Proceedings of the Twelfth Annual Conference on Advances in Cognitive Systems ACS-2025 (241-258)

Accumulated Expectations Improve Intelligent Agent Adaptation in Dynamic Environments

Paola Rizzo P.RIZZO@INTERAGENS.COM
Interagens s.r.l., Rome, Italy

Héctor Muñoz Avila HEM4@LEHIGH.EDU
Lehigh University, Bethlehem, PA, USA

David W. Aha DAVID.W.AHA.CIV@US.NAVY.MIL
Navy Center for Applied Research in AI; Naval Research Laboratory, Washington DC, USA

Abstract

Intelligent agents that adapt to dynamic environments need to (1) verify that the actual effects of their actions are consistent with their expected effects (generated during planning) and (2) change their execution schedules, actions, plans or goals if discrepancies arise among the expected and actual effects. In this paper we introduce *accumulated expectations*, which are a new type of expectations that maintain a history of the assumed effects of an agent's executed actions to enhance its adaptation abilities. We formally define accumulated expectations, contrast them with other forms of expectations that have previously been studied, and test them in dynamic simulation environments using both a single agent task domain (maritime search and rescue) and a multiagent task domain (aerial survey). In our evaluations, we found that agents that use accumulated expectations significantly outperform agents that do not use them. More specifically, for the maritime search and rescue domain, using accumulated expectations reduces the negative impact of a dynamic element on performance, and for the aerial survey domain, it improves performance through positive interactions among the agents' behaviors.

1. Introduction

To successfully achieve goals and execute plans in dynamic environments, intelligent agents need to represent the expected effects of their actions (generated during planning) and constantly compare them with the actual effects when actions are executed. By doing so, agents can change their execution schedules, actions, plans or goals when they detect discrepancies among the expected and actual effects that either (1) disrupt the execution or schedule of the original plan or the achievement of the initial goal (negative discrepancies), or (2) favor the execution of a better plan or the achievement of a preferable goal (positive discrepancies).

An intelligent agent can compare the expected and actual effects of its actions by examining the most recent action that it executed, and possibly by also considering the facts that its previous actions made true before such execution. For instance, an autonomous robot needs to ensure that its pick-up action has indeed produced the effect of holding a screwdriver, and that the object on

which to use the screwdriver is still on the workbench where the robot placed it earlier. Another example is an autonomous car needing to verify that its right turn has actually produced the effect of starting to cross an intersection, and that the turning signal it switched on before the turn is still blinking. These verifications assume that all the agent needs to know is embedded in the effects of each single action, and that there is no need to store a history of past effects of one or more actions. For instance, the robot does not need to remember which objects it picked before the screwdriver, and the vehicle can forget about the past episodes in which it switched on the signal.

However, in some situations the ability to remember an action's history of expected effects can permit agents to be more adaptive to the environment. For instance, an autonomous boat that pursues a maritime search and rescue (S&R) mission by surveying the coasts of islands after a tsunami to find and take survivors to a safe base, may inadvertently lose some people at sea (e.g., when a strong wave hits the boat and lets a survivor fall in water). In such cases, when dropping survivors at the safe base, the boat would realize that someone is missing, and would need to remember the path it traversed so as to traverse it again backwards to find and collect the fallen people.

Alternatively, in a multiagent setting where communication might not be available (e.g., due to faults or insufficient power, areas without signal, missing communication protocol, privacy in an adversarial domain), an autonomous drone A that surveys an area assuming that it is the only agent doing so needs to remember which subareas it traversed: this would enable A to (1) recognize when another drone B is surveying the same area because A's perception of the area's traversed portions does not correspond to the portions it remembers having traversed, and (2) replan a more efficient survey path by avoiding the parts already surveyed by B.

These examples concern action effects that are computed but "forgotten" (i.e., are overwritten in the state) during planning and, consequently, also during execution. In fact, while planning, the boat and drone in the above examples do not need to store the sequence of effects of each of their movements, as every instance of the latter produces an effect that is used as a precondition of the next movement. This guarantees that each movement produces the desired trajectory to achieve some goal, and requires keeping track of only the agent's most recent position. However, not recording the history of expected effects can limit the agent's ability to adapt to the environment as illustrated in the previous examples. While humans can use their working memory (Baddeley & Hitch, 1974) to store the sequence of expected effects of their plans, planning systems do not have this cognitive component. Here we show how to endow a planning system with a similar functionality so as to improve the agent's adaptability to its environment.

We define *accumulated expectations* (AEs) as the history of effects occurring while planning and executing actions. These effects include the intermediate literals that can be overwritten in the state by the planner and not retained during plan execution), but that need to be remembered at execution time to achieve, as we will show, a more robust and opportunistic performance. AEs are useful when (1) the planner uses action effects to update literals concerning the same object (e.g., the agent's position, the location it has explored, or the item the agent holds or uses), and when (2) there is the need to check one or both of the following issues: (2.1) the order in which objects are processed (e.g., while putting items on a conveyor belt in repair or logistics domain operations, or while preparing multimedia contents by placing components on a timeline), or (2.2) which specific

objects have been the focus of actions (e.g., which beacons have been lit by a patrolling agent, or which items have been collected by an autonomous character in a video game like Minecraft).

Detecting and addressing discrepancies in AEs is more general and efficient than adding domain-dependent specifications to accommodate predefined situations. For instance, in the aerial survey example above, agent *A* might be designed to survey a different area whenever it senses agent *B* in the area it is currently surveying (i.e., it assumes that *B* is surveying the same area, even though *B* might only be navigating elsewhere). Detecting a discrepancy in the AEs would instead trigger the replanning of *A*'s survey path without making assumptions about *B*'s goal, and without necessarily changing the area that *A* surveys. Also, AEs support explainability: an agent that remembers its actions' history of assumed effects can use them to explain to a human supervisor why it decided to change its plan or goal when those accumulated effects were violated.

In this paper, we describe AEs and examine their benefits for plan execution and replanning. Section 2 describes related work regarding other types of expectations, and contrasts them with AEs. In Section 3 we formalize the concept of AEs. Section 4 presents our empirical study of AEs in two simulated domains: the first involves a single agent and uses negative discrepancies, while the second includes multiple agents and positive discrepancies. In Section 5 we indicate directions for future work.

2. Related work

Our work is broadly placed in the context of plan execution, monitoring, and repair. Several successful techniques have been devised that allow agents to respond appropriately to unexpected circumstances that occur during plan execution, including contingent planning (Hoffmann & Brafman, 2005), conformant planning (Hoffmann & Brafman, 2006), probabilistic planning (Kushmerick et al., 1995), online replanning or plan repair (Fox et al., 2006), continual planning (Brenner & Nebel, 2006), and case-based planning (Hammond, 1990). Our work concerns online replanning, which requires execution monitoring to detect any possible discrepancies between the expected and actual effects of actions. Execution monitoring coupled with online planning has been studied for decades (e.g., Fikes et al. (1972), Wilkins (1988)), but still has unaddressed questions that require further investigation, as shown in this paper. In both human and artificial cognition, plan monitoring is also related to meta-reasoning, in that the latter concerns processes that monitor the progress of reasoning and problem-solving activities and regulate the time and effort devoted to them (e.g., Ackerman & Thompson (2017), Cox et al. (2022)). Our work is also related to goal reasoning, which is the process intelligent agents can use to continually reason about the goals they are pursuing to change or adapt them (Aha, 2018).

The monitoring of the expected effects of executed actions can be done in several ways: Munoz-Avila et al. (2019) presented a taxonomy of expectations we follow here to highlight the novelty and benefits of AEs, focusing on the types of expectations that are relevant to the topic of this paper. *Immediate expectations* (Cox, 2007) check that the effects of the latest executed action are valid in the current state, and that the preconditions of the next action to execute are valid as well. They are helpful when one can assume that all the agent needs to know at any time during execution is represented in the most recent pair of consecutive actions in the plan, but such assumption may not

hold, as described in Section 1. *State expectations* (Klenk et al., 2013) project forward the initial state and apply to it the modifications produced by executing each action of the plan: hence, the agent can detect many more discrepancies than with immediate expectations, because the discrepancies concern the most recently executed part of the plan rather than just the most recent pair of consecutive actions. However, state expectations can detect many false positives and therefore trigger unnecessary replanning, since they propagate the whole state and therefore cannot distinguish between discrepancies that are relevant to the plan and those that are not. *Informed expectations* (Dannenhauer et al., 2016) are an advance over state expectations in that they detect discrepancies concerning the trajectory of the executed plan portion without relying on the whole state: in fact, instead of projecting forward the complete state, they propagate only the changes to the state that are produced by the executed actions. However, informed expectations do not maintain the effects of actions that were deleted by subsequent actions, as would be needed in the examples discussed so far. Maintaining the history of effects of actions, and using it to improve the agent's adaptability to the environment, is exactly the problem that AEs aim to solve, as further detailed in Section 3, and evidenced in Section 4, which describes our empirical studies.

In human cognition, accumulating the expected effects of actions that are computed during planning would be done by the central executive of working memory (Baddeley & Hitch, 1974), which is included in many cognitive architectures (Kotseruba & Tsotsos, 2020). For instance, the working memory of a car driver who plans a path from A to B would have to remember a sequence of intermediate locations that the driver expects to traverse along the path (in this example, the locations would be stored in the visuospatial sketchpad). If one of the locations is blocked (e.g., due to roadwork), then the driver's working memory would find a discrepancy between the expected location and the location imposed by the roadwork, forget the remaining sequence of intermediate locations, and compute a new sequence based on an alternative path. The working memory could then store the alternative path and the initial regular path to traverse them in reverse when moving from B to A, so as to exploit the successful overall path and avoid recomputing it. This example shows that adding AEs to planning systems enhances them through a mechanism inspired by human cognition.

Another aspect of human cognition seemingly related to AEs is episodic memory (Tulving et al., 1972), which can store events lived by a person, together with contextual information such as time, place, or emotional state. This memory is also included in many cognitive architectures (e.g., Laird (2019), Menager & Choi (2016), Martin et al. (2022)), where it has been shown to support several important cognitive capabilities such as modeling actions and the environment, managing long-term goals, or predicting success/failure (e.g., Nuxoll & Laird (2012), Derbinsky et al. (2012)). However, AEs are not equivalent to elements stored in the episodic memory because they are generated during planning (i.e., while mentally simulating possible future states to produce preconditions and expected effects) rather than during execution (i.e., while experiencing the states produced by executing the planned actions). Also, AEs do not need to be stored together with contextual information. For instance, the car driver of the previous example needs to remember only the locations to traverse along the planned path from A to B and, while following the path, does not need to store contextual information. Even when the driver encounters a blocked location, there is no need to remember when the block was encountered or what was observed at the block, but only to compute

an alternative path and remember the expected locations along the new path. When driving back from B to A on the reversed path, the driver does not need to recall the experiences lived along the path. In other words, remembering AEs is more similar to remembering a sequence of potentially arbitrary numbers disconnected from context than to remembering a sequence of related events together with their contextual information. State expectations, which maintain information about the entire state expected to be experienced during execution, are more similar to elements of episodic memory.

3. Formalization

A **ground atom** is a logical expression of the form *name*$(c_1, \ldots, c_x)$, where *name* denotes a relation and each c_i denotes a constant. A **state** s is a list of ground atoms (i.e., the facts of the world). We define an **action** a as a triple *a =(name pre eff)*, where *name* is a ground atom (i.e., the name of the action), *pre* is a list of ground atoms (i.e., the action's preconditions), and *eff* (i.e., the action's effects) is also a list of ground atoms but with each atom having a prefix $+$ or $-$.

We denote by *add(a)*, the collection of all effects in action a with prefix $+$ and *del(a)* the collection of all effects in a with prefix $-$. Applying an action a in s, written as $a(s)$, is defined as follows:

$$a(s) = \begin{cases} \emptyset & \text{if } \textit{pre} \not\subseteq s \text{ (i.e., } a \text{ is not applicable in } s) \\ (s \backslash \textit{del(a)}) \cup \textit{add(a)} & \text{otherwise (where } \backslash \text{ is the set difference)} \end{cases}$$

A **plan** $\pi = (a_1, ..., a_n)$ is a list of actions. A plan π is **applicable** in a state s_0 if: (i) $s_1 \neq \emptyset$, where $s_1 = a_1(s_0)$ and (ii) $s_i \neq \emptyset$, where $s_i = a_i(s_{i-1})$ for $2 \leq i \leq n$. We denote by $\pi_i = (a_1, \ldots a_i)$ a plan prefix of π ($1 \leq i \leq n$).

For the sake of completeness and following the taxonomy in Dannenhauer et al. (2016), we define immediate and state expectations. Whenever a discrepancy in the expectations is detected, it triggers a replanning process. The taxonomy classifies the different conditions that trigger such a discrepancy.

Immediate expectations. Let π be applicable in state s_0. We then define the immediate expectations for π_i and s_0, as: $X_{\text{imm}}(\pi_i, s_0) = eff(a_n)$. That is, immediate expectations are the effects of the last action in π. We use them to identify instant reasons for replanning; namely, if any of the action's effects are not valid in the state after executing the action, then a discrepancy is detected.

State expectations. Let π be applicable in state s_0. We then define the state expectation for π_i and s_0, as: $X_{\text{state}}(\pi_i, s_0) = \pi_i(s_0) = s_i$. If the expected state, s_i is not identical to the actual state observed by the agent, a discrepancy is detected.

State expectations are not included in our empirical evaluation because they result in poor performance, forcing the agent to replan whenever there is a mismatched condition in the state, even if the mismatch is unrelated to the plan (Munoz-Avila et al., 2019).

Informed expectations were originally introduced in Dannenhauer et al. (2016). Here we use the simplified definition found in Munoz-Avila et al. (2019):

Informed expectations. Let π be applicable in state s_0. Then the informed expectation for π_i and s_0, $X_{\text{inf}}(\pi_i, s_0)$ is defined recursively as follows:

$$X_{\text{inf}}(\pi_i, s_0) = \begin{cases} add(a_1) & \text{if } i = 1 \\ (X_{\text{inf}}(\pi_{i-1}, s_0) \backslash \textit{del}(a_i)) \cup \textit{add}(a_i) & \text{if } 2 \leq i \leq n \end{cases}$$

Informed expectations identify the atoms in the state that are added by effects of actions in π_i. Hence, informed expectations are a subset of the state expectations. More precisely: $X_{\text{inf}}(\pi_i, s_0) \subseteq X_{\text{state}}(\pi_i, s_0)$. Informed expectations are preferable over state expectations as they allow an agent to monitor only the atoms that are expected to occur in a state as a result of executing the plan's actions, rather than all the atoms in the state. Therefore, discrepancies are triggered only when there is a mismatch on the monitored atoms.

Accumulated expectations. We now formalize the notion of AEs, $X_{\text{acc}}(\pi_i, s_0)$, introduced in this paper.

$$X_{\text{acc}}(\pi_i, s_0) = \begin{cases} add(a_1) & \text{if } i = 1 \\ X_{\text{acc}}(\pi_{i-1}, s_0) \cup \textit{add}(a_i) & \text{if } 2 \leq i \leq n \end{cases}$$

AEs track the history of effects generated by executing a plan. This contrasts with informed expectations, which identify the atoms in only the current state added by effects of actions in the plan. For instance, if the agent is navigating from point P to point Q, then AEs would maintain the track of the path visited so far during plan execution. In contrast, informed expectations keep track of only the current location that the agent is visiting. Informed expectations are a subset of the AEs. Formally, $X_{\text{inf}}(\pi_i, s_0) \subseteq X_{\text{acc}}(\pi_i, s_0)$. In general, there is no set-subset relation between state expectations and AEs.

Here is a simple example to illustrate the differences among types of expectations. Suppose in some domain the state indicates that there is a car, that the car is blue, and that the day is sunny. Suppose we have a plan with 3 actions. The first action paints the car red; the second action moves the car to location B; and the third action moves the car to location C. In this case, the following are the expectations after executing this 3-step plan:

- Immediate: Expects the car to be in location C.
- State: Expects the car to be in location C, the car to be red, and the day to be sunny.
- Informed: Expects the car to be in location C and the car to be red.
- Accumulated: Recalls that the car was in location B, and expects the car to be in location C and be red.

We close this section by introducing the notion of *atoms of interest*. These are lifted atoms (i.e., atoms with variables) indicating which grounded atoms in the expectations we are interested in monitoring, similar to a human attention mechanism that decides which expected action effects should be monitored and stored in working memory. For instance we may be interested in only atoms that indicate an agent's location but not other atoms (e.g., an atom indicating this agent's color). Formally, given a collection of lifted atoms of interest, I, and expectations, X, then the atoms of interest, *Int*, is defined as: *Int(X,I)* = $\{c \in X$: there is an atom $i \in I$ and a mapping θ of variables into constants such that $i\theta = c\}$.

4. Experimental studies

To assess how much AEs can improve the performance of agents, we designed and implemented two simulated domains that reproduce the two examples described in Section 1: (1) a single agent maritime S&R mission, and (2) a multiagent aerial survey. They represent simplified components of a disaster relief task domain in which autonomous vehicles explore areas where survivors can be found and rescued or assisted. The environment is a 2D grid containing quadrangular areas representing, for instance, islands affected by a tsunami, where survivors need to be provided communication relays by aerial vehicles, or can be found and rescued along the coasts by surface vehicles.

4.1 Variables and Hypotheses

In both domains, the independent (binary) variable is whether AEs are used. In the maritime S&R domain, the dependent variable is the ratio of people rescued and taken to the base, while in the aerial survey domain the dependent variable is the number of grid cells surveyed within a limited time period. We hypothesize that agents that use AEs, named *cumulative*, outperform agents that do not use AEs, named *noncumulative*, in two ways. First, in the maritime S&R, we expect cumulative agents to rescue more people by traversing back the same path where the people fell. Second, in the aerial survey, we expect that they survey more cells by avoiding duplicate explorations of cells.

4.2 Simulation environment

As a testbed for our experiments, we developed and used a simulation environment consisting of a 30 x 30 2D grid, where areas to be surveyed by agents are squares of n x n cells automatically placed in random positions by maintaining a minimum distance of 2 cells between them and between their perimeters and the borders of the grid. The environment can be affected by dynamic natural elements like winds, waves, or flocks of birds or fish. Dynamic elements can be grouped in clusters of cells, and can move along the grid according to a desired speed (number of cells traversed per time) and horizontal or vertical direction. The elements gradually enter and exit the grid from its borders: after exiting the grid, they reappear from the opposite side of the grid, to keep a constant desired number of them on the grid. Before re-entering the grid, the elements get shifted one cell forward perpendicular to their direction (e.g., if the elements moves south, the shift is 1 cell east), so as to increase the variability of regions traversed by the elements.

Agents execute actions in parallel, occupy single cells that they can share, move at a default speed of 1 cell at a time, generate plans using the Pyhop Hierarchical Task Network (HTN) planner (Nau, 2013), compute their paths using A*, and are able to detect discrepancies concerning both immediate and AEs, which they can use to trigger replanning. More details about the agents are provided in the subsections concerning each domain.

Each simulation starts with a different random arrangement of the areas, and of the dynamic elements if the latter are included in the experiment. Simulations are run in batches of 100 units.

The Simulation Algorithm. The simulation cycle executed per tick (time interval) is described in simplified form in Algorithm 1: for each agent, lines 3-6 select the agent's goal (which, in both domains, is the closest area to survey) and lines 7-8 produce a plan to achieve the goal if no plan

is available; lines 9-14 remove from the plan its first action and, if the latter's preconditions are true, execute the action and store its immediate and AEs; lines 15-16 apply the effects of dynamic elements if the latter exist; lined 17-19 check the immediate and AEs and, if any of them is violated, empty the plan, which will trigger replanning in the next iteration (lines 7-8); lines 20-21 update the positions of dynamic elements according to their direction and speed; finally, line 22 updates the number of simulation ticks to possibly execute the next simulation iteration in line 2.

Algorithm 1 The Simulation algorithm

```
1:  procedure SIMULATE(max_ticks, agents, dynamic_elements)
2:      while ticks < max_ticks and ∃agt ∈ agents that has unachieved_goals do:
3:          for agt ∈ agents:
4:              if unachieved_goals ≠ ∅:
5:                  state := UPDATESTATE(agt)
6:                  goal := SELECTGOAL(unachieved_goals)
7:                  if π = ∅:
8:                      π := GENERATEPLAN(agt, goal, state)
9:          for agt ∈ agents:
10:             a := POP(π)
11:             if pre(a) is true:
12:                 X_imm, X_acc := EXECUTE(a)
13:             else
14:                 π := ∅
15:         if dynamic_elements ≠ ∅:
16:             APPLYEFFECTS(dynamic_elements)
17:         for agt ∈ agents:
18:             if X_imm or X_acc are violated:
19:                 π := ∅
20:         if dynamic_elements ≠ ∅:
21:             UPDATEPOSITIONS(dynamic_elements)
22:     ticks += 1
23: end procedure
```

4.3 Single agent domain: Maritime S&R

4.3.1 Mission

In this domain, areas can be considered islands after a tsunami, and an Autonomous Surface Vehicle (ASV) traveling on the water surface has to survey all the border cells of these areas to find and take survivors to a safe base. The agent's mission can be negatively affected by the waves, which cause collected persons to fall off the boat: in such cases the agent needs to search and rescue again the fallen people and then resume the general S&R mission.

4.3.2 Domain features

This domain contains 9 areas of size 4x4 (their total border cells are 4 x 4 = 16, + 4 corner cells = 20, x 9 areas = 180), and 18 survivors (which are 10% of the total 180 border cells) randomly placed along the border cells. The agent starts from the safe base located at cell (0, 0) (top-left cell), can move horizontally and vertically (diagonal movements are avoided to prevent the crossing of the corners of areas), has a capacity of 2 persons, and perceives a survivor when it occupies the same cell as the survivor (there can be at most 1 person in each cell). The agent executes the following behaviors, determined through goal reasoning and HTN planning, until all areas' borders have been explored and all perceived survivors have been taken to the base: (1) select the closest area, (2) compute and follow the shortest path to reach and explore the borders of the selected area, (3) pick all survivors found along the path until the agent's capacity is reached, (4) memorize the locations of survivors beyond capacity to pick them later, (5) take the onboard survivors to the base.

If a wave hits the agent while it is taking survivors to the base, a survivor falls in the water without the agent perceiving the event, so the action of dropping people at the base will produce the immediate discrepancy that the number of dropped persons is smaller than the believed number of people onboard. In such a case, the agent defers the goal of surveying the remaining border cells and switches to the goal of searching and picking the fallen person(s).

The waves move east at a constant speed of 2. They can cover 0%, 10%, 20%, 30% or 40% of the 900 cells of the grid (i.e., they consist of 0, 90, 180, 270 or 360 elements continually traversing the grid), although they impact the ASV only when it traverses wave cells and not area cells (the waves' traversal of area cells occurs only because elements move uniformly across the grid, but has no "physical" effect). Figure 1 displays a closeup of an example grid with an area in gray, the current waves' positions indicated by the wave icons, and the previous cells traversed by the waves indicated by the wave icons in light gray. The figure also shows the following: the agent, after surveying 4 border cells (in light pink) and picking 2 survivors found in two of those cells, now drops at the base only 1 of the 2 survivors, as one of them (the person icon) actually fell in the water when the boat was hit by a wave.

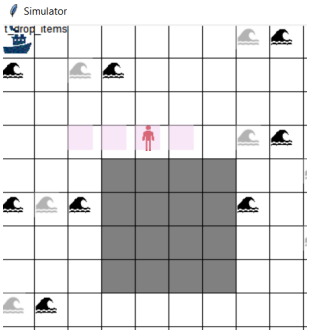

Figure 1. The boat, after surveying some border cells (in light pink) of an area (in dark gray) and picking 2 survivors, has reached the base, but one of the survivors has been pushed into the water by a wave.

4.3.3 Hypothesis and domain-specific variables

The domain-specific hypothesis is that the use of AEs enables the cumulative agent to save (i.e., take to the safe base) more persons than the noncumulative agent that does not use AEs. The reason is that the cumulative agent uses the history of expected effects of its navigation steps to execute them in reverse order, so as to find the survivors that have been pushed off the boat by waves on the path to the base. The domain-specific dependent variable is the ratio of rescued people to the total number of people in each simulation batch. In this domain, the independent variable of the use versus no use of AEs is operationalized as follows:

- the cumulative agent accumulates the expected positions of the path it has traversed (i.e., the atoms of interest are the agent's locations on the path). When it detects an immediate discrepancy between the number of people dropped at the base and the number of people it believed were onboard, then switches to the goal of searching and picking fallen people and traverses backward the expected positions of the path to search and pick the people that have fallen into the water;
- the noncumulative agent does not accumulate the expected positions of the path it believes to have traversed, and therefore cannot use such AEs to traverse backward the same path along which the survivors have fallen. Like the cumulative agent, in case of an immediate discrepancy regarding the number of dropped versus onboard people, it switches to the goal of searching and picking fallen people, but tries to achieve such goal by going back to the latest selected area and surveying again its borders. However, the path from the base to the latest explored area may not be identical to the path previously traversed by the agent, in that the agent may have explored the latest area A coming from another area B (for instance, because it picked a person from a border cell of area A and then picked another person from area B), and therefore the noncumulative agent may miss fallen survivors.

4.3.4 Results

Figure 2 displays the performance of the cumulative and noncumulative agents (i.e., using or not using AEs, respectively). The x-axis displays the percentage of wind cells used in each simulation batch, the main y-axis on the left is the saved persons / total persons ratio, and the secondary y-axis on the right is the simulation duration. The solid lines, red for the cumulative agent and blue for the noncumulative agent, display the average ratio of saved persons, while the dashed lines, once again red for the cumulative agent and blue for the noncumulative agent, display the average duration of the simulations. The graph shows that the cumulative agent always saves all the survivors regardless of the percentages of waves affecting the environment, while the noncumulative agent's S&R performance drops heavily as a function of the percentage of waves. The graph also shows, as expected, that the cumulative agent takes longer than the noncumulative agent in saving all the survivors: this happens because, by traversing back the same path on which the people have fallen, the cumulative agent does not miss any survivors, whereas the noncumulative agent can miss survivors and therefore devotes less time to pick and take them to the base. The differences in performance and times between the cumulative and noncumulative agent imply that there is a trade-off between the goal to rescue as many people as possible and the possible constraint to finish the

S&R mission as quickly as possible, a constraint that might be due, for instance, to fuel shortage or to the need to complete the rescue operations before it is too late to find people still alive. This tradeoff indicates that, in this task domain, the benefits of using AEs depend on context conditions and preferences.

The differences in ratios of saved persons between cumulative and noncumulative agent are not subject to statistical tests because there is no variability in the ratios of the cumulative agent, which always saves 100% of persons. All the differences between the simulation times of the cumulative and noncumulative agent for each level of waves, except for the 0% level (in which case, as expected, both the cumulative and noncumulative agent save 100% of people in almost identical times), measured using the one tail t test for two samples assuming equal variances with df = 198, are statistically significant, with every $p < 0.001$ (e.g., with 10% of waves, noncumulative M = 1465.86, SD = 305.94, cumulative M = 2108.1, SD = 435.85, t = -12,06).

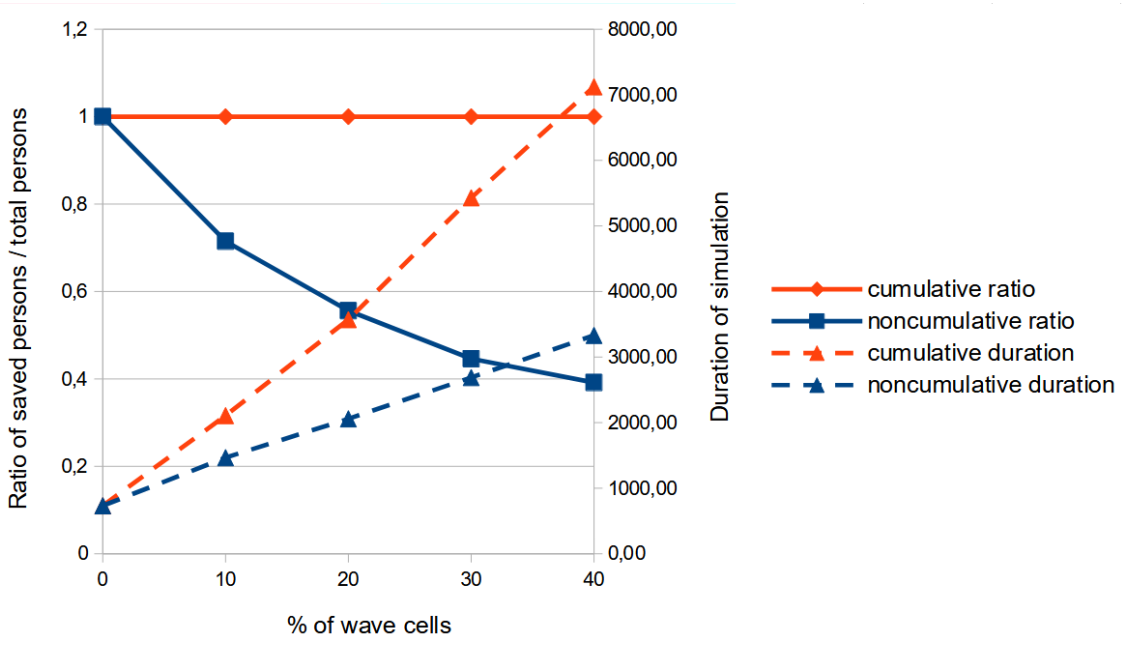

Figure 2. Differences in performance between cumulative / noncumulative agent in the maritime S&R task domain.

4.4 Multi-agent domain: Aerial survey

4.4.1 Mission

In this domain, partially inspired by Johnson et al. (2016), two Autonomous Aerial Vehicles (AAVs) have to survey areas from above, as they would have to do in a disaster relief scenario to find survivors and possibly provide them with communication relays. Unlike the previous domain, here

the AAVs have to survey the internal rather than the border cells of areas, and the dynamic elements that can affect the agents' performance are the agents themselves, as explained below.

4.4.2 Domain features

AAVs can move not only horizontally and vertically, but also diagonally, since they do not have obstacles. They begin their mission from the top left corner as initial location, and survey each unexplored area following a lawnmower pattern (as is usually the case in search tasks), starting from the area's corner closest to them. If, instead, some cells of the target area have already been explored by themselves or other agents, the agents start or continue the survey by moving along the closest cells of the area. Each agent applies to all areas the following actions, determined through goal reasoning and HTN planning: (1) select the not yet fully explored area with the closest unexplored cell (which is a corner cell if the area is totally unexplored)[1]; (2) use such cell as starting point to compute the "survey path" for surveying all the unexplored cells of the selected area (using the lawnmower pattern if the area is totally unexplored, or the closest unexplored cells pattern if the area is partially explored); (3) compute the "navigation path" to the starting cell, which will therefore also be the last cell of the navigation path: and (4) follow the navigation and survey paths. The AAVs can sense other vehicles and the area cells surveyed by them; whenever they find themselves in an identical cell of an area they are surveying, they select a random unexplored cell of the area as new starting point for the survey path, and then replan the navigation and survey paths. To reproduce the situation of the autonomous drones described in Section 1, we do not let agents communicate with each other.

The agents act in parallel and use the same navigation and survey logic: therefore, if they happen to survey the same area concurrently, and only check their actions' preconditions and immediate expectations to possibly trigger replanning, they ultimately traverse the same cells. For instance, if the two agents resume the survey of an area from a different randomly selected cell because they are in the same area cell, they do plan different "closest cells" paths, but the latter end up traversing the same cells, because during action execution none of the preconditions or immediate expectations of the agents is violated (each agent can execute a move action, which successfully changes the agent's position and marks the latter as surveyed). Another, more striking, example is when agents, after finishing the survey of an area using the "closest cells" paths and finding themselves placed a few cells away from each other, select the same closest area as next target, and follow identical "lawnmower paths" to survey it. This happens because the agents generated their plans before any of them had reached the area, and when they reach the area one after the other they are in adjacent cells with no violation of immediate expectations. As shown in Figure 4, the brown agent is traversing the same lawnmower path that the blue agent is traversing.

Notice that the agents do not always select the same next area to survey: in fact, when the agents happen to be in the same area cell and therefore select a different new random cell as starting position for their surveys, their new paths can cause the agents at the end of the survey to be sufficiently distant from each other to subsequently select different areas. Of course, later on the agents may still happen to select the same target area, especially when there are few remaining areas to survey.

1. We consider each area to survey to be an independent goal in view of future experiments about goal reasoning, in which we will study interactions among goals as done in relevant literature, e.g., Johnson et al. (2016)

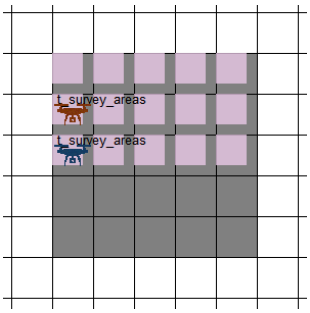

Figure 3. Two noncumulative agents surveying the same area cells (in light pink) of an area (in dark gray).

Of course, the agents traversing the same cells of an area are highly inefficient, as they duplicate their efforts. The agents might prevent this problem by replanning their paths whenever they see another agent in the same area they are surveying, or by doing some form of plan recognition to guess the other agents' exploration paths and adapt their own survey paths accordingly, but this would require more knowledge engineering and the designer's prior analysis of possible interactions among agents. Instead, AEs can prevent the problem in a general way, as described in the following.

4.4.3 Hypothesis and domain-specific variables

The domain-specific hypothesis is that the use of AEs enables the cumulative agents to survey more area cells than the noncumulative agents that do not use AEs, because the cumulative agents replan their survey paths whenever they find a discrepancy between the history of cells surveyed by themselves and the cells surveyed in an area. The domain-specific dependent variable is the ratio of surveyed area cells on the total number of area cells in each simulation batch, where each simulation ends after 100 ticks or when the agent has surveyed all cells of all areas. In this domain, the use versus no use of AEs is operationalized as follows:

- the cumulative agent accumulates the area cells it expects to have surveyed (i.e., the atoms of interest are the agent's traversed cells in the target area) and, whenever its perception of the area's surveyed cells does not correspond to the cells it remembers to have traversed, it replans a more efficient survey path by avoiding the already surveyed cells;

- the noncumulative agent does not accumulate the area cells it expects to have surveyed, and therefore cannot realize whether other cells of that area besides those it planned to explore have been already surveyed, and keeps following the survey path it originally planned, thereby duplicating the effort of the other agent when both agents are surveying the same area.

Notice that each agent (be it cumulative or noncumulative) uses a "(surveyed ?cell)" predicate to store the perceived surveyed cells regardless of which agent surveyed them, and that the cumulative agent stores in its AEs only the cells that it surveys. This enables the cumulative agent to compare the general list of surveyed cells with the AEs concerning its own surveyed cells, and to trigger replanning if the comparison of these lists reveals discrepancies. By using the more detailed

predicate "(surveyed-by ?agt ?cell)" that indicates which agent has surveyed which cell, informed expectations could work as well in this domain. However, by using AEs and the agent's position as atom of interest, we can more easily focus the agent's detection of expectation violations and save computational overhead, as informed expectations might add dozens of atoms besides those that really matter for monitoring the agents' most important expectations.

4.4.4 Results

Table 1 lists the dimensions of areas (as "side x side", i.e., n side cells x n side cells), their cells (n x n), the number of areas, and the total number of cells, used in each simulation batch.

Table 1. Dimension of areas, cells per area, number of areas, and total number of internal cells.

Dimension	Cells	Areas	Tot. cells	Dimension	Cells	Areas	Tot. cells
3x3	9	13	117	6x6	36	4	144
4x4	16	9	144	7x7	49	4	196
5x5	25	4	100	8x8	64	1	64

Figure 4 compares the performances of the cumulative vs noncumulative agents. The x axis shows the number of area cells used in each simulation batch as per Table 1 (since in two cases the number of cells is the same (144), for the sake of clarity we report on each mark of the axis also the *(side x side)* information of each area). The y axis shows the surveyed area cells / total area cells ratio. The red vs blue bars, where error bars indicate standard deviations, show the average ratio of surveyed cells of the cumulative vs noncumulative agents, respectively (the condition with 36 border cells is not included in the chart because there is no difference in performance between the two conditions, as the agents are always able to survey all border cells before the 100 maximum simulation ticks). The bars clearly show that the cumulative agents survey more cells than the noncumulative agents. The bars also show that, in both conditions, the ratio of surveyed cells tends to be inversely proportional to the total number of area cells, as one would expect: in fact, the more cells are to be surveyed, the longer it takes for the agents to survey them. All the differences in performance between cumulative and noncumulative agents, measured using the one tail t test for two samples assuming equal variances, with df = 198, are statistically significant, with every $p < 0.001$ except for the c117 (3x3) condition, in which p = 0.02 (e.g., in condition c196 (7x7), noncumulative M = 0.54, SD = 0.09, cumulative M = 0.73, SD = 0.05, t = -18.00).

We can also notice an unintuitive thing: the performances of cumulative vs noncumulative agents are almost the same in the condition with 117 cells (13 areas of size 3x3), and similar in the condition with 144 cells (9 areas of size 4x4). The reason is that, as mentioned earlier, the more areas are placed in the grid, the more chances are that when the agents end up in different positions after finishing the survey of a common area, they select different new target areas: once they do so, they no longer duplicate their efforts, unless at some point they happen to target the same area again. This is an intervening variable that partially affects the relation between the independent and dependent variable when many areas are in the grid. In fact, with noncumulative agents, all R^2 coefficients between the time (number of simulation ticks) during which the agents have surveyed

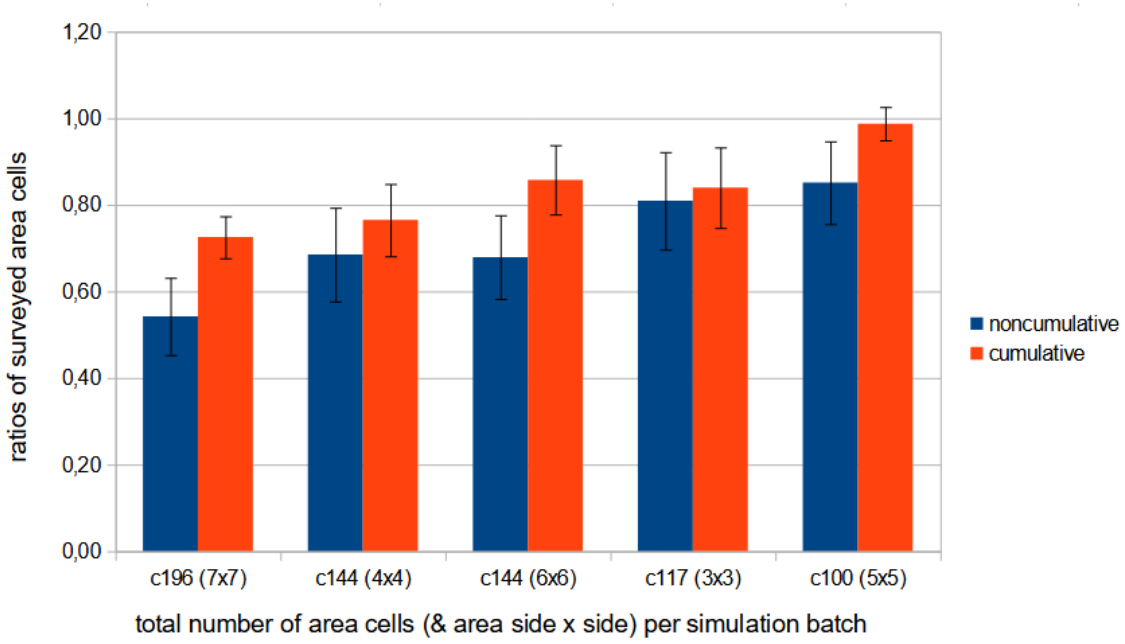

Figure 4. Differences in performance between cumulative / noncumulative agents in the aerial survey task domain.

different areas and the ratios of surveyed cells are above .60 (e.g., in c196 (7x7), R^2 = .90, F(1, 98) = 905,52, $p < 0.001$), with the only exception of the c100 (5x5) condition, in which R^2 = .39. This means that the longer noncumulative agents happen to explore different areas and therefore do not traverse overlapping paths, the more cells they are able to survey. With cumulative agents, the coefficients are high in the two conditions mentioned above (c117, 13 areas of size 3x3: R^2 = .82, F(1, 98) = 451,84, $p < 0.001$; c144, 9 areas of size 4x4: R^2 = .75, F(1, 98) = 295,59, $p < 0.001$) and low in the other conditions (with R^2 ranging between .02 in the c100 (4 areas of 5x5) condition and .25 in the c144 (4 areas of 6x6) condition), which implies that these agents do not need to use AEs when the high number of areas lets them explore different areas in parallel. In other words, the cumulative agents mostly benefit from the use of AEs when there are fewer areas, and therefore when the agents have more chances to survey the same areas and use AEs to adapt their paths and avoid the duplication of efforts.

It is interesting to notice that the cumulative agents in this task domain are producing what can be considered a form of emergent collaboration, in that they improve the global performance of the multiagent system by serendipitously exploiting the positive interactions among their behaviors, without using communication or coordination mechanisms.

5. Final remarks and future work

In this paper we motivated and formalized AEs, which maintain a history of the assumed effects of executed actions. We showed that AEs differ from other types of expectations and provide opportunities to improve performance that are not available with the other types. We then carried out an empirical study of the value of AEs in both a single and multiagent task domain. Our hypothesis that cumulative agents (which use AEs) would outperform noncumulative agents (which do not use AEs) is confirmed in the two selected task domains. The results show that cumulative agents perform much better than noncumulative agents in all the configurations of the simulation environment, with the only exception, in the multiagent aerial survey domain, of two simulation batches containing many small areas that favor the concurrent survey of different areas, thereby reducing the benefit of using AEs to avoid duplicating the agents' efforts. In general, our results show that providing planners with a form of working memory that accumulates the intermediate results of the planning process can benefit the agents' ability to adapt.

We envision several directions for future work. For the sake of experimental rigor we purposely used simple task domains that we studied separately, to ensure clear experimental settings with no or minimal confounding factors. Despite such simplicity, there is already an intervening variable in the aerial survey domain, so more complicated domains would have more confounding factors and their results would be harder to interpret. Also, our empirical studies were not designed to solve the two example problems using the best possible methods, but instead to test how much AEs could make agents more adaptive when solving those problems, which is also why we did not consider resources (e.g., fuel, costs) other than time. Anyway, our ultimate goal is to combine the two domains into a single one where vehicles pursue a common disaster relief mission using AEs for its most efficient accomplishment. In such combined, more complex domain, we plan for heterogeneous agents to use both their own and team-level AEs to implicitly coordinate with each other even in the absence of communication. Application-wise, we also look forward to use AEs in ad hoc agent teams (Mirsky et al., 2022) that may not have shared communication protocols and cannot make specific assumptions about their peers.

We did not compare AEs with informed expectations because the latter require changes to the planner and do not support the features of AEs, but in future work we will compare AEs with informed expectations in terms of memory and computational overhead. The memory overhead of AEs should be linear w/r the number of stored AEs and plan actions involving them, while their computational overhead includes the number of replanning episodes triggered by the violations of AEs. To reduce the frequency of these episodes, we plan to prevent repeated comparisons between the same AEs and observations by first checking for changes in the latter and by assigning time stamps to AEs so as to ignore "stale" ones.

In environments with stochastic action outcomes, AEs could be explored via sampling techniques, such as Monte Carlo Tree Search. In case of partial observability, each AE could be assigned an accuracy estimate e $(0 \leq e \leq 1)$ that denotes the reliability of AEs and varies according to how environmental conditions affect perception. For instance, in the aerial domain, an AE about the survey of an area cell would have $e = 1$ with clear sky and $e = 0.75$ with clouds, and e would decrease over time while the agent keeps surveying with clouds. If the sky becomes clear again, the agent could update the e values of each AE.

We plan to generalize the use of AEs by maintaining the history of all the expectations that are needed to produce the preconditions of later actions: in other words, rather than selecting and using literals of interest as mentioned in Section 3, we want to enable planning algorithms to accumulate the effects of actions that make true the preconditions of actions to execute later in the plan, possibly using learning mechanisms (e.g., reinforcement learning or neuro-symbolic algorithms for selecting which AEs to retain) to automate such a process in environments that exhibit substantial dynamicity or various types of novelty, as in Loyall et al. (2025). In this direction, negative preconditions will be an interesting additional topic to explore.

Acknowledgements

The work of the first author was supported by the Office of Naval Research Global Award N629092412056. All conclusions are those of the authors alone.

References

Ackerman, R., & Thompson, V. A. (2017). Meta-reasoning: Monitoring and control of thinking and reasoning. *Trends in cognitive sciences*, *21*, 607–617.

Aha, D. W. (2018). Goal reasoning: Foundations, emerging applications, and prospects. *AI Magazine*, *39*, 3–24.

Baddeley, A. D., & Hitch, G. (1974). Working memory. *Psychology of Learning and Motivation*, *8*, 47–89. From `https://www.sciencedirect.com/science/article/pii/S0079742108604521`.

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

Cox, M., Mohammad, Z., Kondrakunta, S., Gogineni, V. R., Dannenhauer, D., & Larue, O. (2022). Computational metacognition. *arXiv preprint arXiv:2201.12885*.

Cox, M. T. (2007). Perpetual self-aware cognitive agents. *AI magazine*, *28*, 32–32.

Dannenhauer, D., Munoz-Avila, H., & Cox, M. T. (2016). Informed expectations to guide gda agents in partially observable environments. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (pp. 2493–2499).

Derbinsky, N., Li, J., & Laird, J. (2012). A multi-domain evaluation of scaling in a general episodic memory. *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 193–199).

Fikes, R. E., Hart, P. E., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial intelligence*, *3*, 251–288.

Fox, M., Gerevini, A., Long, D., Serina, I., et al. (2006). Plan stability: Replanning versus plan repair. *ICAPS* (pp. 212–221).

Hammond, K. J. (1990). Case-based planning: A framework for planning from experience. *Cognitive science*, *14*, 385–443.

Hoffmann, J., & Brafman, R. (2005). Contingent planning via heuristic forward search with implicit belief states. *Proc. ICAPS*.

Hoffmann, J., & Brafman, R. I. (2006). Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*, *170*, 507–541.

Johnson, B., Roberts, M., Apker, T., & Aha, D. W. (2016). Goal reasoning with information measures. *Proceedings of the Fourth Conference on Advances in Cognitive Systems* (pp. 1–14). Cognitive Systems Foundation Palo Alto, CA.

Klenk, M., Molineaux, M., & Aha, D. W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, *29*, 187–206.

Kotseruba, I., & Tsotsos, J. K. (2020). 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, *53*, 17–94.

Kushmerick, N., Hanks, S., & Weld, D. S. (1995). An algorithm for probabilistic planning. *Artificial Intelligence*, *76*, 239–286.

Laird, J. E. (2019). *The soar cognitive architecture*. MIT press.

Loyall, B., Pfeffer, A., Niehaus, J., Harradon, M., Rizzo, P., Gee, A., Campolongo, J., Mayer, T., & Steigerwald, J. (2025). Coltrane: A domain-independent system for characterizing and planning in novel situations. *Artificial Intelligence*, (p. 104336).

Martin, L., Jaime, K., Ramos, F., & Robles, F. (2022). Bio-inspired cognitive architecture of episodic memory. *Cognitive Systems Research*, *76*, 26–45.

Menager, D. H., & Choi, D. (2016). A robust implementation of episodic memory for a cognitive architecture. *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Mirsky, R., Carlucho, I., Rahman, A., Fosong, E., Macke, W., Sridharan, M., Stone, P., & Albrecht, S. V. (2022). A survey of ad hoc teamwork research. *European conference on multi-agent systems* (pp. 275–293). Springer.

Munoz-Avila, H., Dannenhauer, D., & Reifsnyder, N. (2019). Is everything going according to plan? expectations in goal reasoning agents. *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 9823–9829).

Nau, D. (2013). Game applications of htn planning with state variables. *Planning in Games: Papers from the ICAPS Workshop*.

Nuxoll, A. M., & Laird, J. E. (2012). Enhancing intelligent agents with episodic memory. *Cognitive Systems Research*, *17*, 34–48.

Tulving, E., et al. (1972). Episodic and semantic memory. *Organization of memory*, *1*, 1.

Wilkins, D. E. (1988). *Practical planning: extending the classical ai planning paradigm*. Morgan Kaufmann.