisting of the starting partial state s_0), a default goal g, a set of goals that the agent is currently pursuing $\hat{G}$, and a counter t_{check} that indicates the number of plan actions that have been executed since sensing expectations was last performed. It also uses a global variable, G with all potential goals that the agent might pursue. The algorithm also uses an

expectation function, $X(s, \pi)$, as defined in the previous section (excluding goal regression expectations).

Algorithm 1 Goal-Driven Autonomy.

The plan π is a sequence of actions; $\hat{S}$ is a sequence of states; s_0 is the initial state; g_0 is the initial goal; $\hat{G}$ is the goal agenda; t_{check} is a counter; G is the set of all possible goals; ϕ_g is a heuristic function returning an action that will achieve a goal from the given state; and X is a function that takes a state and plan, returning the current expectations

```
1: Global Variables π, Ŝ, s_0, g_0, Ĝ, t_check, G
2: Global Functions φ_g : S → A, X : S × Π → S
3: π ← ⟨⟩; Ŝ ← ⟨s_0⟩; g ← g_0; Ĝ ← {g}; t_check ← 0;
4: GDA()
5:
6: procedure GDA()                                   ▷ Ŝ = ⟨s_0, ... s_n⟩
7:     s ← s_n                                       ▷ Last state in Ŝ
8:     if terminationCondition(Ĝ, s, Ŝ, π) then
9:         return (π, Ŝ)
10:    a ← φ_g(s)                                    ▷ Selects applicable action
11:    (d, s) ← check(pre(a), s, π)
12:    if d = ∅ then
13:        execute(a)
14:        s ← γ(s, a)
15:        π ← π • ⟨a⟩
16:        if (X.frequency mod t_check) = 0 then
17:            (d, s) ← check(X(s, π), s, π)
18:            t_check ← 0
19:        else t_check ← t_check + 1
20:        if d = ∅ then
21:            Ŝ ← Ŝ • ⟨s⟩
22:            return GDA()
23:        else
24:    else
25:    Ŝ ← Ŝ • ⟨s⟩
26:    χ ← explain(d, s)                             ▷ There was a discrepancy
27:    Ĝ ← formulate_new_goals(G, Ĝ, d, χ, s)
28:    g ← goal_selection(Ĝ, s)
29:    return GDA()
```

The algorithm returns the pair $(\pi, \hat{S})$ (Line 9), where π is the trace of all planning and sensing actions executed and $\hat{S}$ is all states the agent believes it visited (depending on the kind of expectations used, the agent may not have checked if every condition in the state is valid in the environment). The procedure begins by taking the last partial state, *s*, believed to be visited (Line 7). It first checks if the termination conditions are met in *s* (Line 8). The *terminationCondition* procedure is detailed in Algorithm 2: Line 14 and is explained later. If the termination condition is not met, then the agent selects an applicable action *a* (based on the agent's internal world state *s*) to execute using the heuristic for the current goal (Line 10). It checks if the preconditions of *a* are valid in the environment by performing sensing actions (Line 11). The procedure *check* is detailed in Algorithm 2: Line 1. We will explain it later, but briefly, it will log in π any sensing action performed and it will modify *s* based on the discrepancies *d* it detected. If the preconditions are satisfied in the environment (Line 12), then the action is executed (Line 13), the internal world state is moved forward by applying action *a* on *s* (Line

14). Action a is added to the end of the plan π (Line 15). Afterwards, the agent performs sensing to check if the expectations are met in the environment (Lines 16–19). Line 16 checks to see how long it has been since sensing occurred. For example, if the expectations function is I_2 then $X.frequency$ is 2 and the resulting behaviour is that every other time Line 16 is reached, control will continue to Line 17, which performs the sensing. Line 18 resets the counter t_{check} because sensing has just occurred. Line 19 simply increments the t_{check} whenever sensing is not performed.

If sensing had been performed and no discrepancies resulted (Line 20) then s is added into $\hat{S}$ and calls GDA() recursively (Lines 21–22). Otherwise, it adds s into $\hat{S}$ (Line 25).

If the preconditions are not satisfied, $\hat{S}$ is simply updated with the current state s (Line 25). If there is a discrepancy either in the preconditions or in the expectations, Line 25 is reached. In either case, $\hat{S}$ is updated with the current state s. Next, the algorithm generates an explanation[3] χ for the discrepancy, formulates new goals[4] $\hat{G}$ to achieve, selects a new goal g to pursue among those in $\hat{G}$ and calls GDA recursively (Lines 26–29).

We now discuss the auxiliary procedures *check* and *terminationCondition* (see Algorithm 2). The *check* procedure (Line 1) receives as parameters the conditions x to be checked, the agent's internal world state s and the actions executed in the environment so far π. It checks for every atom τ in x if τ is currently true in the state (Line 4) while accounting for the fact if it is a positive or negative condition (Line 5). d maintains all discrepancies found (Lines 2 and 6). π is updated with any sensing actions performed (Line 7). The state is updated when there is a discrepancy (Lines 8–11). The procedure returns the discrepancies and the updated state (Line 12). The auxiliary procedure *terminationCondition* (Lines 14–20) checks if the current goals g' (with $g' \subseteq \hat{G}$) are (1) satisfied in the internal world state s (lines 15–16) and (2) satisfied in the environment (Line 17).

Algorithm 2 Subroutines check and terminationCondition

```
1: procedure CHECK(x, s, π)
2:     d ← ⟨⟩; s' ← s                                   ▷ Initialisation
3:     for τ ∈ x do
4:         cond ← γ(s, a_τ)                             ▷ Sensing Action
5:         if (positive(τ) ∧ ¬cond) ∨ (negative(τ) ∧ cond) then   ▷ Discrepancy!
6:             d ← d • ⟨τ⟩
7:             π ← π • ⟨a_τ⟩
8:             if positive(cond) then
9:                 s' ← s'\{τ}                          ▷ τ is not valid in s
10:            if negative(cond) then
11:                s' ← s' ∪ {τ}                        ▷ τ is valid in s
12:    return (d, s')
13:
14: procedure TERMINATIONCONDITION(Ĝ, s, Ŝ, π)
15:    g' ← goalsSatisfied(s, Ĝ)
16:    if g' ≠ ∅ then
17:        if check(g', s, π) then
18:            return true
19:        else return false
20:    else return false
```

Goal-driven autonomy with sensing costs

GDA agents may change their goals over time as a result of discrepancies detected between the agent's expectations and the observed state. When discrepancies are detected, the GDA agents triggers a process that results in the generation of new goals, and in turn, a new plan is generated to achieve our goals. In our formalisation, we will omit a description of the mechanism by which the goals are selected. We simply consider the history of the goals pursued by the agent $\langle g_1 g_2 \ldots g_n \rangle$ over time. This includes the special case when $g_1 = g_2 = \ldots = g_n$; that is, when the goals do not change. For a formalisation of goal formulation, see Cox et al. (2017).

We formalise the problem of Goal-Driven Autonomy with sensing costs as the *guiding sensing problem*. The input to this problem is defined as $P_{sense} = (\Sigma, s_0, G, c)$, composed of the following elements:

Σ, $A = A_{plan} \cup A_{sense}$: As defined in Sections 3.1 and 3.1.
s_0: An initial partial state.
G: A collection of goals; all possible goals that the agent may pursue to achieve.
c: Sensing Cost Function, $c : A_{sense} \rightarrow \mathbb{R}_{\geq 0}$. Returns a non-negative number for each sensing action (c.f., Choudhury et al. (2020)).

A sequence of actions $\pi = \langle a_1 \ldots a_m \rangle$ (each $a_k \in A$) solves the guiding sensing problem $P_{sense} = (\Sigma, s_0, G, c)$ if there is (1) a sequence of partial states $\langle s_0 \ldots s_m \rangle$, and (2) a sequence of goals $\langle g_1 \ldots g_m \rangle$ (each $g_i \neq \emptyset$ and $g_i \subset G$) such that:

(1) If $\pi_{plan} = \langle a_{k1} \ldots a_{kn} \rangle$ denotes the subsequence of all planning actions in π (i.e., each $a_{kj} \in A_{plan}$), then a_{kj} is applicable in state s_{kj-1}; that is, the preconditions of each a_{kj} are valid in the environment at the moment when a_{kj} was executed.
(2) g_m holds in s_m.
(3) If $\pi_{sense} = \langle a_{i1} \ldots a_{iz} \rangle$ denotes the subsequence of all sensing actions in π (i.e., each action in π_{sense} is of the form $\gamma(s, a_\tau)$ for some condition τ, then the total sensing cost $C(\pi) = \sum_{j=1}^{z} c(a_{ij})$ is minimal.
(4) Sensing actions must occur between each pair of contiguous planning actions a_{ki}, a_{ki+1} in π_{plan}, one for each effect e of a_{ki}.

Condition 1 guarantees that the actions taken while the agent was acting in the environment are sound by checking if an action's preconditions are valid before executing the action.

Condition 2 guarantees that at least some of the goals are achieved. This condition is compatible with the special case of oversubscription planning (Smith, 2004), where the agent tries to achieve the maximum number of goals. The sequence of goals $\langle g_1 \ldots g_m \rangle$ is reflective of the goal-reasoning process where agents can change its goals over time and as the circumstances of the environment change. For a formalisation of goal formulation and change, see (Cox et al., 2017).

Condition 3 represents an ideal condition where the agent minimises the cost of sensing while achieving its goals.

Condition 4 requires that each effect of a planning action committed to the plan is checked in the environment before committing to the next planning action.

Modelling action's costs

Although the guiding sensing problem doesn't explicitly consider actions' costs, we are going to show that action costs are subsumed in this formalisation.

Suppose we want to solve the STRIPS planning problem $P = (\Sigma, s_0, g)$ with plan having at most k actions, we can show that P can be transformed into a planning sensing problem $P_{sense} = (\Sigma, s_0, \{g\}, c)$ as follows:

(1) We modify each planning action $a \in A_{plan}$ by adding a unique effect e^a.
(2) For each effect e of each planning action $a \in A_{plan}$ we define a unique sensing action a_{τ_e} (this includes the effects added in Step 1). We define the set of sensing actions as $A_{sense} = \{a_{\tau_e} | e \in a^+ \cup a^-, a \in A_{plan}\}$.
(3) The evaluation of every a_{τ_e} is always satisfied.
(4) We define $A = A_{plan} \cup A_{sense}$
(5) For each effect e in each action $a \in A_{plan}$ that was an effect of the action prior to Step 1, we define $c(a_{\tau_e}) = 0$. For each effect e added in Step 1, we define $c(a_{\tau_e}) = 1$

The following steps are each linear on the number of actions, $|A_{plan}|$, in A_{plan}:

- The modification in Step 1 requires to make one pass through each $a \in A_{plan}$, so it is linear on $|A_{plan}|$.
- The construction of A_{sense} in Step 2 requires one function always returning true for each effect e^a and there are as many such effects as there are actions in $|A_{plan}|$.
- The construction of the cost function c in Step 5 is polynomial on $|A_{plan}|$ assuming the number of effects of each $a \in A_{plan}$ is polynomial on $|A_{plan}|$; if it wasn't, then planning would be exponential or worse since computing the effects of any action with more than polynomial-many effects in an state would require exponential-many steps or worse.

In Step 3 we make each sensing action γ_{τ_e} always satisfied because we are simulating classical planning and, hence, the effects of the actions are always satisfied following the STRIPS assumption (Fikes & Nilsson, 1971).

The additional and unique effect e^a added for each action a in Step 1 and the cost function in Step 7, results in $C(\pi)$ counting the number of planning actions of any plan π solving the guiding sensing problem.

Our formulation guarantees that g will be satisfied since Condition 2 of the guiding sensing problem requires that g_m a nonempty subset of $\{g\}$ is satisfied. Hence, we can consider actions' costs in our framework.

One conclusion from this analysis is that the guiding sensing problem is at least PSPACE-hard since we show how to transform the problem, PLANMIN (Bylander, 1994), of generating a plan with at mist k actions as a guiding sensing problem, PLANSENSE (defined below). Furthermore, we show that this transformation can be done in polynomial time on $|A_{plan}|$, so we proved that PLANMIN $\leq_p$ PLANSENSE. Since PLANMIN is PSPACE-complete (Bylander, 1994), then PLANSENSE must be at least PSPACE-hard. For completeness, we define both of these problems below:

- **Definition**. (PLANMIN) Given $k \geq 1$ and a STRIPS planning problem $P = (\Sigma, s_0, g)$, is there a solution plan π_{plan} for P such that π_{plan} has at most k steps?
- **Definition**. (PLANSENSE) Given $m \geq 1$ and a guiding sensing problem $P_{sense} = (\Sigma, s_0, g, c)$, is there a solution plan π for P_{sense} such that $C(\pi)$?

Experiments and empirical results

We conduct multiple experiments in two simulated environments: Marsworld and Blockscraft, to compare how these different kinds of expectations affect the agents' performance. Both domains are partially observable and dynamic.

Marsworld is a simulated environment that contains randomly located resources for an autonomous agent to use in its pursuit of its goals. Marsworld-like domains have been used in goal-reasoning literature before (see Mudworld from (Molineaux & Aha, 2014) and a slightly different

variation of Marsworld from Dannenhauer and Muñoz-Avila (2015)). Modifications were needed to make the domain partially observable. The high-level task of the agent in this Marsworld domain is to make a signal. A signal can be made by activating enough resources. Here, goals are states that contain a minimum number of activated objects (i.e. beacons are activated by being turned on, wood piles are activated by lighting them on fire, and flares are activated if they are lit). So a goal requiring x resources will be to have any x number of beacons activated, x number of flares lit, or x number of wood pile fires. As the agent explores the environment, flares, beacons, and fires may become deactivated (fires and flares become extinguished by wind, beacons may fail on their own). When fires or flares become extinguished, they are no longer usable; beacons can be re-activated if they were previously deactivated. The agent is endowed with seven plan actions: move up, move down, move right, move left, activate beacon, make fire, drop flare. The agent can sense anything in its current tile and any adjacent tiles (N,S,E,W) with a cost of zero. When an object is no longer within view, the agent can check on the object with a sensing action at a cost of one. Hence, an agent can perform enough sensing to know everything it has seen, but at a high cost (i.e., the cost for each sensing action required to view everything in its internal world state).

The *Blockscraft* domain is a blocksworld-like domain extended from inspiration by the popular sandbox game Minecraft. In this domain the agent's goal is to build a ten-block tower by picking up and stacking blocks. Blocks can only be stacked on the ground or on top of blocks of the same type. The agent does not know what blocks will be available to it over the course of its execution. In the game Minecraft, often the player will dig for blocks and uncover different types of blocks, where the types are only known after becoming visible. The agent always has three blocks in a nearby quarry to choose from, and only after it uses a block will a new one become available. In our experiments there are three different types of blocks and each new block has a randomly selected type. Blockscraft is dynamic due to external agents, unseen to our agent, that may remove blocks from a tower as well as building their own nearby towers. This is akin to the online multiplayer aspect of Minecraft, where many players can modify a shared world. Blockscraft is partially observable in that our agent only has a top-down view of the blocks. The top-down view enables sensing actions of cost zero for the top two blocks of each tower it has built. Any other block (i.e., those under the top two) can be sensed with a cost of one.

Experimental hypotheses

Our experimental hypothesis are the following:

- We hypothesise that *informed* expectations will achieve all goals while having less sensing costs than other expectations.
- We hypothesise that periodic *informed* expectations, as introduced in Section 3.3, will still achieve all goals when checked with some frequency $f > 1$ while reducing sensing costs.

To test these hypotheses we designed two sets of experiments on both the Marsworld and Blockscraft: in the first we directly compared informed expectations versus other forms of expectations (Section 5.2) and in the second one we use periodic sensing (Section 5.3).

Experiment 1: disabled goal sensing

In our first set of experiments, we examine the effect of expectations on goal achievement and sensing costs. We disable the agent's ability to perform sensing when the agent incorrectly believes it has achieved its goal. In each domain, we generated 1000 random scenarios and ran five GDA agents. Four of the GDA agents implemented the four expectations described previously (excluding goal regression expectations, which is not possible in partially observable environments). The fifth GDA agent acts as an upper bound on our experiments to show what the behaviour would be if the

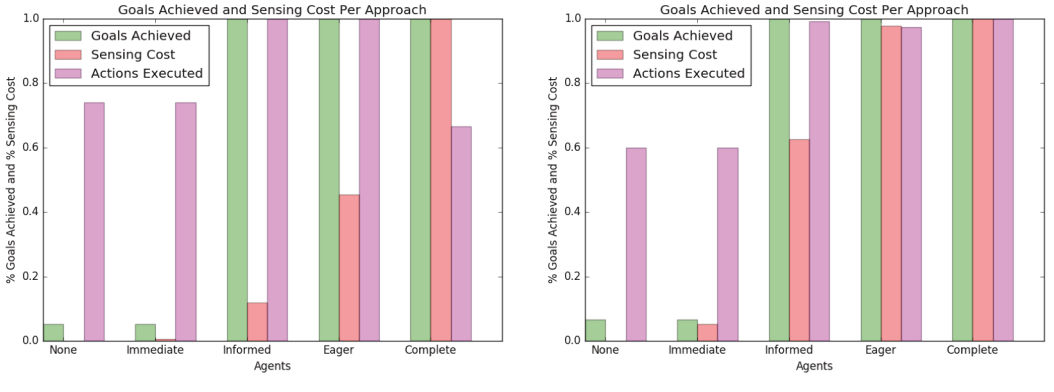
agent could sense the whole environment. In this way, we are able to observe what would happen if the agent can gain full observability of the environment with the same cost model as the other agents (we call this the *complete expectations* agent). Because we are concerned with goal achievement at the time an agent expects a goal to be completed, we do not consider varying sensing (see Section 5.3 for experiments and results that vary the frequency of sensing).

In Figure 2, the first bar of each agent (green) is the percentage of goals achieved. The second bar (red) is the percentage sensing cost out of the maximum sensing cost. The maximum sensing cost is the sensing cost of all atoms in the state (as shown by our upper bound: complete expectations). The third bar (purple) shows the normalised total of actions executed by each agent (the normalisation scale is 0% to 100%). In Figure 2(a) the chance of failure per action executed was 20% for beacons, fires, and flares each. In Figure 2(b), the chance that a block would be removed was 10% and the chance that a block would be added was 30%. Chances for a discrepancy to occur are computed per every planning action executed by the agent.

Examining Figure 2, we see that agents using *none* and *immediate* expectations were unable to achieve most of their goals. Agents using *informed* and *eager* were able to achieve all of their goals. However, *informed* expectations incurred significantly less sensing costs than *eager*, and less than the upper bound shown by *complete*. The *none* and *immediate* expectations agents do not become aware of failures outside their limited view and thus fail to switch goals, reaching their (falsely believed) goal with less actions (compared to *informed* and *eager*). In these experiments, we turned off checking if the goals were satisfied (Line 17 of the Algorithm 1), in order to examine goal achievement.

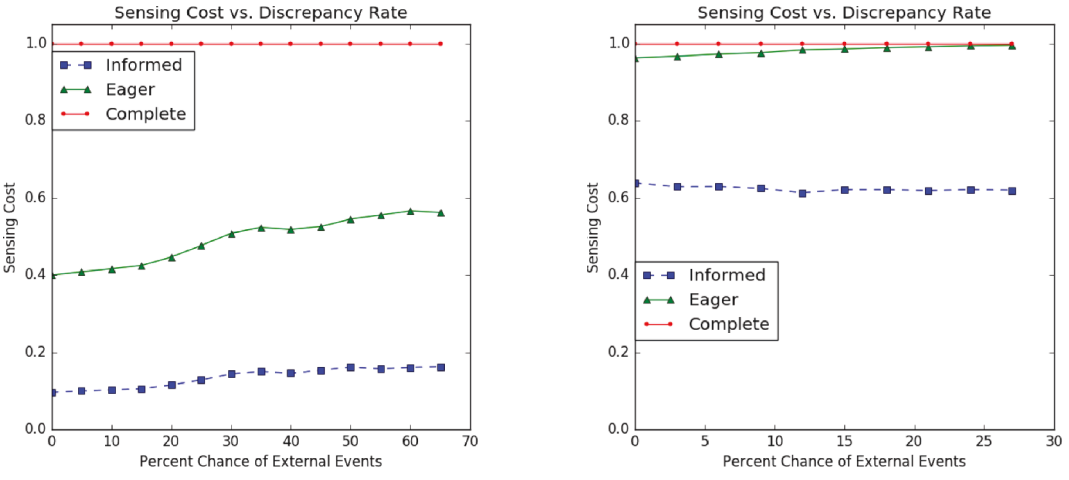
Figure 2(b) shows results from the Blockscraft domain. We see similar results to those in Marsworld. The *none* and *immediate* fail to achieve goals most of the time because even a change in a single block outside the agent's view will go unnoticed without additional sensing. We see a cost gap in the *informed* and *eager* expectations as a result of the *eager* expectations agent sensing everything it has ever seen (including in this case the other towers under construction by other agents), while *informed* only keeps track of those atoms in the state that are related to its previous actions. Thus, it performs sensing only on the blocks the agent itself has stacked.

Figure 3 reports total sensing as a fraction of maximum possible sensing on the y-axis. These values varied by the chance of external events along the abscissa in each domain. Each data point is the total sensing cost performed out of the maximum possible sensing cost over 100 runs. In Figure 3 (a) the chance for each of beacons, fire, and flares to fail varied from 0% to 65% in increments of 5%. Figure 3(b) shows the results of Blockscraft where the chance that blocks were removed varied from 0% to 27% in increments of 3%. Only the chance that blocks were removed was varied; the possibility



(a) Marsworld (b) Blockscraft

Figure 2. Goals achieved and sensing cost per approach per agent without goal sensing.



(a) Marsworld. (b) Blockscraft.

Figure 3. Sensing cost per failure rate.

for blocks being added was held at 30%. We did not test with values close to a 100% failure rate in either domain, because in this range, the environment changes so frequently it is not possible to achieve any goals, even with perfect sensing. By stopping at 65% and 27%, respectively, we are able to see what is happening while still enabling agent's to achieve all of their goals. In both of these figures, we see that informed expectations are performing substantially less overall sensing than eager and complete expectations. In Blockscraft the difference between eager and complete is negligible because both agents basically see the same blocks regardless if used by the agent itself or by the external agents.

In summary, our experiments validate the first hypothesis: informed expectations achieve all the goals; other forms of expectations that also achieve the goals such as eager expectations will do so with much higher sensing costs.

Experiment 2: enabled goal sensing; frequency $>$ 1

In our second set of experiments, we enable goal sensing, allowing the agent to verify its goal is actually achieved and to continue acting if needed. By enabling goal sensing, we are able to evaluate informed expectations with sensing frequencies greater than 1 (i.e. I_2, I_5, I_{10}, I_{20} from Section 4) which may incorrectly believe that a goal has been achieved. We implemented the new informed expectations I_i in the Marsworld and Blockscraft domains. We hypothesise that some of the informed expectations with a higher frequency will outperform informed expectations with a frequency of 1.

For these experiments, we increased the dynamism of both domains. Marsworld has a 35% chance per action executed that a single beacon may become deactivated if it is not already deactivated. Additionally, fires and flares each have an independent chance (also 35%) of failure for a beacon, flare, or fire per action executed providing a high level of dynamism. In Blockscraft, the chance for external events is 25% for adding a block and 25% for removing a block. Each sensing action has a cost of one.

Figure 4 shows the average results of an agent over 1000 randomly generated scenarios. Each agent achieves all of its goals. The x-axis is the type of expectations that agent is using (all agents are identical except for expectations). The y-axis shows the average percentage of maximum possible sensing that was performed. Bars are divided into two colours, red and blue, with red signifying the amount of sensing that was done prior to the agent believing it reached its goal, and with blue

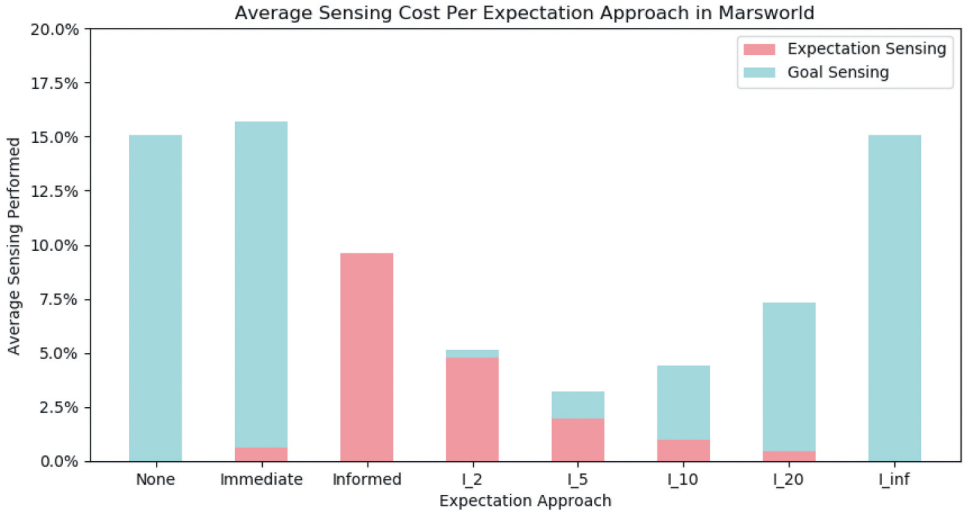


Figure 4. Average sensing costs per approach in the Marsworld domain.

signifying the amount of sensing the agent performed to verify its goal conditions were met. Goal sensing is only performed when the agent believes it reaches its goal but has not actually (hence informed does not require any goal sensing costs). Since we are only concerned with improving upon informed expectations, eager and complete are not shown because they are worse than informed.

The first two bars show the performance of agents using *None* and *Immediate* expectations. Their behaviour is as we expected: no or minimal sensing is done prior to goal achievement, and as a result the agents spend most or all of their sensing costs checking to see their goal is actually achieved. The third bar, which is the original informed expectations guarantees that the goal is always reached when the agent believes it is reached, and this reduces sensing from *Immediate*. We hypothesised that at least some of the new approaches to expectations (I_2, I_5, I_{10}, I_{20}) will reduce sensing and we observe this to be true for I_2, I_5, I_{10}, and I_{20}. I_5 performs best. These results show that varying the frequency of sensing can reduce the overall sensing cost, but the rate of sensing is important.

There is some randomness in how much sensing is required, and Figure 5 shows the running sum of accumulated total sensing performed per goal in the Marsworld Domain. We see that I_5 performs best after 1000 runs, followed by I_2, I_{10}, *Informed*, I_{20}, and eventually *None*, I_{inf}, and *Immediate*.

Table 3 shows the average sensing performed for the 1000 scenarios as a percentage out of maximum possible, along with the standard deviation. It is interesting to note that the standard deviation values for Informed expectations are the smallest of all approaches in both domains (1.89 for Marsworld and 2.42 for Blockscraft).

Figures 6 and 7 show the results for the Blockscraft domain. Figure 6 shows the average sensing results, with the same trend that all new informed expectations with frequencies greater than one (except for I_{inf}) improve over Informed expectations. Additionally, and unlike the Marsworld domain, None and Immediate expectations perform less sensing than Informed (but still not better than I_2, I_5, I_{10}, I_{20}). This result does not appear in Figure 7, where accumulated sensing costs for None and Immediate are higher than Informed. This is due to a few outlier scenarios that require extremely high amounts of sensing for agents using None and Immediate, but for which Informed and its frequency variations are not affected to the same degree. These outliers can be seen in the jumps in Figure 7 where the accumulated sensing cost for *None, Immediate*, and I_{inf} prior to the 200 goals achieved mark. Over many more scenarios, we see that these approaches require significantly more sensing to achieve the same number of goals.

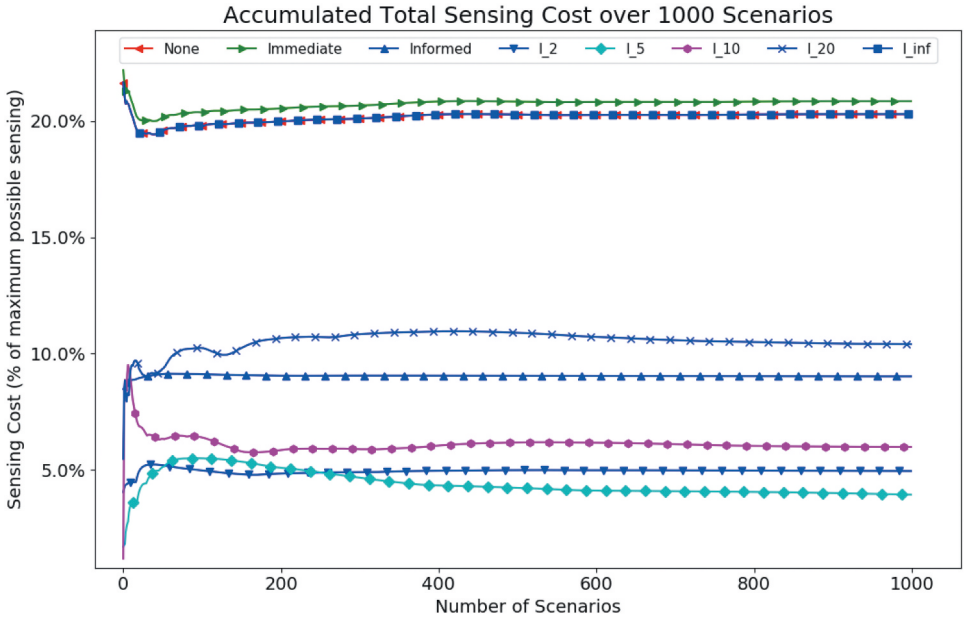


Figure 5. Accumulated sensing cost over 1000 scenarios in the Marsworld domain.

Table 3. Average sensing per domain over 1000 scenarios.

	Marsworld		Blockscraft	
Expectations	Avg	Std	Avg	Std
None	15.05	8.45	13.26	8.81
Immediate	15.68	8.38	16.04	9.39
Informed	9.63	**1.89**	20.38	**2.42**
Informed (F =2)	5.12	2.61	11.50	3.93
Informed (F =5)	**3.22**	4.51	**4.64**	2.59
Informed (F =10)	4.39	6.90	6.13	6.99
Informed (F =20)	7.34	8.22	8.11	8.58
Informed (F =inf)	15.06	8.47	13.31	9.82

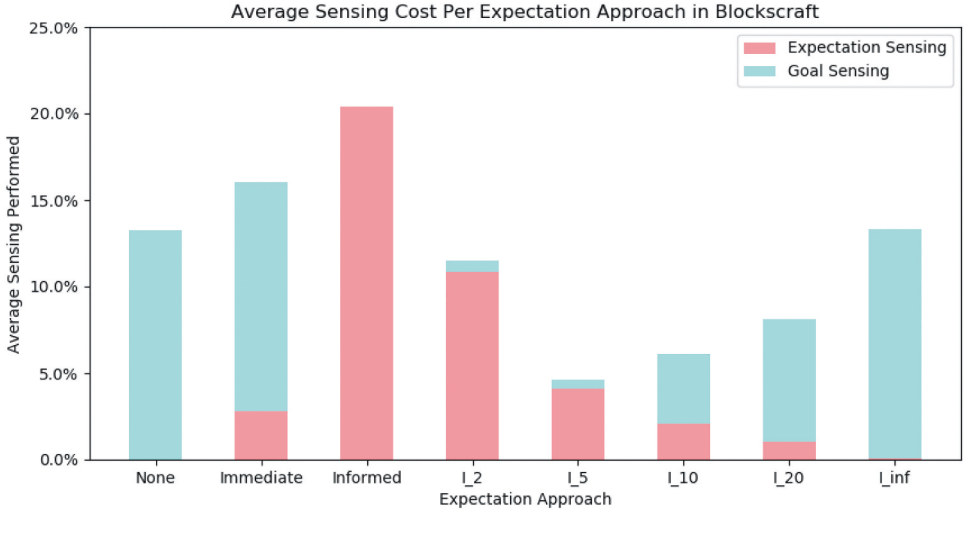


Figure 6. Average sensing costs per approach in the Blockscraft domain.

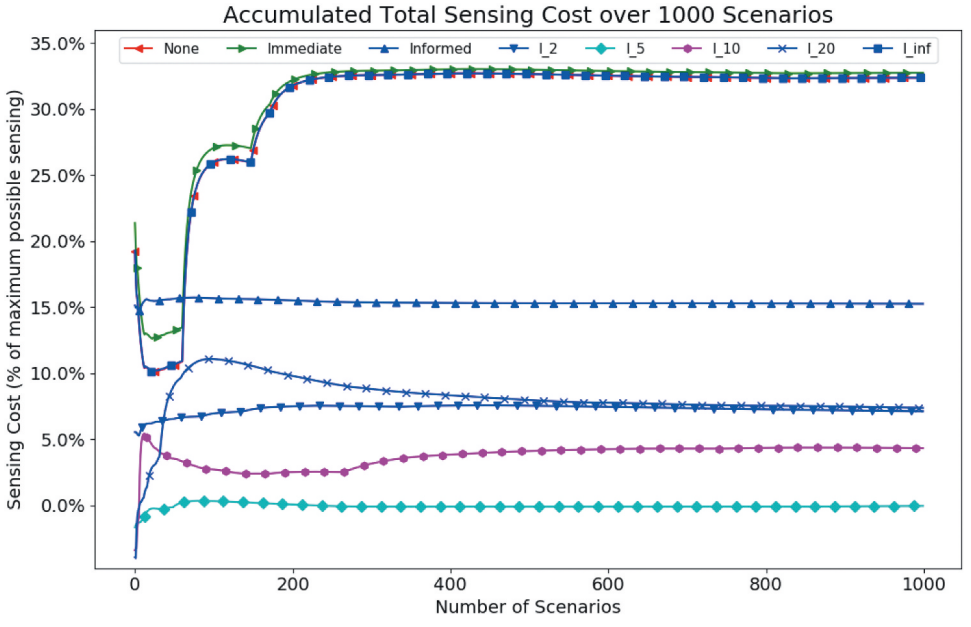


Figure 7. Accumulated sensing costs over 1000 scenarios in the Blockscraft domain.

In summary, these results confirm our hypothesis: with a variety of frequencies $f > 1$ we reduce the sensing costs, while still achieving all goals. With $f = 5$ we see the best results in terms of a reduction in sensing costs for both domains while still achieving all goals.

Conclusion and final remarks

Goal reasoning addresses robustness at a higher level than agents who only re-plan in response to an anomaly. This is warranted in domains where the agents goal may become invalid and attempts at re-planning are futile. We introduced informed expectations, which bridge the gap between immediate expectations and state expectations. Immediate expectations fail to identify discrepancies from past actions, thereby failing to guarantee an agent will identify all relevant changes in the environment leading to goal failure. State expectations (the sequence of states corresponding to the results of each action of the plan) as well as Eager expectations, cover too wide an area for discrepancy detection. In large, complex domains, they falsely trigger discrepancy detection when it is not needed should anything irrelevant change. Informed expectations accumulate the effects of actions as they are propagated through the plan, resulting in only those effects that are necessary in the future states or the goal state. Informed expectations rely on the assumption that the effects of actions are relevant to the rest of the plan and the goal. In situations where one or more action effects may be irrelevant to future plan actions or the goal, goal regression can be used to trim these unnecessary actions, provided future plan actions are known.

We present a formulation of the guiding sensing problem where sensing actions have associated costs for environments that are partially observable. Our formulation is amenable to, both, environments where GDA agents can plan ahead (e.g., as a sequence of grounded actions) and environments where this is not possible. We analyse trade-offs between five forms of expectations (i.e., none, immediate, eager, informed, and goal regression) used by GDA agents when dealing with dynamic environments. We presented an algorithm for a GDA agent operating in both partially observable and dynamic environments approximating a solution to the guiding sensing problem when planning ahead as a sequence of grounded

actions is not possible. We evaluated our algorithm in two simulated environments. From this evaluation, we see that informed expectations perform the best among the four types of expectations (i.e., none, immediate, eager, informed) and using complete expectations (i.e., when full observability is enabled); informed expectations have less sensing costs compared to other expectations that achieve all goals.

Dynamic and partially observable environments present a challenge for agents with sensing capabilities: how to maximise goal achievement while reducing sensing. Optimal sensing can only be known in hindsight, a perfect solution would involve an agent 'magically' knowing what fact outside its view will change and then sense it immediately following that change. The solutions given in this article improve upon previous approaches for guiding sensing and reducing overall sensing cost.

The specific contributions of this article are: (1) a re-formulation of the guiding sensing problem followed by a complexity proof that the guiding sensing problem has a lower-bound complexity of PSPACE-hard in Section 3, (2) new approaches that vary the frequency of sensing, in order to reduce overall sensing while still achieving goals, and (3) empirical results showing the benefit from sensing.

While an optimal solution to minimal sensing is unavailable, it is our opinion that even further improvements can be made from the new approaches described in Section 4. This leaves multiple areas for future work, including:

- The results from Section 5 show that varying the frequency of sensing leads to reducing overall sensing costs. However, finding the best frequency rate is important. An agent with too sparse a frequency rate (compared to how many actions it executes) may incur higher overall sensing because it fails to achieve its goal too many times, leading to more overall sensing (as is the case with $X_1 0$ and $X_2 0$ in Figure 1). A future approach may be to use reinforcement-learning to decide to sense a particular condition of a state with some probability correlated to how much time has passed since the agent last sensed that condition.
- We would like to examine the relationship between the rate of change (dynamism) of the environment and the ideal frequency of sensing. We hypothesise that the more dynamic the environment, the more important that the frequency is smaller.
- We would like to consider a model of sensing costs that is non-uniform, such that some sensing actions have a higher cost than others (i.e. the farther away an object is from the agent, the more expensive it is to sense). Perhaps the cost of a particular action should be considered in deciding whether to perform that sensing action in order to minimise overall sensing cost.
- If replanning is needed only sporadically, a balance between informed and regression expectations might be needed. This is a topic to consider in future work.

Notes

1. For further details, please see an overview of GDA (Klenk et al., 2013).
2. Another way to think about frequency is the percentage of total sensing that occurs per each action executed. With a frequency of 1, 100% of sensing occurs every action, with a frequency of 2, 50% of sensing occurs with every action, with a frequency of 5, 20% of sensing occurs per action, etc. An infinite frequency means 0% sensing occurs per action, and instead sensing only occurs when the goal is believed to be true. The ideal frequency choice in order to optimise total sensing costs depends on the degree to which the environment is dynamic. More dynamic domains warrant higher frequencies for sensing.
3. Explanation is the process of determining causes that lead to the observed discrepancy, we direct the interested ready to prior work on explanation in GDA systems such as (Molineaux et al., 2012).
4. Goal formulation is the process by which new goals are generated dynamically, for examples in GDA systems see (Johnson et al., 2018).

Acknowledgments

This research was supported by the Office of Naval Research under grant N00014-18-1-2009, by the Air Force Office of Scientific Research grant FA2386-17-1-4063, and by the National Science Foundation grants 1909879, 1217888 and

1849131. We also thank the anonymous reviewers for their comments. The views, opinions and findings expressed are those of the authors and should not be interpreted as representing the official views or policies of Navatek LLC, the Department of Defense, or the US. Government.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Air Force Office of Scientific Research [FA2386-17-1-4063]; National Science Foundation [1217888,1849131,1909879]; Office of Naval Research [N00014-13-1-0798,N00014-18-1-2009].

References

Ayan, N., Kuter, U., Yaman, F., & Goldman, R. (2007). HOTRiDE: Hierarchical ordered task replanning in dynamic environments. In *Planning and Plan Execution for Real-World Systems - Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop*.

Benson, S., & Nilsson, N. (1993). Reacting, planning, and learning in an autonomous agent. *Machine Intelligence*, *14*,29-62. https://dl.acm.org/doi/10.5555/242040.242045

Bonet, B., & Geffner, H. (2014). Belief tracking for planning with sensing: Width, complexity and approximations. *Journal of Artificial Intelligence Research*, *50*, 923–970. https://doi.org/10.1613/jair.4475

Bouguerra, A., Karlsson, L., & Saffiotti, A. (2007). Active execution monitoring using planning and semantic knowledge. *ICAPS Workshop on Planning and Plan Execution for Real-World Systems*.

Brenner, M., & Nebel, B. (2006). Continual planning and acting in dynamic multiagent environments. In *Proceedings of the 2006 international symposium on practical cognitive agents and robots* (pp. 15–26).

Bylander, T. (1994). The computational complexity of propositional strips planning. *Artificial Intelligence*, *69*(1–2), 165–204. https://doi.org/10.1016/0004-3702(94)90081-7

Cashmore, M., Coles, A., Cserna, B., Karpas, E., Magazzeni, D., & Ruml, W. (2019). Replanning for situated robots. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling* (Vol. 29, pp. 665–673). AAAI Press.

Choi, D. (2011). Reactive goal management in a cognitive architecture. *Cognitive Systems Research*, *12*(3–4), 293–308. https://doi.org/10.1016/j.cogsys.2010.09.002

Choi, D., & Langley, P. (2018). Evolution of the icarus cognitive architecture. *Cognitive Systems Research*, *48*, 25–38. https://doi.org/10.1016/j.cogsys.2017.05.005

Choudhury, S., Gruver, N., & Kochenderfer, M. J. (2020). Adaptive informative path planning with multimodal sensing. *arXiv preprint arXiv:2003.09746*.

Coddington, A., Fox, M., Gough, J., Long, D., & Serina, I. (2005). Madbot: A motivated and goal directed robot. In *Proceedings of the National Conference on Artificial Intelligence* (Vol. 20).

Cox, M. T. (2007). Perpetual self-aware cognitive agents. *AI Magazine*, *28*(1), 32. https://www.aaai.org/ojs/index.php/aimagazine/article/view/2027/1920

Cox, M. T. (2013). Question-based problem recognition and goal-driven autonomy. In *Goal Reasoning: Papers from the ACS workshop* (pp. 10–25). University of Maryland, Department of Computer Science.

Cox, M. T., Alavi, Z., Dannenhauer, D., Eyorokon, V., Munoz-Avila, H., & Perlis, D. (2016). MIDCA: A metacognitive, integrated dual-cycle architecture for self-regulated autonomy. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (Vol. 5, pp. 3712–3718). AAAI Press.

Cox, M. T., & Dannenhauer, D. (2016). Goal transformation and goal reasoning. In *Proceedings of the 4th workshop on goal reasoning*.

Cox, M. T., Dannenhauer, D., & Kondrakunta, S. (2017). Goal operations for cognitive systems. In *Proceedings of the Thirty-first AAAI Conference on Artificial Intelligence* (pp. 4385–4391). AAAI Press.

Cox, M. T., Oates, T., Paisner, M., & Perlis, D. (2012). Noting anomalies in streams of symbolic predicates using A-distance. *Advances in Cognitive Systems*, *2*, 167–184. http://www.cogsys.org/pdf/paper-3-2-8.pdf

Cox, M. T., & Veloso, M. M. (1998). Goal transformations in continuous planning. In *Proceedings of the 1998 AAAI fall symposium on distributed continual planning* (pp. 23–30).

Dannenhauer, D., & Muñoz-Avila, H. (2013). LUIGi: A goal-driven autonomy agent reasoning with ontologies. In *Advances in Cognitive Systems (ACS-13)*.

Dannenhauer, D., & Muñoz-Avila, H. (2015). Raising expectations in GDA agents acting in dynamic environments. In *International Joint Conference on Artificial Intelligence (IJCAI-15)*.

Dannenhauer, D., Munoz-Avila, H., & Cox, M. T. (2016). Informed expectations to guide GDA agents in partially observable environments. In *International Joint Conference on Artificial Intelligence (IJCAI-16)*.

Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D. E., & Washington, R. (2003). Incremental contingency planning. In *ICAPS-03 Workshop on Planning under Uncertainty*.

Fikes, R. E., & Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*(3–4), 189–208. https://doi.org/10.1016/0004-3702(71)90010-5

Fox, M., Gerevini, A., Long, D., & Serina, I. (2006a). Plan stability: Replanning versus plan repair. *ICAPS*.

Fox, M., Gerevini, A., Long, D., & Serina, I. (2006b). Plan stability: Replanning versus plan repair. In *ICAPS* (Vol. 6, pp. 212–221).

Fox, S., & Leake, D. B. (1995). Learning to refine indexing by introspective reasoning. In *International Conference on Case-based Reasoning* (pp. 431–440).

Fritz, C., & McIlraith, S. A. (2007). Monitoring plan optimality during execution. In *ICAPS* (pp. 144–151).

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated planning: Theory & practice*. Elsevier.

Ghallab, M., Nau, D., & Traverso, P. (2014). The actor's view of automated planning and acting: A position paper. *Artificial Intelligence*, *208*(1), 1–17. https://doi.org/10.1016/j.artint.2013.11.002

Goldman, R., Boddy, M., & Pryor, L. (1996). Planning with observations and knowledge. In *AAAI-97 woRkshop on Theories of Action, Planning and Control*.

Goldman, R., & Boddy, M. S. (1996). Expressive planning and explicit knowledge. In *AIPS* (Vol. 96, pp. 110–117).

Grastien, A., & Scala, E. (2020). CPCES: A planning framework to solve conformant planning problems through a counterexample guided refinement. *Artificial Intelligence Journal*, *284*, 103271. https://doi.org/10.1016/j.artint.2020.103271

Hawes, N., Hanheide, M., Hargreaves, J., Page, B., Zender, H., & Jensfelt, P. (2011). Home alone: Autonomous extension and correction of spatial representations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (pp. 3907–3914).

Howard, R. A. (1960). *Dynamic programming and Markov processes*. Technology Press Research Monographs.

Jaidee, U., Muñoz-Avila, H., & Aha, D. W. (2011). Integrated learning for goal-driven autonomy. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence* (Vol. 3, pp. 2450–2455).

Jaidee, U., Muñoz-Avila, H., & Aha, D. W. (2012). Learning and reusing goal-specific policies for goal-driven autonomy. In Belén Díaz Agudo, Ian Watson (Eds.), *Case-based reasoning research and development* (pp. 182–195). Springer.

Johnson, B., Floyd, M. W., Coman, A., Wilson, M. A., & Aha, D. W. (2018). Goal reasoning and trusted autonomy. In *Foundations of trusted autonomy* (pp. 47–66). Springer.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1–2), 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X

Klenk, M., Molineaux, M., & Aha, D. W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, *29*(2), 187–206. https://doi.org/10.1111/j.1467-8640.2012.00445.x

Knoblock, C. A. (1995). Planning, executing, sensing, and replanning for information gathering. In *In Proceedings of The Fourteenth International Joint Conference On Artificial Intelligence*.

Kondrakunta, S. (2017). *Implementation and evaluation of goal selection in a cognitive architecture* [Unpublished master's thesis]. Wright State University, Computer Science.

Kondrakunta, S., & Cox, M. T. (2017). Autonomous goal selection operations for agent-based architectures. In *Working Notes of the 2017 IJCAI Goal Reasoning Workshop"*.

Kurup, U., Lebiere, C., Stentz, A., & Hebert, M. (2012). Using expectations to drive cognitive behavior. *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Mei, Y., Lu, Y.-H., Hu, Y. C., & Lee, C. G. (2005). A case study of mobile robot's energy consumption and conservation techniques. In *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on* (pp. 492–497).

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, *1*(1), 47–80. https://doi.org/10.1007/BF00116250

Molineaux, M., & Aha, D. W. (2014). Learning unknown event models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Molineaux, M., Klenk, M., & Aha, D. W. (2010). Goal-driven autonomy in a navy strategy simulation. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (pp. 1548–1554).

Molineaux, M., Kuter, U., & Klenk, M. (2012). Discoverhistory: Understanding the past in planning and execution. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems* (Vol. 2, pp. 989–996).

Muñoz-Avila, H., Aha, D. W., Jaidee, U., Klenk, M., & Molineaux, M. (2010). Applying goal-driven autonomy to a team shooter game. In *FLAIRS Conference*.

Muñoz-Avila, H., Jaidee, U., Aha, D. W., & Carter, E. (2010). Goal-driven autonomy with case-based reasoning. In Isabelle Bichindaritz, Stefania Montani (Eds.), *Case-based reasoning. research and development* (pp. 228–241). Springer.

Murdock, J. W., & Goel, A. K. (2008). Meta-case-based reasoning: Self-improvement through self-understanding. *Journal of Experimental & Theoretical Artificial Intelligence*, *20*(1), 1–36. https://doi.org/10.1080/09528130701472416

Pryor, L., & Collins, G. (1996). Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, *4*, 287–339. https://doi.org/10.1613/jair.277

Rao, A. S., & Georgeff, M. P. (1995). *BDI agents: From theory to practice*. AAAI Press.
Roberts, M., Vattam, S., Alford, R., Auslander, B., Karneeb, J., Molineaux, M., … Aha, D. W. (2014). Iterative goal refinement for robotics. In *ICAPS Workshop on Planning and Robotics*.
Shani, G., & Brafman, R. I. (2011). Replanning in domains with partial information and sensing actions. In *International Joint Conference on Artificial Intelligence* (Vol. 2011, pp. 2021–2026).
Smith, D. E. (2004). Choosing objectives in over-subscription planning. In *ICAPS* (Vol. 4, p. 393). AAAI Press.
Sugandh, N., Ontañón, S., & Ram, A. (2008). On-line case-based plan adaptation for real-time strategy games. *AAAI*.
Talamadupula, K., Benton, J., Kambhampati, S., Schermerhorn, P., & Scheutz, M. (2010). Planning for human-robot teaming in open worlds. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *1*(2), 14. https://dl.acm.org/doi/10.1145/1869397.1869403
Talamadupula, K., Briggs, G., Scheutz, M., & Kambhampati, S. (2017). Architectural mechanisms for handling human instructions for open-world mixed-initiative team tasks and goals. *Advances in Cognitive Systems*, *5*, 37–56. http://www.cogsys.org/papers/ACSvol5/paper-5-6.pdf
Vattam, S., Klenk, M., Molineaux, M., & Aha, D. W. (2013). *Breadth of approaches to goal reasoning: A research survey (Tech. Rep.)*. Naval Research Laboratory.
Veloso, M., Pollack, M., & Cox, M. (1998). Rationale-based monitoring for planning in dynamic environments. *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, 171–179.
Weber, B. (2012). *Integrating learning in a multi-scale agent* [Unpublished doctoral dissertation]. University of California.
Weber, B. G., Mateas, M., & Jhala, A. (2010). Applying goal-driven autonomy to StarCraft. In *AIIDE*.
Weber, B. G., Mateas, M., & Jhala, A. (2012). Learning from demonstration for goal-driven autonomy. In *AAAI*.
Wilson, M. A., McMahon, J., & Aha, D. W. (2014). Bounded expectations for discrepancy detection in goal-driven autonomy. In *AI and Robotics: Papers from the AAAI Workshop*.