# Reinforcement Learning at the Cognitive Level in a Belief, Desire, Intention UAS Agent

*David Sacharny, Thomas C. Henderson,*
*Michael Cline and Ben Russon*
*University of Utah*

UUCS-20-13

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

23 October 2020

**Abstract**

We have proposed a lane-based approach to handle large-scale Unmanned Aircraft Systems (UAS) traffic management. Given the number of UAS in the airways simultaneously, it is necessary to imbue each UAS with a learning capability at the cognitive level so that it can optimize its performance in the face of weather, airway congestion and other contingencies. Here we describe a Belief, Desire, Intention (BDI) architecture for the representation and reasoning over cognitive states, where these beliefs include goals for the UAS such as staying in its lane, on heading and at the desired speed. Such goals are represented as logical propositions, and if false, they may be selected as goals (intentions). We apply reinforcement learning to optimize the selection of a plan to achieve the goal of the UAS. That is, a policy is determined which given the cognitive state of the UAS, including for example weather conditions, a plan is selected which achieves the goal with minimal cost and maximal effectiveness.

# Reinforcement Learning at the Cognitive Level in a Belief, Desire, Intention UAS Agent

David Sacharny[1] Thomas C. Henderson[2] Michael Cline[3] Ben Russon[4]

***Abstract*— We have proposed a lane-based approach to handle large-scale Unmanned Aircraft Systems (UAS) traffic management [1]. Given the number of UAS in the airways simultaneously, it is necessary to imbue each UAS with a learning capability at the cognitive level so that it can optimize its performance in the face of weather, airway congestion and other contingencies. Here we describe a Belief, Desire, Intention (BDI) architecture for the representation and reasoning over cognitive states, where these beliefs include goals for the UAS such as staying in its lane, on heading and at the desired speed. Such goals are represented as logical propositions, and if false, they may be selected as goals (intentions). We apply reinforcement learning to optimize the selection of a plan to achieve the goal of the UAS. That is, a policy is determined which given the cognitive state of the UAS, including for example weather conditions, a plan is selected which achieves the goal with minimal cost and maximal effectiveness.**

## I. INTRODUCTION AND BACKGROUND

The coming wave of autonomous, semi-autonomous, and human-operated vehicles in low-altitude airspace over densely populated areas requires a new system to automate air traffic control (ATC). The current system relies heavily on the intuition of human pilots and controllers who benefit from over one-hundred years of recorded trials and errors. Even after millions of test cases per year, contingencies occur that confound experts and result in disastrous outcomes (e.g., the failure of coordination that resulted in a mid-air collision over Uberlingen, Germany in 2002 [2]). The computational intractability of enumerating all possible sequences of actions and outcomes that lead to contingencies is the root-cause for these disasters; if engineers knew the Uberlingen scenario was possible they would have avoided it (after the accident, TCAS software was patched to handle it, albeit at great expense due to the complexity of the software and subsequent testing [3]). However, the density and dynamism of the anticipated low-altitude air traffic mandates an automated approach; it is difficult to imagine human controllers managing the separation of thousands of flights per hour. This is the conclusion of most professionals in aerospace across the United States, particularly the Federal Aviation Administration (FAA) and the National Aeronautics and Space Administration (NASA), as well as in Europe and Asia, where UAS Traffic Management (UTM) systems are being developed rapidly. However, if current ATC systems still experience contingencies after a hundred years, and millions of flights per year, what hope do engineers have in constructing a safe automated traffic management system? This question lies at the heart of this research, and it not only applies to the safe coordinated access of airspace (i.e., maintaining safe separation between aircraft), but also to a plethora of other issues that air traffic controllers face.

The issues that require human intervention, and that make experts nervous about automated air traffic systems, are typically contingency scenarios. Situations that require safe-separation *and* unplanned priority, for example medical rescue, temporary flight restrictions (TFR) due to wildfires, low-fuel, electronics failure, etc. The scenarios, and the combinations or sequences of events that lead to them, are difficult to enumerate (as mentioned before), and therefore difficult to plan for. They are *difficult* because in a concrete sense, the determination of whether a solution exists to return to a nominal state may be as difficult as, or likely more difficult than, the Satisfiability (SAT) problem. In the language of computer science, the public relies on pilots and controllers to *heuristically* search for solutions that maintain our safety. It is likely that humans perform these searches at a high-level, using abstraction (and perhaps analogies) to reduce the space of possible solutions. So far, no one has shown that the brain, or any part of the nervous system routinely and exactly solves NP-complete problems [4]. For this reason, the problem of air traffic control, when considering the automation of what human operators are currently responsible for, falls squarely within the purview of computer science. This problem is fundamentally an issue of cognition and computation.

Considering the cognitive and computational nature of the UTM problem, a good strategy for constructing a system to replace human pilots and controllers becomes clear: reduce computational complexity on all fronts. A direct effect of this strategy is the reduction of the number of possible contingencies because by definition there are fewer states to consider (and by implication, fewer undesirable states). It is our contention that this strategy should be executed via two channels. First is through structure and deconfliction; since safe separation is a constraint that must be satisfied in any contingency scenario, it serves to reason that this problem should be easy to solve, and hence low complexity. This is the foundation that the lane-based approach provides. Second, complexity in cognition should be reduced via abstraction, which we explore using agent based modeling and simulation (ABMS), and the Belief-Desire-Intention (BDI) architecture. With this strategy, we expect that the

[2]Thomas C. Henderson is with the School of Computing, University of Utah, Salt Lake City, UT, USA tch@cs.utah.edu

[1]David Sacharny is with the School of Computing, University of Utah, Salt Lake City, UT, USA sacharny@cs.utah.edu

[3]Michael Cline is with the School of Computing, University of Utah, Salt Lake City, UT, USA m.cline@utah.edu

[4]Ben Russon is with the School of Computing, University of Utah, Salt Lake City, UT, USA benrusson@gmail.com

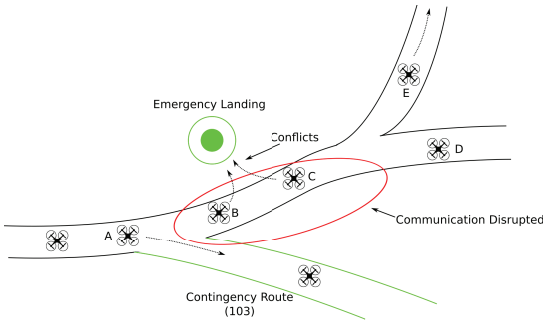


Fig. 1. Scenario: Ground Communication Disrupted for Multiple UAS. Explain

resulting system design will be more robust in the face of contingencies than anything else currently proposed.

### A. The Role of Reinforcement Learning

Reinforcement learning provides engineers with a tool for generating complex computer programs from high-level requirements. Traditionally, engineers receive high-level requirements, such as "UAS should avoid rain," then proceeds to define all the behavioral rules necessary to fulfill that goal. The effort required to define this program logic is a complex function of the software technology, development cycle, and requirements. Estimating the effort required is itself a large topic of software-engineering research [5], and changes to requirements or logic errors can have dramatic costs. Reinforcement learning, however, does not require a manual development of the program logic for the desired behavior. It does require a careful definition of possible states, available actions, and rewards, but it is not necessary to consider all the possible combinations of states and actions.

Consider a scenario in which ground infrastructure supporting UAS communications is disrupted during normal operations (see Figure 1). Currently, the published protocol for handling this contingency is to fly back to base if communications cannot be re-established within a given amount of time [6]. Since this is a pre-defined policy, it is worth considering whether such a policy is robust. For example, depending on how many UAS communications are disrupted, the number of conflicts that result from the simultaneous replanning of multiple agents may have negative cascading effects [7]. As the complexity of the UTM system increases, it becomes harder for experts to enumerate all the failure modes and effects; assigning liability and performing post-failure diagnoses will also be difficult.

A comparison between Traffic Alert and Collision Avoidance System (TCAS), which is currently in widespread use by airlines, and a new system called the Airborne Collision Avoidance System X (ACAS X) offers a compelling analogy. TCAS has been described as an "ad hoc rule-based specification" [3]. Limits to its robustness arise primarily because programmers are unable to anticipate the spectrum of operational scenarios, one of which caused a collision over Uberlingen, Germany in 2002. ACAS X, in contrast, adopts a process of modeling and optimization that improves robustness. Kochenderfer describes an early prototype of ACAS X, in which the anti-collision problem is formulated as a partially observable Markov decision process (POMDP) [3]. In this way, collision and alert preferences are treated as inputs, and the system logic as an output.

### B. Complexity and Cognitive Structure

When the number of possible states and actions is large it forces engineers to carefully create abstractions of states and actions (e.g., object-oriented software), otherwise the program logic becomes fragmented across a large and flat organizational structure. A fragmented program is undesirable because the conceptual links between a high-level requirement and low-level actions are buried in the program logic. On the other hand, a hierarchical structure of states and actions encodes conceptual links explicitly and enables efficient searching through a categorical index (e.g., *Desires* in the BDI architecture).

The Belief-Desire-Intention architecture [8] is a hierarchical organization of states and actions (grouped into *plans*) that was designed specifically for agent models. The architecture not only defines the conceptual structure of a program, but also a process structure that enables dynamic planning, similar to a Markov Decision Process (MDP) ([9], [10], [11], [12]). Organizing the program in this way supports both reasoning by the autonomous agent as well as reasoning by human operators. The structure of desires, intentions, beliefs, and plans coincides well with the reasoning of the human operator. For example, a human observer could ask a BDI agent directly, "What is your current plan?", and the agent could respond, "Correcting my heading to get back in the lane."

### C. Complexity and Airspace Structure

In previous work, we outlined the structure and analytical capabilities made available by the lane-based airspace structure [1], [13]. From a cognitive perspective, the environment in which autonomous agents operate are dynamic and uncertain, and agents are resource-constrained and have only a local view of the world. The lane-based structure provides autonomous agents with more information about the state of the airspace, while requiring less computational effort to reason about it. The primary basis for this is that agents only need to consider the schedule on each lane, as opposed to the entire trajectory of other aircraft in a free-flight airspace structure.

In the free-flight model, where any trajectory is allowed, agents must sample other trajectories and their own to ensure safe separation. The sampling resolution necessary to ensure safe separation, i.e., the discretization of trajectories, depends on how trajectories are specified and the available time and resources to perform the sampling. Each agent must perform this computation every time it considers a new or altered path. The lane-based approach, however, pre-calculates safe-separation in the spatial domain, so agents only need to consider the schedule.

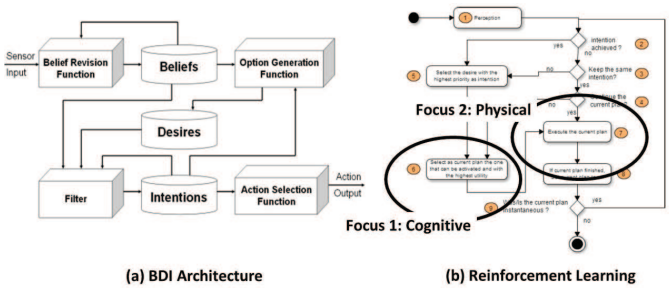


Fig. 2. (a) BDI Architecture (taken from [14]). (b) Reinforcement Learning Focus 1: cognitive level plan selection to achieve goal, and Focus 2: actions in the individual plan (this figure is adapted from [15]).

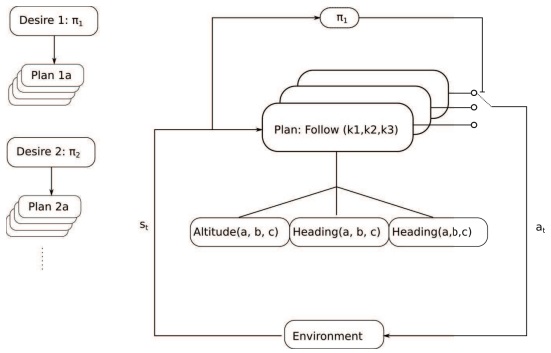


Fig. 3. MDP Representation of BDI Architecture (this figure was adapted from [16]

## II. Learning and BDI

UAS agents have a Belief-Desire-Intention architecture which functions as shown in Figure 2a. The BDI cycle involves updating the beliefs, determining the desires, choosing an intention (goal), and then selecting an appropriate plan to achieve that goal. A *belief* is represented as a disjunction of logical variable literals, and the entire belief set is a conjunction of such beliefs (in Conjunctive Normal Form (CNF)). A *desire* is a belief that the agent would like to make true, and an *intention* is a belief that is a current goal (of which only one is active at a time). The selection of a plan is the action at the cognitive level, and optimal cognitive policies pick the best plan for a given state. Focus 1 in Figure 2b is where the cognitive learning takes place; i.e., the actions are a set of possible plans to achieve a specific goal, and an optimal policy chooses the best plan for a given state. The selected plan is then executed until either completion or preemption. Focus 2 concerns policies at the physical level (i.e., moving through space).

For example, given cognitive states $S = \{S_1 \equiv GoToDestination-NoDrift, S_2 \equiv GoToDestination-Drift, S_3 \equiv Fail, S_4 \equiv Succeed\}$, where there may be several plans to achieve a goal (e.g., shorter or safer routes), and the action is to select one of these plans. Rewards are associated with states and actions, and reinforcement learning is applied to find optimal policies. This is done over a large number of environmental and UTM conditions. Actions are parameterized by considerations like estimated required time, risk, communications connectivity, etc.

At the physical level, a plan may consist of a sequence of lanes with associated entry-exit times. Alternatively, a plan may consist of a sequence of GPS waypoints and times. We have already performed a preliminary study of this aspect of UAS plan optimization (see [17]), and shown how optimal policies (for moving through space) can be determined in the context of environmental conditions (e.g., wind).

Cognitive-level reinforcement learning follows the same process as traditional reinforcement learning. A policy, $\pi$, is learned to maximize the utility, $U(s)$, of high-level states and actions, then it deterministically specifies the action for each state. The goal is to maximize the expected utility. The utility for each state is defined by the Bellman equation:

$$U(s) = R(s) + \gamma max_{a \in A(s)} \sum P(s' \mid s, a) U(s') \quad (1)$$

where $U(s)$ is the utility of state $s$, $a$ is an action, $A(s)$ is the set of actions possible for state $s$, and $P$ is the probability of state $s'$ given state $s$ and action $a$. In our experiments we use *value iteration* to solve for the state utilities; i.e., the above equation is iterated, updating the utility of each state until convergence is achieved. Once the utilities are known, the optimal policy at each state corresponds to the action which maximizes the expected utility from the action:

$$\pi^*(s) = argmax_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s') \quad (2)$$

## III. Experiments

To demonstrate the BDI reinforcement learning at the cognitive level, a simulation was run to determine the optimal policies for UAS agents in an environment that contains rain and wind. The cognitive state of each UAS before the simulation begins is represented by a knowledge-base containing the clauses in Table I. The initial goal is for the UAS to be assigned a mission, so the intention stack is started with the desire to be *ASSIGNED*. Once assigned, the UAS selects intentions from the precedence list shown in Table II (i.e., after being assigned, the next desire is to remain *IN_LANE*).

A combined state and sequence diagram outlining a single reasoning cycle is shown in Figure 4. When the UAS agent receives a percept from the simulator, containing both state information as well as messages from other agents, it parses the data and updates its knowledge-base. In the *Analyze* state, the agent considers its desires and updates its current intention. The *Filter* state involves selecting the optimal plan for the current intention, and in *Execute Plan* the required low-level actions are taken.

The process for generating and assigning flights follows the general design proposed by NASA where a UAS Service Supplier (USS) is responsible for deconflicting flights with other USS in the system. To accommodate the ABMS setup, the USS is also responsible for generating random flight requests and auctioning them to UAS agents. This process is diagrammed in Figure 5.

TABLE I
UAS KNOWLEDGE-BASE AT INITIALIZATION

| ID | Clause |
|---|---|
| 1 | IN_LANE ∧ ON_HEADING ∧ SPEED_OK → NOMINAL |
| 2 | LAST_LANE ∧ AT_NEXT_WAYPT → AT_FINISH |
| 3 | ¬IN_LANE |
| 4 | ¬ON_HEADING |
| 5 | ¬SPEED_OK |
| 6 | ¬ASSIGNED |
| 7 | ¬IN_FLIGHT |
| 8 | ¬AT_START |
| 9 | ¬AT_NEXT_WAYPT |
| 10 | ¬ADVANCE_LANE |
| 11 | ¬WRAP_UP |

TABLE II
UAS DESIRES AND PRECEDENCE

| Desire | Precedence |
|---|---|
| ASSIGNED | 10 |
| IN_LANE | 20 |
| ON_HEADING | 30 |
| SPEED_OK | 40 |
| AT_NEXT_WAYPT | 50 |
| ADVANCE_LANE | 60 |
| WRAP_UP | 70 |

### A. Environment Model

The environment used to train a policy is shown in Figure 6, where solid circles mark areas of rain and dashed circles mark areas of wind. Rain is modeled as a scalar intensity value that decreases with distance from the center of the feature. Rain affects the speed of UAS proportional to the amount of rain. Wind is similarly defined, except it affects both speed and heading of a UAS in a direction tangential to the radius of the wind feature.

### B. Actions

UAS agents in this simulated framework have the ability to set their velocity after each reasoning cycle. In a scenario without wind or rain, the desires are selected in order of precedence (lower precedence happens first). However, due to the dynamics in the environment brought in by wind and rain, agents are affixed with logic that requires replanning when the situation is not *NOMINAL*. In this case, the agent must choose a contingency plan that returns the cognitive state to *NOMINAL*. There are a number of plans that may achieve this, given in Table III. The selection of plans is determined by a policy obtained through reinforcement learning.

### C. Transition Probabilities and Rewards

To generate the transition probabilities, a Monte Carlo simulation was run and a three-dimensional state-action transition matrix, representing the probability $P(s'|s,a)$, was generated from the data.

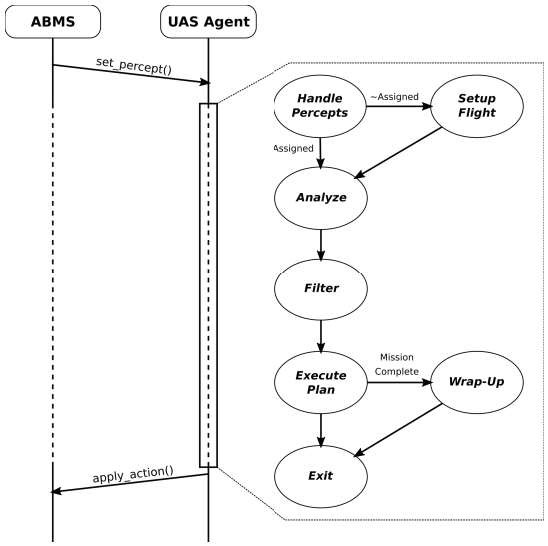


Fig. 4. Combined Sequence and State Diagram Showing Agent Architecture

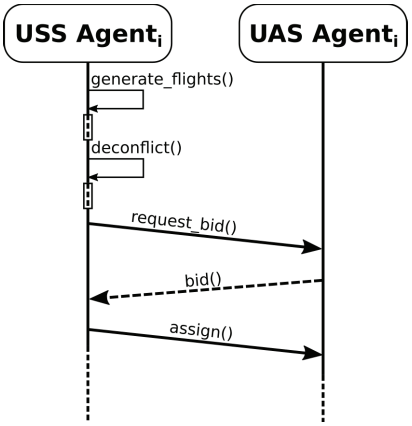


Fig. 5. Flight Path Generation and Assignment

The reward model, $R(s,a)$, considers only the eight states generated by the Cartesian product of *NOMINAL*, *RAIN*, and *WIND*, and the four high-level plans in Table III. A state reward ($R_s$) of $+6$ was assigned to any state that was *NOMINAL*, and $-2$ for any state that was not. Plan rewards were set as follows (reflecting their cost to execute):

$$R_a(\text{FOLLOW_LANE}) = -1$$
$$R_a(\text{CORRECT_SPEED}) = -3$$
$$R_a(\text{CORRECT_HEADING}) = -5$$
$$R_a(\text{GO_TO_LANE}) = -8$$

### D. Policy Selection

A policy was selected by running value iteration and generating state utilities. A trace of the state utilities after each iteration (Eq. 1) is shown in Figure 7. A slice of the transition probability matrix for the plan *FOLLOW_LANE* is depicted by the digraph in Figure 8. The UAS agent then

TABLE III
UAS HIGH-LEVEL PLANS

| Plan | Description |
|---|---|
| CORRECT_HEADING | Heading Optimized Controller |
| CORRECT_SPEED | Speed Optimized Controller |
| FOLLOW_LANE | Main Lane Following Control |
| GO_TO_LANE | Take Immediate Action and Fly to Lane Segment |

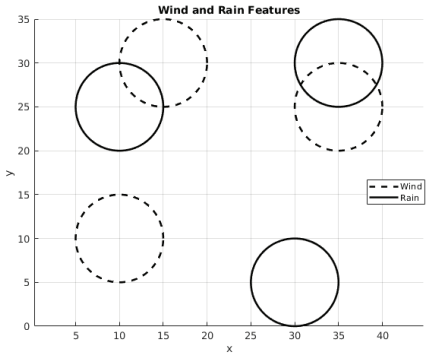


Fig. 6. Wind and Rain Placement in for Training

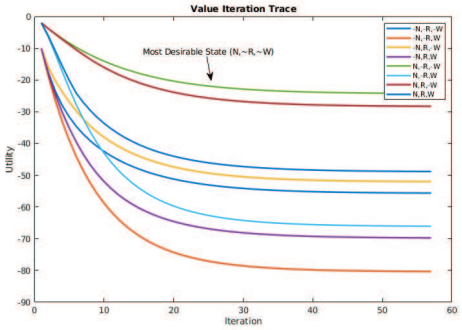


Fig. 7. Value Iteration Trace. N-Nominal, W-Wind, R-Rain

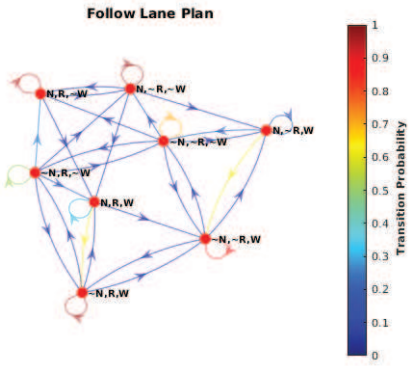


Fig. 8. Transition Probabilities for *Follow Lane* Plan

combines the state utilities and the transition probabilities using Equation 2 to select the optimal plan.

A graphical depiction of the behavior of a single UAS mission without rain or wind contingencies is shown in the plan and state graph in Figure 9. Figure 10 shows the behavior of a UAS navigating the same trajectory through a rain feature with a learned policy. Finally, in Figure 11, a behavior trace of a UAS navigating the rain feature using a programmed policy that deals with the rain contingency directly by correcting its speed.

## IV. DISCUSSION

The experiments demonstrate that reinforcement learning at the cognitive level is a viable option for programming agents in a UTM system. The program in this instance was comprised of a number of plans that could be engineered independently, in contrast to the currently proposed strategy of enumerating risk factors and developing contingency plans in concert across the industry.

In this simple experiment, the resulting policy is guaranteed optimal with respect to the rewards because dynamic programming was used. In a large-scale system, the number of possible states and actions may be too large to precalculate utilities. However, a plan can be engineered in which the UAS agent performs dynamic programming over a narrowed set of states provided by the *Analyze* step in the BDI architecture. Since the BDI architecture supports replanning when a plan fails or when desires change, the agent can avoid over-committing to a plan that did not consider a particular contingency.

This fact is demonstrated in the experimental output of Figures 9, 10, and 11. The explicitly programmed policy is brittle in the face of a rain contingency because the speed correction causes the UAS to overshoot lane waypoints. The learned policy selected a different plan, one that corrects heading and speed concurrently, and proves to be more robust.

## V. CONCLUSION AND FUTURE RESEARCH

We have shown how states and actions at the cognitive level can be combined with reinforcement learning to generate optimal policies for UAS agents. Additionally, the BDI architecture provides a convenient structure for defining high-level states and actions, while reducing the engineering complexity by allowing plans to be designed independently. The benefit of this approach is that it reduces the amount of programming logic required to build robust policies, and utilizes a logic structure that supports operational insight since decisions are expressed in a human-understandable form.

In upcoming research, our intention is to demonstrate the performance of the combined cognitive-level reinforcement learning BDI architecture with respect to contingency scenarios where a large number of states exist. In these scenarios the agent must replan when optimal plans do not produce the correct outcome due to unforeseen states.

Also, the effects of replanning and contingency handling on the aggregate state of all UAS agents in a UTM system must be understood. If it can be shown that the best available policy for UAS agents includes dynamic replanning and the

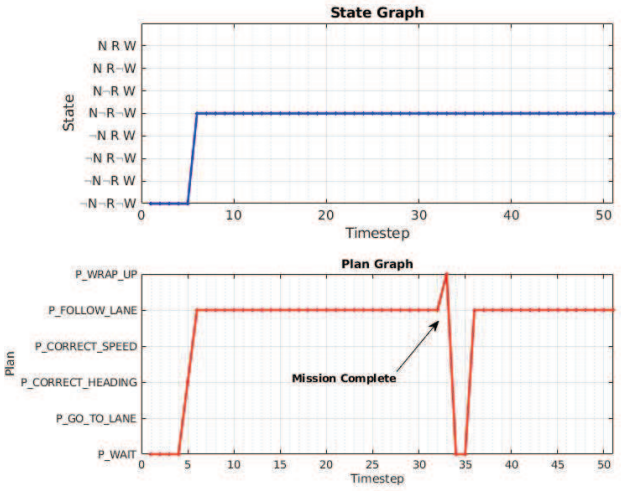


Fig. 9. Nominal Behavior without Contingencies

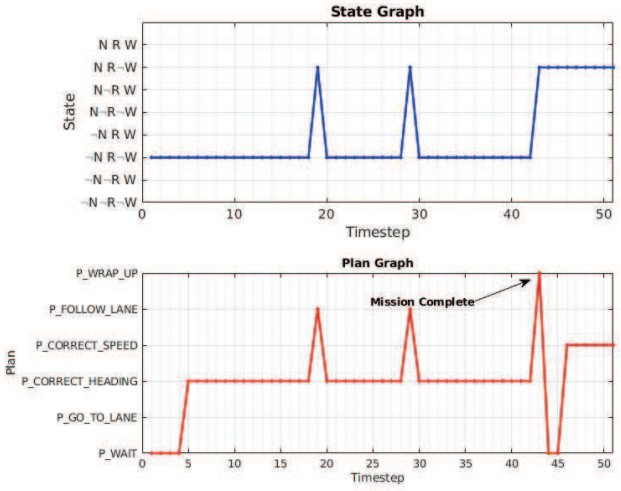


Fig. 10. Learned Policy with Rain Contingency

proposed cognitive structure, then this strategy will enable a more rapid adoption of autonomous agents due to the decreased engineering complexity.

## REFERENCES


[1] D. Sacharny and T. C. Henderson, "A Lane-Based Approach for Large-Scale Strategic Conflict Management for UAS Service Suppliers," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. Atlanta, GA: Institute of Electrical and Electronics Engineers (IEEE), June 2019, pp. 937–945.

[2] G. F. B. of Aircraft Accidents Investigation, "Investigation Report," Bundesstelle für Flugunfalluntersuchung, Hermann-Blenk-Str. 16, 38108 Braunschweig, Tech. Rep., May 2004.

[3] M. J. Kochenderfer, C. Amato, G. Chowdhary, J. P. How, H. J. D. Reynolds, J. R. Thornton, P. A. Torres-Carrasquillo, N. K. Üre, and J. Vian, *Decision Making Under Uncertainty: Theory and Application*, 1st ed. The MIT Press, 2015.

[4] E. Dietrich and A. B. Markman, *Cognitive Dynamics: Conceptual and Representational Change in Humans and Machines*. New York, NY: Psychology Press, 2014.


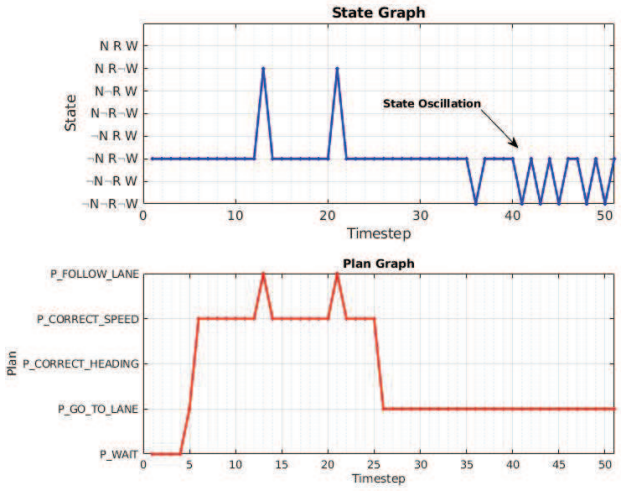


Fig. 11. Programmed Policy with Rain Contingency


[5] B. Boehm, C. Abts, and S. Chulani, "Software Development Cost Estimation Approaches – A Survey," *Annals of Software Engineering*, vol. 10, no. 1, pp. 177–205, Nov. 2000.

[6] J. Baculi and C. Ippolito, "Onboard Decision-Making for Nominal and Contingency sUAS Flight," in *AIAA Scitech 2019 Forum*. San Diego, California: American Institute of Aeronautics and Astronautics Inc, AIAA, 2019.

[7] M. R. Jardin, "Analytical Relationships Between Conflict Counts and Air-Traffic Density," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 6, pp. 1150–1156, 2005.

[8] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge, "The Belief-Desire-Intention Model of Agency," in *Intelligent Agents V: Agents Theories, Architectures, and Languages*, J. P. Müller, A. S. Rao, and M. P. Singh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 1–10.

[9] G. Rens and D. Moodley, "A Hybrid POMDP-BDI Agent Architecture with Online Stochastic Planning and Plan Caching," *Cognitive Systems Research*, vol. 43, pp. 1–20, 2017.

[10] G. I. Simari and S. Parsons, "On the Relationship between MDPs and the BDI Architecture," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 1041–1048.

[11] M. Schut and M. Wooldridge, "Principles of Intention Reconsideration," in *Proceedings of the Fifth International Conference on Autonomous Agents*, ser. AGENTS '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 340–347.

[12] M. Schut, M. Wooldridge, and S. Parsons, "On Partially Observable MDPs and BDI Models," in *Foundations and Applications of Multi-Agent Systems*, M. d'Inverno, M. Luck, M. Fisher, and C. Preist, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 243–259.

[13] T. C. Henderson, D. Sacharny, and M. Cline, "An Efficient Strategic Deconfliction Algorithm for Lane-Based Large-Scale UAV Flight Planning," University of Utah, Tech. Rep. UUCS-19-005, September 2019.

[14] I. of Informatics, "Belief, Desire and Intention Agents," 2019, web Page for Research Group.

[15] P. Caillou, B. Gandou, A. Grignard, C. Q. Truoung, and P. Taillandier, "A Simple to Use BDI Architecture for Agent Based Modeling and Simulation," in *The 11th Conference of the European Social Simulation Association*, Groningen, The Netherlands, September 2015.

[16] P.-L. Bacon, J. Harb, and D. Precup, "The Option-Critic Architecture," in *The Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, CA, February 2017.

[17] D. Sacharny and T. C. Henderson, "Optimal Policies in Complex Large-scale UAS Traffic Management," in *IEEE Conference on Industrial Cyber-Physical Systems*, Taipei, Taiwan, May 2019.