Novelty Accommodating Multi-Agent Planning in High Fidelity Simulated Open World

James Chao
Naval Information Warfare Center Pacific
San Diego, United States
james.chao.civ@us.navy.mil

Wiktor Piotrowski
Palo Alto Research Center
San Francisco, United States
wiktorpi@parc.com

Mitch Manzanares
Naval Information Warfare Center Pacific
San Diego, United States
mitch.c.manzanares.civ@us.navy.mil

Douglas S. Lange
Naval Information Warfare Center Pacific
San Diego, United States
douglas.s.lange2.civ@us.navy.mil

ABSTRACT

Autonomous agents acting in real-world environments often need to reason with unknown novelties interfering with their plan execution. Novelty is an unexpected phenomenon that can alter the core characteristics, composition, and dynamics of the environment. Novelty can occur at any time in any sufficiently complex environment without any prior notice or explanation. Previous studies show that novelty has catastrophic impact on agent performance. Intelligent agents reason with an internal model of the world to understand the intricacies of their environment and to successfully execute their plans. The introduction of novelty into the environment usually renders their internal model inaccurate and the generated plans no longer applicable. Novelty is particularly prevalent in the real world where domain-specific and even predicted novelty-specific approaches are used to mitigate the novelty's impact. In this work, we demonstrate that a domain-independent AI agent designed to detect, characterize, and accommodate novelty in smaller-scope physics-based games such as Angry Birds and Cartpole can be adapted to successfully perform and reason with novelty in realistic high-fidelity simulator of the military domain.

KEYWORDS

Novelty, Planning, Open World

ACM Reference Format:
James Chao, Wiktor Piotrowski, Mitch Manzanares, and Douglas S. Lange. 2023. Novelty Accommodating Multi-Agent Planning in High Fidelity Simulated Open World. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 7 pages.

1 INTRODUCTION

Current artificial intelligence (AI) systems excel in narrow-scoped closed worlds such as board games and image classification. However, AI systems performance drops when accommodating to constantly changing conditions [7]. On the other hand, automated Planning has long been utilized for military applications. For example, aircrew decision aiding modern military air missions [18], generating complex battle plans for military tactical forces [15], handling crisis and disaster relief [26], and controlling autonomous unmanned aerial vehicle in beyond-visual-range combat [9]. The military not only requires a quick and decisive course of action (COA), but also flexibility to handle unforeseen situations. Good crisis management is characterized by quick response, decisive action, and flexibility to adapt to changing environments [26].

In another example, unmanned aerial vehicles (UAVs) engaged in air combat, seen in [9], continuously monitoring the actions of opponent aircraft and performing behavior recognition to predict their opponents' current plans and targets. However, this does not include enemy behavior or weapons that are far out of scope, rendering the original domain knowledge no longer feasible. Furthermore, a simple goal change or replanning can no longer resolve the problem. In order to accommodate meaningful and impactful real-world novelty, we propose integrating novelty reasoning approaches into conventional state of the art AI systems using a realistic military simulator. This is a step towards showing how AI adapts to open-real-world messiness.

In the realistic military simulator we use, there is an existing baseline AI agent that does not perform well when encountering novelty during a mission. The non-novelty agent determines its COAs using a branch-and-bound search algorithm. To improve the existing AI and be able to handle novelty will include updating the existing agent to use PDDL+ with model manipulation operators (MMOs) to monitor and repair in an event of novelty detection via a model consistency checker. Finally, an execution engine is required to translate the PDDL+ plan to run in the realistic military simulator.

In this work, we adapt Hydra [24], an existing single-agent AI planning-based novelty-aware approach to executing complex real-world scenarios in a high-fidelity military simulator. Previously, Hydra has mostly been applied to physics-based single-agent lower fidelity games such as Angry Birds [11] and OpenAI Gym's C+artpole [4]. In a high-level overview, instead of relying solely on the AI planner integrated in the simulator, we delegate the mid- and high-level decision making to Hydra while continuing to exploit the integrated AI for low-level planning and execution. The resulting composite agent architecture enables reasoning with novelty introduced in the multi-agent high-fidelity simulator which was previously not possible. The novelty-aware functionality is of great importance in military applications where environmental phenomena and enemy behavior must be accurately captured and

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

reasoned with. We demonstrate improvement over the baseline AI agent decision making.

2 RELATED WORK

Novelty [3] [16] [1] [5] that is unknown, unexpected, or out of distribution is important to consider when moving AI agents from closed world games and simulations to the open real world. There exists many domains where research is currently being performed, including Monopoly [14], Minecraft [20], [19], Angry Birds [11], Doom [13], Cartpole [3], natural language processing [17], and computer vision [12]. In general, novelty that does not effect any outcome of the system is considered nuisance novelty and does not have to be addressed.

Hypothesis-Guided Model Revision over Multiple Aligned Representations (Hydra) [24] is an AI framework that uses a model-based planning approach to detect, characterize, and accommodate to various novelties in multiple domains including Angry Birds, Minecraft, and Cartpole. Hydra uses PDDL+ [10], a standardized planning modeling language for mixed discrete-continuous systems, to capture the composition and dynamics of the modeled scenario. PDDL+ is a highly expressive language which enables accurate capture of dynamical system characteristics. Currently, Hydra auto-generates PDDL+ problem instance files from the environment perception information, which are then combined with a manually written domain (describing the generic system dynamics) to create a full planning model (note that the domain only needs to be written once per environment). Hydra then uses the PDDL+ planning model to detect novelty via consistency checking. A consistency checker detects divergence between the expected behavior (i.e., the planning trace) and the observations collected when executing the plan in the simulation environment. Consistency checkers can be general (i.e., taking into account all components of the environment) or focused (i.e., only focusing on a smaller subset of the environment features (e.g., unexpected behavior of a particular agent acting within the environment, or unexpected effect of an executed action). Initially, the PDDL+ planning model only accounts for non-novel behavior since novelty existence is not proved and the type of novelty is not known. Once an impactful novelty is detected via consistency checking, Hydra engages the repair module to find an explanation for the divergent behavior. The PDDL+ planning model is modified until the planning-based predictions are re-aligned with the observations. Currently, this is performed via a heuristic search process which adjusts the values of state variables such that the inconsistency (i.e., the euclidean distance between expected and observed state trajectories) is minimized.

AFSIM [8] provides a realistic simulation of behavior for the entities including F-35 fighter jets and surface-to-air missiles (SAM). The environment simulates real world environmental characteristics, including partial observability, stochasticity, multi-agent, dynamic, sequential, continuous, and asymmetric battles. An example of each characteristic is provided, but not limited to, the below examples:

- Partially observable: Nothing is observed at the beginning of a battle, the agent can use sensors to provide observations from its limited abilities.
- Stochasticity: Missiles fired at a target may not always hit, radar and sensors may not always function properly. Starting positions of battles will vary (it is worth noting that stochasticity is different from novelty that we accommodate in this paper).
- Adversarial: inherently competitive multi-agent environment.
- Dynamic: the environment is constantly changing in real time.
- Sequential: agent actions will have a long term effect for the mission at a later time. Continuous: States, time, actions are continuous.
- Asymmetric: rewards and objectives are different for the agents on opposite sides.
- Noisy: Sensor errors can cause confusion to the AI.

3 PROBLEM SETUP

We will simulate the use case from the Hollywood movie Top Gun:Maverick. Where two aircraft must cooperate and fire separate missiles to destroy the enemy target. During the mission, we consider the novelty which extends the effective range of enemy SAM missiles. This violates the assumption of the friendly force's perception and default planning models which underestimate the safe distance from the enemy SAM sites. As a result, the protagonist team led by Maverick is shot down by unexpected enemy weapons because they flew too close to the enemy, assuming they were at a safe distance.

To allow PDDL+ planning, required by the Hydra architecture, in the AFSIM environment, the environment is modified to resemble an OpenAI Gym interface [4] through a Python framework, where real time is segmented into discrete steps, missions are segmented into episodes (battles in military terms), and multiple episodes are segmented into a tournament (campaign in military terms).

Following the theory of Occam's razor [23], the states and actions will be limited to the ones relevant to the current mission using domain knowledge. Primitive actions will be combined into higher level actions. For example, actions such as tasking and routing a friendly surveillance and reconnaissance autonomous drone to gain information on the operation area will be automatically performed at the beginning of a battle.

The observations of enemy entities for both experiments will be assumed to be sensed by the surveillance drone in the beginning of all battles. There will be 10 enemy radars acting as enemy sensors (used by different military commands in the enemy chain of command), a supply depot, an ammo storage station, a SAM site, a chemical storage unit, a command post, a defense headquarters. The mission area and entities are shown in figure 2. The teal circle represents the starting location of the F-35 aircraft, it also represents the starting location of the autonomous surveillance drone, the red rectangle with a teal outline represents the enemy SAM location, the red circle with a teal outline represents where the target radar station is located, the the red triangles denote enemy radar sensors, and the red pentagons represent all other enemy entities.

4 PLANNING DOMAIN FORMALIZATION

The planning domain requires a transition system such as a simulation environment to generate a plan while maximizing a utility function such as shortest path or highest reward. Because novelty detection, characterization, and accommodation is the main research, we also require a formal definition for the novelty response problem. Following the definition of a planning domain and a concept of novelty, we use the Hydra methodology to calculate the degree in which the planning model is consistent with the environment (for our case a open-real-world-like high fidelity simulator), and finally we define a set of MMOs to facilitate a meta model repair until the model becomes more consistent with the environment. Formally:

Definition 1. ***Environment*** *Let E be the environment, a transition system defined as:*

$$E = \langle \mathcal{S}, \mathcal{S}_I, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{V} \rangle \tag{1}$$

where $\mathcal{S}$ is the infinite set of states, $\mathcal{S}_I \subseteq \mathcal{S}$ is a set of possible initial states, $\mathcal{A}$ is a set of possible actions, $\mathcal{E}$ is a set of possible events, $\mathcal{G}$ is a set of possible goals, and $\mathcal{V}$ is a set of domain variables including both discrete and numeric.

Definition 2. ***Novelty response problem*** *Following [24], let $\prod$ be the novelty response problem defined as:*

$$\prod = \langle E, \varphi, t_N \rangle \tag{2}$$

where E is the transition system environment, φ is the novelty function, and t_N is a non-negative integer specifying the battle in which novelty φ is introduced within a tournament. In the military domain we operate in, the terms battle is interchangeable with episode, and campaign is interchangeable with tournament.

Definition 3. ***Consistency*** *Let C be the consistency checking function defined as:*

$$C : \mathcal{M} \times E \times \pi \times t_e \times t_o \times \mathcal{T} \rightarrow \mathbb{R}_{\geq 0} \tag{3}$$

where $\mathcal{M}$ is the PDDL+ model of mapping E to the realistic high fidelity simulator, π is a sequential plan that solves solves E to reach $\mathcal{G}$, t_e is the set of trajectory [22] we expect to observe using model $\mathcal{M}$ to solve the planning problem, t_o is the set of trajectory we actually observe in the environment when applying the plan π, and $\mathcal{T}$ is a threshold, where if the distance between t_e and t_o exceeds, we can assume novelty is detected. The threshold is a hyperparameter that can be "consistency shaped" based on domain knowledge. In theory, consistency checking does not required domain knowledge, in practice, it requires domain knowledge to reduce computation requirements. The domain knowledge does not map one to one to unknown novelty, but rather to parameters an agent has at its disposal to detect and accommodate the novelty.

A Model Manipulation Operator (MMO) is a single change to the agent's internal model. In this work we limit the scope of MMOs to modifying the values of variables present in the agent's internal model by a predetermined interval. MMOs are then used in the model repair mechanism to accommodate novelty by adapting the agent's internal model. In fact, repair finds a sequence of MMOs that, when applied to the agent's internal model $\mathcal{M}$ yields an updated model $\mathcal{M}'$ which accounts for the introduced novelty.

Definition 4. ***MMO*** *In the scope of this work, an MMO is a function $m : V \times \Delta V \rightarrow V$, where $V \in \mathbb{R}$ is a numeric variable present in the agent's model $\mathcal{M}$, and $\Delta V \in \mathbb{R}$ is the numeric change to the value by some interval.*

In practice, an MMO is applied in a straightforward manner $v(\mathcal{M}) = v(\mathcal{M}) \pm \Delta v(\mathcal{M})$ where $v(\mathcal{M})$ is the numeric value of some variable in model $\mathcal{M}$ and $\Delta v(\mathcal{M})$ is a predefined change interval specific to that model variable $v(\mathcal{M})$. By taking advantage of notation, this approach can also be extended to propositions $p(\mathcal{M})$ by casting each one as a numeric variable $v(p(\mathcal{M}))$. After MMOs have been applied, the variable is then re-cast into a true proposition such that $p(\mathcal{M}) = True$ if $v(p(\mathcal{M})) > 0$ and $p(\mathcal{M}) = False$ otherwise.

Following from the above MMO definition, a repair R is a function which takes in a model $\mathcal{M}$ and a sequence of MMOs $\{m\}$ that modify the model's variables. The repair returns the modified model $\mathcal{M}'$.

Definition 5. ***Repair*** *Let repair R be a function $R(\mathcal{M}, \{m\}) \rightarrow \mathcal{M}'$, where $\mathcal{M}$ is the model of the environment E, and $\{m\}$ is the a set of MMOs defined over model $\mathcal{M}$. The repair function yields an updated model $\mathcal{M}'$ which is generated by applying $\{m\}$ to the default model $\mathcal{M}$*

5 IMPLEMENTING NOVELTY ACCOMMODATING AGENTS

5.1 High Level Architecture

Figure 1 describes the components and information flow of the agent architecture. The PDDL+ model generator automatically builds a PDDL+ problem from initial observations and intrinsic assumptions about the environment. A full model $\mathcal{M}$ is created by combining the auto-generated problem with a general manually-defined PDDL+ domain. A PDDL+ planner will solve the model $\mathcal{M}$ for a plan π, an execution engine will translate the plan into instructions executable in the AFSIM environment. After each battle t_N in AFSIM, the agent will calculate a consistency score C comparing the expected outcomes t_e and observed outcomes t_o. If consistency C exceeds threshold $\mathcal{T}$ it is likely that the underlying environment E has been substantially altered by novelty, beyond interference from noisy sensor readings. The agent then starts the meta model repair process which adjusts model $\mathcal{M}$ by iteratively applying MMOs m.

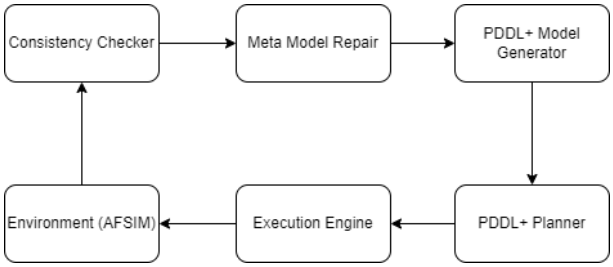

Figure 1: Agent Architecture

5.2 PDDL+ and Execution Engine

In order to utilize the Hydra novelty detection, characterization, and accommodation, it is required to define a PDDL+ model and search for a plan to execute in the pre-novelty environment E.

The continuous mission area space is discretized into a grid space as shown in figure 2. The infinitely large and continuous action space is discretized into five actions: move in four directions one cell and fire the JDAM (i.e., missiles with additional precision guidance kit) when you are within range of 1 cell (28,000 meter JDAM range [2], while a cell length is estimated at 25,000 meters). The actions can be further broken down into similar lower level actions that will be used to execute actions in AFSIM. There will be a PDDL+ event of the F-35 aircraft being shot down by the enemy SAM if it flies within range of the enemy SAM.

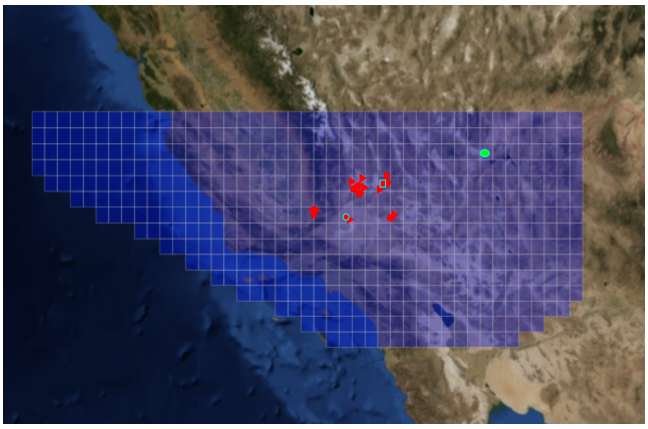

Figure 2: Mission Area Grid And Entities: teal circle is F-35 aircraft and autonomous surveillance drone, red rectangle with teal outline is the enemy SAM, red circle with teal outline is the target radar station, red triangles are enemy radar sensors, and red pentagons are all other enemy entities

States, time, actions are all continuous variables in AFSIM: The continuous actions controlling air movement with high fidelity physics are taken in a continuous timeline. Time is continuous and there is no concept of a discrete time tick. Space is continuous and there is no concept of a discrete grid cell. However, the PDDL+ planner uses a discretization-based approach to solve planning tasks, discretizing it into a geo-spatial grid, time step, and discrete actions. The execution engine will use a low-level planner to translate the discrete variables used by the PDDL+ model back into continuous variables for the environment.

A move command is in one of the four cardinal directions (north, south, west, and east). Each grid cell will have its location defined as an integer $I \subseteq \mathbb{N}$ for the column and row coordinates in the grid. The engagement area composed of grid cells is irregularly shaped, meaning not a rectangular shape. The centroid point of each grid cell is mapped to a geographical latitude and longitude, and the execution engine will calculate the geo-coordinate and route the F-35 between cells. The F-35 position will be kept track using a column and row integer $I \subseteq \mathbb{N}$.

Transitioning actions between the continuous time in the real-world-like high fidelity environment and the discrete time in our PDDL+ planner provides an engineering challenge. To solve this problem, we 1) take the continuous timeline and divide it into discrete time ticks, which are chosen as a hyperparameter of the PDDL+ planner (denoted as Δt).

The domain contains move actions, which can route each aircraft between cells in cardinal directions (north, south, east, and west). A fire weapon action is also available for the agent, with the precondition that the friendly plane is in range of its target. The maximum range to fire a specific weapon is calculated using Manhattan distance between the grid cells:

$$|p_{row} - t_{row}| + |p_{col} - t_{col}| \leq p_r \tag{4}$$

Where p_{col} is the column number of the F-35 aircraft, t_{col} is the column number of the target, p_{row} is the row number of the F-35 aircraft, t_{row} is the row number of the target, and p_r is the range of the JDAM missile.

Similarly, an event is defined where the friendly plane will be shot down if it is in range of the enemy SAM missile range. Note this is using our internal model of the enemy weapon range. A PDDL+ event will have a precondition where the friendly plane will be shot down if we move within the enemy SAM range represented in Manhattan distance, with equation:

$$|s_{col} - p_{col}| + |s_{row} - p_{row}| \leq s_r \tag{5}$$

Where p_{col} is the column number of the F-35 aircraft, s_{col} is the column number of the enemy SAM, p_{row} is the row number of the F-35 aircraft, s_{row} is the row number of the enemy SAM, and s_r is the range of the enemy SAM. The enemy missile range will be a numeric MMO where our agent can adjust the internal model when dealing with novelty.

The Manhattan distance (w.r.t. grid cells) of the friendly plane range is $p_r = 1$ and enemy SAM range is $s_r = 2$ is shown in Figure 3.

Because the continuous state space is translated into a discrete grid, the enemy SAM is likely not in the exact center of a cell, therefore, the range may include additional or fewer cells than a perfect circle, the model with the longest possible range of the enemy SAM and shortest possible range of our F-35 aircraft will be used to err on the side of caution.

A scheduler will parse the plan involving every action of every agent, and convert each routing end point location to a geo-coordinate with latitude and longitude values. The center point of each grid cell, which we can get information from the environment, is mapped to geo-coordinates in the AFSIM environment, and we convert the PDDL+ plan into a AFSIM plan that can be executed in sequential order according to the PDDL+ plan. For example, a particular grid cell can be queried from the environment to map to geographic coordinates (60.14405310652675, 169.73387958469417). This data will be encoded into a pre-existing table of domain knowledge.

We introduce a hyperparameter t which designates the time delay to execute actions between aircraft. We set t to 18 seconds. 5 nautical miles generally allows plenty of room and distance to maneuver, for example, an evasive maneuver if an emergency was to happen, and not cause an air to air collision between the leader

and the wingman. The aircraft speed is about 536 meters per second [25], and 5 nautical miles is 9260 meters, thus $9260m/536\frac{m}{s} = 18$ seconds.

Details of the PDDL+ implementation, scheduling, and execution can be found at [6].

5.3 Novelty Injection

We ran campaigns experiments $\prod$ consisting of 20 battles each. The novelty φ was injected into $\prod$ in the second battle ($t_N = 2$) for both domains. The introduced novelty are changes to the value of the enemy SAM range, from 40,000 meters to 90,000 meters. The expected result is the SAM range increasing from 2 cells up to 4 cells (depending on the SAM exact location), as each cell length is estimated to be 25,000 meters. Further novelty examples and injections can be found in the novelty paper [5]. And the detection and accommodation to such paper will be addressed in future work.

5.4 Consistency Checking

The consistency checking $C \in \mathbb{R}_{\geq 0}$ is a calculated number representing how accurate the model $\mathcal{M}$ represents the realistic high fidelity simulator transition system E. Consistency $C = 0$ means that the agent is operating on an internal model $\mathcal{M}$ that is perfectly aligned with the simulation environment E. The higher the value of C, the less accurate the agent internal model represents the environment E. t_e is simulating using the PDDL+ model $\mathcal{M}$. t_o is generated by converting direct observations of the environment into PDDL+ to calculate C using equation 6 (because t_e and t_o has to be the same datatype). While comparing every single variable V will be ideal in catching any model inconsistency, however, computation limitations of calculating ∞ amounts of ΔV makes that infeasible, so we focus on comparing the MMOs and the Euclidean distance between the trajectories of the defined MMOs that are defined using domain knowledge.

$$C = \sum_i \gamma^i \cdot \|t_o(\pi, E)[i] - t_e(\pi, \mathcal{M})[i]\| \tag{6}$$

where $t_o(\pi, E)$ is the observed trajectory of executing the plan π in the environment E. And $t_e(\pi, \mathcal{M})$ is the expected trajectory of executing the plan π using the PDDL+ model. $0 < \gamma < 1$ is a discount factor to account for compounding errors of Euclidean distances for later states.

5.5 Meta Model Repair

The meta model repair is triggered once the consistency C of the default model $\mathcal{M}$ exceeds the threshold $\mathcal{T}$. Repair aims to alter the model $\mathcal{M}$ so that the expected trajectory t_e (yielded by the planner, based on $\mathcal{M}$) is consistent with the trajectory t_o observed when executing plan π in simulation environment E. As stated in [24], defining appropriate MMOs is the key to a good repair that leads to good novelty accommodation. Although monitoring every variable $v \in \mathcal{M}$ will create the most general agent, the search space might become too large to find feasible repairs in a reasonable time. Thus, currently, model variables which MMOs will modify during repair are chosen manually. In the presented missions, as a proof of concept, we labeled the missile range variable as "repairable" via MMOs being one of the most mission critical variables V in our environment E. The main observation of interest is whether the F-35 aircraft and the enemy target is destroyed. Consistency score will yield $C = 1.0$ if the F-35 aircraft is destroyed and the enemy target is fully functional, and a $C = 0.0$ score if the F-35 aircraft survives and the enemy target asset is destroyed.

Algorithm 1: PDDL+ meta model repair algorithm

```
C_best ← EstimateConsistency(M, π, t_e, t_o, T)
while C_best ≥ T do
    forall MMO ∈ m do
        M' ← R(M, MMO)
        C_new ← EstimateConsistency(M, E, π, t_e, t_o, T)
        if C_new < C_best then
            C_best ← C_new if C_best ≤ T then
                M ← M'
            end
        end
        UndoUpdateM(MMO)
    end
    return M
end
```

The search-based algorithm for model repair is domain-independent. The model repair will follow algorithm 1, which shows the pseudo-code for the meta model repair. First, we check the consistency score C_{best}, if $C_{best} > \mathcal{T}$ then we repair the MMOs {m} one by one. For each repaired MMO, we check the consistency C_{new}, if any C_{new} is smaller than the best consistency score C_{best}, then we deem the MMO repair as successful and update the model $\mathcal{M}$ until we find the smallest C_{best} possible, which reflects the least amount of model inconsistency.

The procedure turns this problem into a search task which iteratively considers different changes to the model via predefined MMOs (e.g., increase range by 1 cell, increase range by 2 cells, decrease range by 1 cell, etc.). The goal of this search is to sufficiently reduce the inconsistency between expected and observed state trajectories. The procedure is adapted from Stern et al., 2022, "Model-Based Adaptation to Novelty for Open-World AI".

6 EXPERIMENT SETUP

Experimental evaluation was conducted on a machine with MacOS, an Intel Core i7 2.6 GHz 6 core processor with 16 GB 2667 MHz DDR4 memory.

Since the baseline agents have no concept of novelty, detection and false positives are both 0%. And the mission is a total failure 37% of the time, meaning the F-35 is destroyed, and the target is still fully functional. An interesting result is that the win percentage is relatively high, succeeding in the mission over half of the tries. This leaves an interesting final decision whether to execute the mission in the next battle or not, knowing the mission can either be a successful strike or end in total failure, without any reasonable explanation.

6.1 Novelties

In this mission a F-35 aircraft is given a task to strike a radar station device roughly 90,000 meters south-west of a SAM missile launcher

in enemy territory. A plan is generated using a model of the world given to it before the mission begins. In the scenario the agent can successfully plan a strike given its prior information is correct. However, when the information it assumes to be true is incorrect, it is unable to develop a successful plan and the F-35 aircraft is destroyed by the SAM site. Figure 3 shows the pre-novelty and post-novelty range of the SAM, the dotted yellow line shows the pre-novelty range, and the dotted purple line shows the post-novelty range.

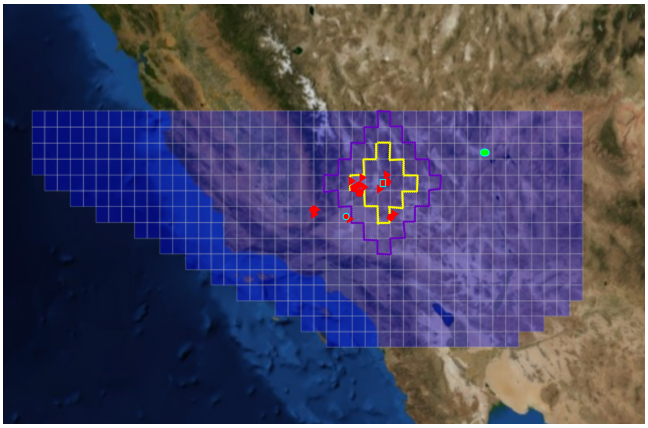

Figure 3: SAM Range Before And After Novelty

A 20 battle campaign $\prod$ with novelty φ injected in the second battle ($t_N = 2$) is ran, every battle it generates a plan using the agent internal model of the world. In the pre-novelty scenarios, the agent can successfully generate a plan to complete the mission. The agent search statistics is shown in Table 2.

After novelty injection, the SAM missile range is increased, using the old model, it is unable to develop a successful plan and the aircraft risks being destroyed by the SAM upon entering the new SAM range. Due to environment stochasticity described in the related work section, it takes the risk of entering the SAM without knowing it has and successfully executes the mission 15 times, and is destroyed 4 times post-novelty. The performance of novelty accommodation is shown in Table 1. Since the baseline agents have no concept of novelty, detection and false positives are both 0%. And the mission becomes a total failure 21% of the time, resulting in the F-35 aircraft being destroyed and the enemy target still functional. An interesting result is that the win % is very high, succeeding in the mission at a high rate. Perhaps making a human decision to execute the mission difficult knowing the statistics, as 79% of the time, it results in a mission success.

7 RESULTS

Of the 20 battle campaign $\prod$, the first battle the agent successfully completes the mission with no novelty. The second battle includes the missile range novelty, and the agent proceeds to complete the mission, the reason is due to environment stochasticity of high fidelity simulators mentioned in the related work section, the aircraft can unknowingly takes the risk of entering the SAM missile range area without knowing it has and successfully executes the mission sometimes, but is destroyed other times without knowing why. The agent performance of novelty detection and accommodation is shown in Table 1 using performance measurement described in [21].

For the fourth battle through the last, the consistency function C passes the threshold $\mathcal{T}$, the consistency score C becomes 1 because the distance between t_e and t_o is now 1 for losing the aircraft last battle, and repair is turned on. The SAM range is the numeric MMO {m} monitored in the repair R; the numeric value of the SAM range is adjusted to increase from 2 cells to 4 cells. The MMO change ΔV resulted in decreased consistency score C and the repair is deemed successful, it is also able to characterize the novelty by repairing "ss-weapon-range" from 2 to 4. With the repair, the agent will navigate around the risky area starting from the next battle after novelty is detected. As shown in figure 4, the agent is able to destroy the enemy target with a new route after repairing the PDDL+ model. The yellow solid line shows the pre-novelty route, while the purple solid line denotes the post-novelty route. The grey symbol of a bomb with dotted teal outlines represents the location where the F-35 fired the JDAM missile from, in both cases the F-35 aircraft proceeds to return to home base after firing the JDAM.

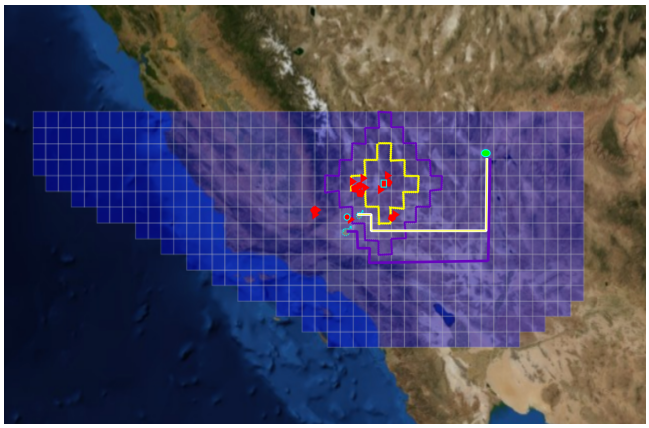

Figure 4: Agent Plan Before And After Novelty

The performance of the novelty accommodating agent is in Table 1. Of the 20 battles, compared to the baseline agent, our novelty detection false positive rate is still 0%. The false negative rate decreased to 5%. The pre-novelty win rate is 100%, and the post novelty win rate increases to 95%. In this tactical mission result, there are no complications, the agent completes the mission and is not destroyed. Novelty detection rate increases to 95%.

The novelty accommodating agent search statistics is shown in Table 3. The novelty accommodating plan explored less states but the plan is longer due to navigating around a longer range. While the low level primitive actions in AFSIM are still combined and abstracted, routing options are fixed to geo-coordinates defined as the centroids of each grid cell, and all the primitive firing actions are combined into one action for a range of one cell.

Table 1: Performance measurement of baseline and novelty accommodating agents

	Baseline	Hydra
False neg%	21	5
Win% post-Nov	79	95
Win% pre-Nov	100	100
Detection%	0	95
False pos%	0	0

Table 2: Pre-Novelty Agent Plan

Planning Time	Plan Length	States Explored
19.118	33	2632

Table 3: Novelty Accommodating Agent Plan

Planning Time	Plan Length	States Explored
18.312	41	2421

8 CONCLUSION AND FUTURE WORK

We demonstrated a prototype framework to utilize the Hydra AI system in a realistic high fidelity simulator that is being used to execute intricate military scenarios, and novelties that resemble open-real-world problems. Our results prove that model-based AI systems like Hydra can be adapted from smaller-scoped games with hypothetical game related novelties to more realistic applications.

The next step towards accommodating real open-world novelty is to extend this framework and model a richer domain including additional real-world novelties and complexity such as hidden enemy units, faulty sensors, unknown delays, different types of weapons, different environmental effects, different capabilities by enemy fighting units, complex routing for different mission objectives, and unknown civilian obstacles. Furthermore, in future scenarios, we plan to add more actions, agents, and goal complexity to the scenario, to allow the agent to explore more complex decisions and novelty detection, characterization, and accommodation opportunities.

ACKNOWLEDGMENTS

This research was sponsored by DARPA. The views contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. government.

REFERENCES

[1] Joshua Alspector. 2021. Representation Edit Distance as a Measure of Novelty. *arXiv preprint arXiv:2111.02770* (2021).

[2] WS Bell. 2015. *Joint Direct Attack Munition (JDAM).* Technical Report. US Air Force Hill AFB United States.

[3] Terrance Boult, Przemyslaw Grabowicz, Derek Prijatelj, Roni Stern, Lawrence Holder, Joshua Alspector, Mohsen M Jafarzadeh, Toqueer Ahmad, Akshay Dhamija, Chunchun Li, et al. 2021. Towards a unifying framework for formal theories of novelty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 15047–15052.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). arXiv:1606.01540 http://arxiv.org/abs/1606.01540

[5] Theresa Chadwick, James Chao, Christianne Izumigawa, George Galdorisi, Hector Ortiz-Pena, Elias Loup, Nicholas Soultanian, Mitch Manzanares, Adrian Mai, Richmond Yen, and Douglas S. Lange. 2023. Characterizing Novelty in the Military Domain. https://doi.org/10.48550/ARXIV.2302.12314

[6] James Chao, Wiktor Piotrowski, Mitch Manzanares, and Douglas S Lange. 2023. Top Gun: Cooperative Multi-Agent Planning. (2023). http://idm-lab.org/wiki/AAAI23-MAPF/index.php/Main/HomePage?action=download&upname=Paper_4.pdf

[7] James Chao, Jonathan Sato, Crisrael Lucero, and Doug S Lange. 2020. Evaluating Reinforcement Learning Algorithms For Evolving Military Games. *Convergence* 200, 44 (2020), 1.

[8] Peter D Clive, Jeffrey A Johnson, Michael J Moss, James M Zeh, Brian M Birkmire, and Douglas D Hodson. 2015. Advanced framework for simulation, integration and modeling (AFSIM)(Case Number: 88ABW-2015-2258). In *Proceedings of the international conference on scientific computing (CSC)*. The Steering Committee of The World Congress in Computer Science, Computer . . . , 73.

[9] Michael W Floyd, Justin Karneeb, Philip Moore, and David W Aha. 2017. A Goal Reasoning Agent for Controlling UAVs in Beyond-Visual-Range Air Combat.. In *IJCAI*, Vol. 17. 4714–4721.

[10] Maria Fox and Derek Long. 2006. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research* 27 (2006), 235–297.

[11] Chathura Gamage, Vimukthini Pinto, Cheng Xue, Matthew Stephenson, Peng Zhang, and Jochen Renz. 2021. Novelty Generation Framework for AI Agents in Angry Birds Style Physics Games. In *2021 IEEE Conference on Games (CoG)*. 1–8. https://doi.org/10.1109/CoG52621.2021.9619160

[12] Sharath Girish, Saksham Suri, Sai Saketh Rambhatla, and Abhinav Shrivastava. 2021. Towards Discovery and Attribution of Open-World GAN Generated Images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 14094–14103.

[13] Larry Holder. 2022. *"HOLDERLB/WSU-Sailon-ng"*. "https://github.com/holderlb/WSU-SAILON-NG"

[14] Mayank Kejriwal and Shilpa Thomas. 2021. A multi-agent simulator for generating novelty in monopoly. *Simulation Modelling Practice and Theory* 112 (2021), 102364. https://doi.org/10.1016/j.simpat.2021.102364

[15] Robert H Kewley and Mark J. Embrechts. 2002. Computational military tactical planning system. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 32, 2 (2002), 161–171.

[16] Pat Langley. 2020. Open-world learning for radically autonomous agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 13539–13543.

[17] Nianzu Ma, Alexander Politowicz, Sahisnu Mazumder, Jiahua Chen, Bing Liu, Eric Robertson, and Scott Grigsby. 2021. Semantic Novelty Detection in Natural Language Descriptions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 866–882. https://doi.org/10.18653/v1/2021.emnlp-main.66

[18] Philip A Merkel, Paul E Lehner, Roger A Geesey, and John B Gilmer. 1991. *Automated planning with special relevance to associate systems technology and mission planning*. Technical Report. BDM INTERNATIONAL INC ARLINGTON VA.

[19] Faizan Muhammad, Vasanth Sarathy, Gyan Tatiya, Shivam Goel, Saurav Gyawali, Mateo Guaman, Jivko Sinapov, and Matthias Scheutz. 2021. A novelty-centric agent architecture for changing worlds. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 925–933.

[20] David J Musliner, Michael JS Pelican, Matthew McLure, Steven Johnston, Richard G Freedman, and Corey Knutson. 2021. OpenMIND: Planning and Adapting in Domains with Novelty. (2021).

[21] Vimukthini Pinto, Jochen Renz, Cheng Xue, K Doctor, and D Aha. 2020. Measuring the Performance of Open-World AI Systems. (2020).

[22] Wiktor Mateusz Piotrowski, Maria Fox, Derek Long, Daniele Magazzeni, and Fabio Mercorio. 2016. Heuristic planning for PDDL+ domains. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

[23] Stuart Russell and Peter Norvig. 2021. Artificial intelligence: a modern approach, global edition 4th. *Foundations* 19 (2021), 23.

[24] Roni Stern, Wiktor Piotrowski, Matthew Klenk, Johan de Kleer, Alexandre Perez, Jacob Le, and Shiwali Mohan. 2022. Model-Based Adaptation to Novelty for Open-World AI. In *Proceedings of the ICAPS Workshop on Bridging the Gap Between AI Planning and Learning*.

[25] Chris Wiegand. 2018. F-35 air vehicle technology overview. In *2018 Aviation Technology, Integration, and Operations Conference*. 3368.

[26] David Edward Wilkins and Roberto Desimone. 1993. *Applying an AI planner to military operations planning*. Citeseer.