GPT-in-the-Loop: Supporting Adaptation in Multiagent Systems

Nathalia Nascimento, Paulo Alencar, Donald Cowan
David R. Cheriton School of Computer Science
University of Waterloo (UW)
Waterloo, Canada
{nmoraesd, palencar, dcowan} @uwaterloo.ca

2023 IEEE International Conference on Big Data (BigData) | 979-8-3503-2445-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/BigData59044.2023.10386490

***Abstract*—This paper introduces the 'GPT-in-the-loop' approach, which seeks to investigate the reasoning capabilities of Large Language Models (LLMs) like Generative Pre-trained Transformers (GPT) within multiagent systems (MAS). Moving beyond traditional adaptive approaches that generally require long training processes, our framework employs GPT-4 to enhance problem-solving and explanation skills. To explore this approach, we apply it to a smart streetlight application in the Internet of Things (IoT) context, wherein each streetlight is controlled by an autonomous agent equipped with sensors and actuators, tasked with creating an energy-efficient lighting system. With the integration of GPT-4, these agents have shown enhanced decision-making and adaptability, without necessitating prolonged training. We compare this approach with both conventional neuroevolutionary methods and manually crafted solutions by software engineers, underscoring the potential of GPT-driven behavior in multiagent systems. It is important to note that these comparisons are preliminary, and further, more extensive testing is critical to determine the approach's applicability across a wider range of MAS scenarios. Structurally, the paper delineates the incorporation of GPT into the agent-driven Framework for the Internet of Things (FIoT), details our proposed GPT-in-the-loop approach, presents comparative results within the IoT setting, and concludes with insights and prospective future directions.**

***Index Terms*—GPT-in-the-loop, LLM-in-the-loop, Multiagent system (MAS), self-adaptation, Generative pre-trained transformer (GPT)**

I. Introduction

Exploratory investigations are currently underway to harness the reasoning capabilities of Generative Pre-trained Transformers (GPT) for practical applications. Recent studies [1]–[10] indicate that large language models, especially those exceeding 100 billion parameters, are showcasing emergent reasoning abilities. Webb et al. [1] demonstrated that models like GPT-3 might match or even outdo human reasoning in certain tasks—a trajectory GPT-4 seems set to follow. Further supporting this, [2] reveals that a "chain of thought" approach can significantly enhance reasoning in these models, suggesting new methods to utilize their reasoning prowess in real-world scenarios.

Conversely, in the multiagent domain, developing autonomous systems, especially agents that autonomously develop their skills through environment interactions, is an ambitious scientific endeavor [11]. These agents aim to expand their behavioral repertoire in an open-ended manner. A major thrust is enabling them to employ world models, using common sense knowledge akin to humans, to enhance their performance [11]. Such knowledge can be gleaned via self-supervised learning, allowing agents to mentally plan and reason. While neuroevolutionary approaches offer potential solutions [12]–[14], refining neural networks for performance enhancement is time-intensive, costly, and complex, especially in real-time settings with physical agents. In addition to problem-solving skills, the agents should also explain their decisions [15].

Bridging these two domains, the concept of "GPT-in-the-loop" emerges as a promising approach. By leveraging the advanced reasoning capabilities of GPT models within the loop of agent decision-making, there's potential to address the challenges in multiagent systems more efficiently. This fusion could harness GPT's inherent adaptability and reasoning prowess, potentially reducing reliance on long training processes that are usual to adaptive approaches [11]. Inspired by human-in-the-loop approaches [16], our proposal defines novel GPT and multiagent system interactions. We delineate three interaction types: active MAS, where traditional algorithms guide agents and GPT elucidates results, exploring its explainability skills; interactive MAS, fostering tighter collaboration between GPT reasoning and the MAS; and MAS teaching, where GPT guides the MAS adaptation. This work offers examples of each interaction, showcasing one in detail.

Building upon the FIoT framework for adaptable Internet of Things (IoT) applications [13], we incorporate the "GPT-in-the-loop" methodology. To create self-adaptive agents, FIoT supports the use of different decision-making engines, like neural networks, state machines, and if-else statement; as the use of different adaptative processes, like evolutionary algorithms, backpropagation, and reinforcement learning. FIoT's flexibility in decision-making engines and adaptive processes make it conducive for GPT integration. This flexibility paves the way for GPT to augment reasoning or adaptive functions. For instance, within an interactive MAS setup, GPT can amplify decision-making, aiding agents in outputs and interactions. In MAS teaching, GPT might guide the adaptive process or even dictate the decision-making engine entirely, adjusting agent behaviors based on environmental feedback.

Furthermore, we have applied the GPT-in-the-loop model to smart streetlights, a benchmark MAS application [17].

Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.

In this scenario, agents, equipped with sensors, actuators, and a neural network, evolve to develop a communication system and behavior that optimizes energy while ensuring adequate lighting. As this study [17] also assessed 14 software engineers' solutions to the same challenge, it allows us to perform a direct comparison between the neuroevolutionary approach, the engineers' solutions, and our GPT-in-the-loop method.

The paper is structured as follows: Section 2 delves into the GPT and FIoT background. Section 3 details our primary contribution, the GPT-in-the-loop approach. Section 4 offers performance results and comparisons within the IoT MAS scenario. We conclude in Section 5.

II. Background

A. LLM and GPT

Large Language Models (LLMs) and Generative Pre-trained Transformers (GPT) are integral parts of AI's Natural Language Processing (NLP) realm. While LLM is a broad category encompassing models that predict word sequences and can be used for various tasks such as text generation and translation, GPT, developed by OpenAI [18], is a specific LLM type. GPT, renowned for generating text akin to human writing, undergoes extensive pre-training before fine-tuning for specialized tasks. In essence, GPT is a subclass of LLMs, but not all LLMs are GPT models. Other prominent LLM examples include BERT, RoBERTa, and XLNet.

GPT (Generative Pre-trained Transformer) is rooted in the Transformer neural network design [19], [20]. Representing breakthroughs in natural language processing, GPT, especially in its advanced iterations like GPT-4, utilizes a deep architecture of many layers of these transformers. A GPT solution comprises several key components, such as a pre-trained neural network model, a fine-tuning component to improve the model for specific tasks, an inference engine that uses the fine-tuned GPT model to generate responses or predictions (i.e. the inference engine feeds input data into the model and processes the model's output), and data pipeline that handles the flow of data in and out of the model [20].

B. FIoT: Framework for Self-Adaptive IoT Multiagent Systems

The Framework for the Internet of Things (FIoT) [13], [21] is a software framework designed for building control systems for self-operating agents through learning or rule-based methods. Utilizing FIoT results in a Java software element pre-loaded with features for recognizing autonomous entities, assigning control, developing software agents, collecting device data, and ensuring agent-device interactions.

FIoT's features can be customized based on the application's needs. These include: 1) a control unit, ranging from basic if-else conditions to neural networks or preset state machines; 2) a controller adaptation method using techniques like reinforcement learning or genetic algorithms; and 3) a mechanism to evaluate decision-making processes in controlled devices.

There are two primary agents in FIoT: AdaptiveAgent and ObserverAgent. The former oversees IoT devices and uses the controller for decision-making. Its foundation is the MAPE-K loop [22], an esteemed model for enhancing system autonomy. It perceives, acts, and reasons, tailoring outputs based on the chosen decision system. Meanwhile, the ObserverAgent gauges overall agent activity and can refine the control system adopted by IoT agents.

III. Approach: GPT-in-the-loop

Drawing inspiration from human-in-the-loop methodologies [16], our proposition delineates novel interactions between GPT and multiagent systems. We propose three main interaction modes:

- **Active MAS:** Traditional algorithms drive agents while GPT clarifies outcomes.
- **Interactive MAS:** This encourages a more integrated collaboration between the GPT's reasoning and the MAS, which has been our primary focus in this work as depicted in Figure 1.
- **MAS Teaching:** Here, the GPT directs the MAS adaptation.

In the interactive MAS model, GPT shapes the decision-making engine of the agent. This engine processes inputs, generates outcomes, and influences the manner in which the agent engages with its environment, which in turn impacts application performance. Feedback from these engagements can re-engage the GPT, leading to refinements in agent behaviors.

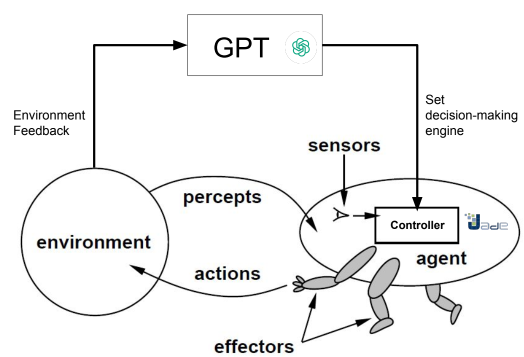

Fig. 1. GPT-in-the-loop: GPT crafts the decision-making engine for the agent, drawing from environmental feedback.

While the interactive MAS mode stands at the heart of our research, we integrated GPT with FIoT [1]. This framework paves the way for probing diverse interaction forms. It permits a complete overhaul of the agents' decision-making engine or steers the evolution/training process orchestrated by the ObserverAgent.

Figures 2 and 3 illustrate the seamless extension of FIoT to accommodate the GPT-in-the-loop model, tapping into both the AdaptiveAgent's controller and the ObserverAgent's adaptation process. Notably, both the decision-making controller and the adaptive procedure are flexible points in the framework. This allows for varied runtime implementations,

[1]The source for FIoT framework integrated to GPT API is available at: Github (accessed on Nov 01 2023)

Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.

so long as class signatures (parameters, inputs, and outputs) remain consistent. For instance, environmental feedback can prompt GPT to craft a new controller for agents.

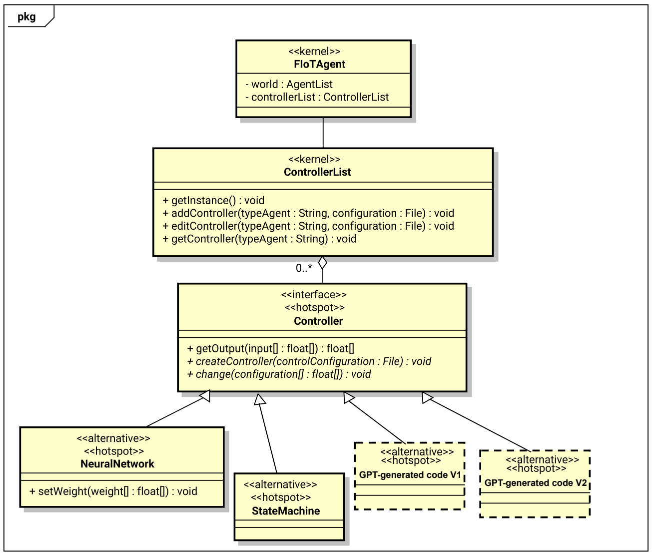

Fig. 2. Augmenting FIoT to empower agents with decision-making abilities using GPT-crafted code.

IV. Evaluation: Illustrative Examples

This section provides an evaluation of our proposed GPT-in-the-loop framework, as delineated in the introduction with the definitions of Active MAS, Interactive MAS, and MAS Teaching. We assess the efficacy of this framework through two agent application instances within a smart streetlight scenario and provide a conceptual outline for a future MAS teaching application.

A. Application Scenario: Smart Streetlights

The smart streetlight scenario serves as a testbed for both the Active and Interactive MAS evaluations, providing a context for the application of these approaches. It is within this scenario that the functionality and potential of the GPT-in-the-loop framework are examined and the groundwork for future MAS Teaching explorations is laid.

In our experiment, we replicated the streetlight scenario from [17] using the FIoT framework [2]. The goal was to create autonomous streetlights balancing energy conservation with effective illumination, ensuring individuals could navigate their paths seamlessly. These streetlights, equipped with sensors and communicative tools, had three core functions: data collection, decision-making, and action execution. The focus of this experiment was on the decision-making aspect.

Each streetlight was outfitted with ambient light and motion sensors, a light intensity regulator, and wireless communication capabilities, as depicted in Figure 4.

[2]The streetlight application code is available at: Github (accessed on Nov 01 2023)

Each streetlight performs three tasks every second: data collection, decision-making, and action enforcement. The first task involves collecting data on people flow, ambient light, neighboring streetlight data, and the current light status (sensor activation level and previous listeningDecision output value). The second task involves scrutinizing the collected data to make action decisions. The final task is the enforcement of actions, wherein the values of three output variables are set: (i) listeningDecision, enabling the streetlight to accept signals from adjacent streetlights in the next cycle; (ii) wirelessTransmitter, a signal value transmitted to neighboring streetlights; and (iii) lightDecision, controlling the light's OFF/DIM/ON functions. For a more detailed description of the application scenario, refer to [17].

Given that the data collection and action enforcement tasks were pre-implemented, the main challenge of this application scenario is to devise a solution for decision-making. The original study utilized a three-layer neural network, evolved through a genetic algorithm, to automate the streetlights' decision rules. Software engineers also tackled the challenge, of developing decision-making solutions. They were presented with the same simulated scenario, facilitating a comparison of human-devised solutions with the automated neural network method. Subsequently, these solutions were tested in an expanded environment. This second phase aimed to assess whether the decision-making module, originally designed for the first scenario, could be effectively reused in a different environment.

To devise a solution, software engineers received a detailed problem statement outlining the application scenario and the objectives for the street light agents. [3]. The described problem served as the initial prompt for the GPT in the interactive MAS methodology.

B. Interactive MAS Evaluation

The Interactive MAS approach fosters a symbiotic partnership between GPT-4's reasoning processes and the MAS's functionality. This bidirectional interaction, which forms the crux of our investigation, is visualized in Figure 5.

Figure 5 depicts the GPT-in-the-loop methodology applied to the streetlight control scenario, where autonomous agents, represented by streetlights, are tasked with making energy-efficient decisions. The GPT API is used to generate the agents' decision-making code based on a initial problem description. This code is then loaded into the agents within a simulated multiagent system (MAS), where each streetlight is an autonomous agent.

The process begins with a user providing the GPT API with the same problem statement that was provided to the software engineers in [17]. If the generated code does not meet the required fitness score or after a set number of iterations, an automatic prompt generation cycle enhances the solution by integrating feedback from the previous attempts. This loop

[3]This statement was provided in Portuguese and can be found here: (Google Form - accessed on Nov 01 2023)

Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.

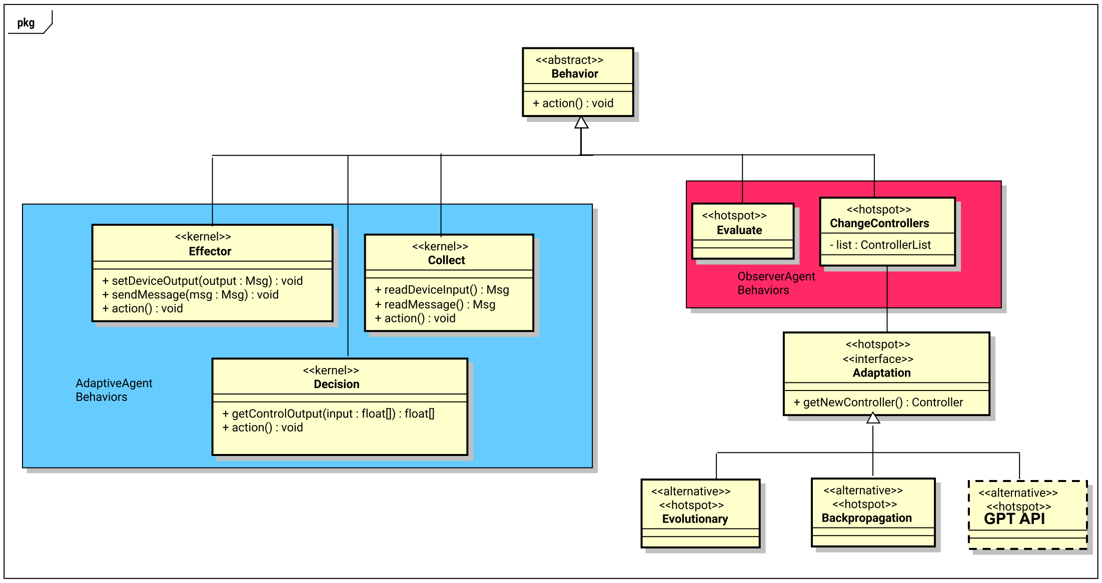

Fig. 3. Elevating FIoT to incorporate GPT as a potential adaptive strategy for the Observer Agent.

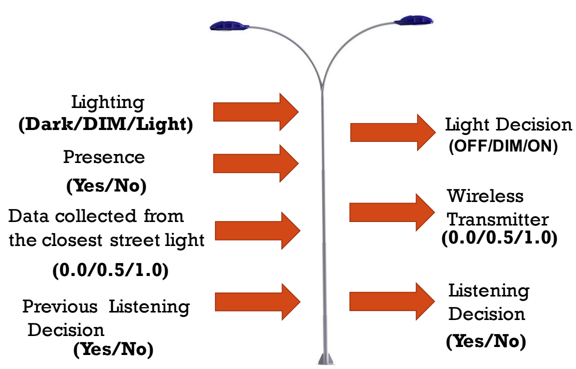

Fig. 4. Data gathered and manipulated by streetlights [17].

continues until the solution meets the fitness threshold or reaches the iteration limit. Application fitness is calculated based on energy consumption, completion of routes by individuals, and total travel time, ensuring that the system's performance aligns with real-world efficiency metrics.

For instance, we set the fitness goal of 62, considering the best fitness value presented in [17]. Once an acceptable solution is found, it is tested in the expansive environment to check if the decision-making module can adapt to different scenarios effectively. Conclusively, we set the GPT-in-the-loop results as a benchmark, juxtaposing them against the top solutions from the neuroevolutionary algorithm and the best software engineer participant.

To facilitate a clear comparison between the GPT and original (neuroevolutionary) methods, Table I showcases the application of the Streetlight Control case study using a neuroevolutionary approach, highlighting the flexible points of the FIoT framework. Conversely, Table II delineates the implementation of the Streetlight Control application through the GPT-in-the-loop-based approach, capitalizing on the adaptability of the FIoT framework. Both tables aim to provide a foundation for evaluating the efficacy of each solution within the same application context.

TABLE I
IMPLEMENTING FIOT FLEXIBLE POINTS TO SYNTHESIZE STREETLIGHT CONTROLLERS USING AN ML-BASED APPROACH [17].

FIoT Framework	Light Control Application
Controller	Three Layer Neural Network
Making Evaluation	Collective Fitness Evaluation: the solution is evaluated based on the energy consumption, the number of people that finished their routes after the simulation ends, and the total time spent by people to move during their trip
Controller Adaptation	Evolutionary Algorithm: Generate a pool of candidates to represent the neural network parameters

1) Results and Discussion: The GPT-in-the-loop approach required three iterations to reach a fitness score of 62 in the first scenario [4]. Comparatively, the original evolutionary approach underwent 200 generations, with each generation undergoing 50 interactions with the environment. This section details the solutions GPT proposed for the streetlight controllers in each iteration, elucidating the adaptive changes influenced by GPT's explanations. Ultimately, we compared the outcomes of this innovative method against those of neuroevolution and human-generated solutions.

[4]The behavior programming code generated by GPT API on each iteration is available at:
Github (accessed on Nov 01 2023)

Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.

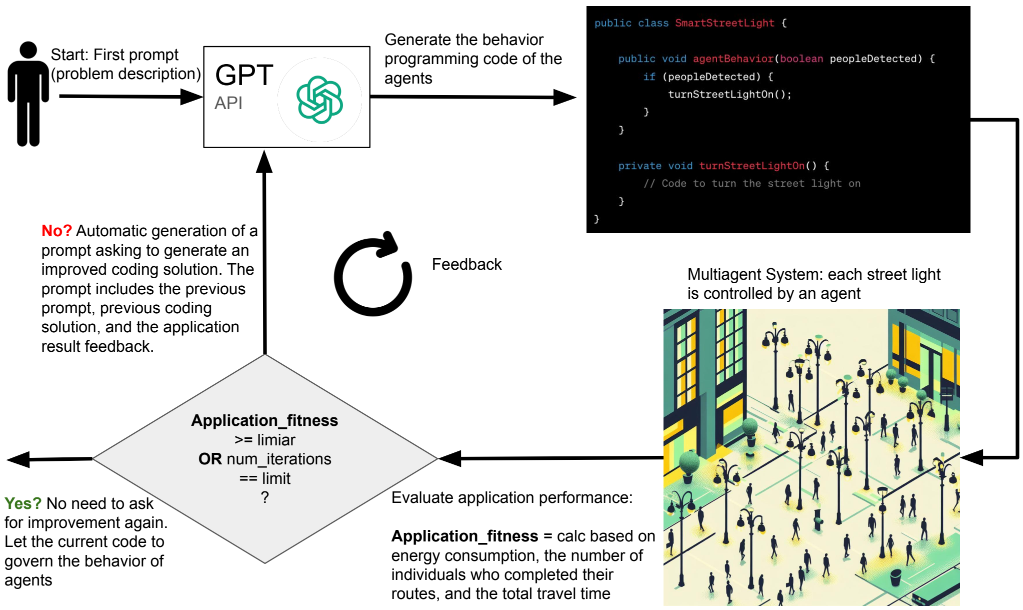

Fig. 5. Overview of the GPT-in-the-loop approach applied to the street light application scenario, representing an interactive MAS approach.

TABLE II
DETAILED IMPLEMENTATION OF FIOT FLEXIBLE POINTS TO SYNTHESIZE STREETLIGHT CONTROLLERS USING THE GPT-IN-THE-LOOP APPROACH.

FIoT Framework	Light Control Application
Controller	GPT-based Decision Engine: Use if-else statement controllers optimized for the given scenario's constraints and goals
Making Evaluation	Iterative Fitness Evaluation: the solution iterates until it exceeds a fitness score of 62, evaluating based on energy consumption, the number of people that complete routes, and the cumulative time of people's journeys
Controller Adaptation	GPT-in-the-loop: GPT engages in interactive loops, refining its if-else controllers based on environment feedback until the desired fitness level is reached

In the initial iteration, the problem description presented to the software engineers in [17] served as the prompt. For every iteration, GPT generated an 'if-else statement' controller accompanied by a rationale for the decision.

The code, as shown in Figure 6, emerged from the first iteration. The foundational idea behind this solution was: 1) Ambient Light Detection: It's the sole strategy highlighting that poles will perpetually be in "listening" mode, regardless of external lighting conditions; 2) Wireless Communication: Stresses the significance of interaction with the preceding pole and adjusting illumination based on that communication; 3) Energy Conservation: Sets a distinct condition whereby, in the absence of motion detection for a predetermined cycle count, the light's intensity diminishes.

The first iteration resulted in an average fitness of 29.49. Given its inadequacy, GPT was re-engaged to produce an enhanced solution. For the second iteration, the environment's outputs, such as energy consumption and trip time, were incorporated into the GPT prompt. The code from the second iteration is showcased in Figure 7. Contrary to the first solution, which emphasized persistent ambient light listening, previous pole communication, and energy preservation, the second solution accentuates anticipatory lighting influenced by adjacent movements and subdued illumination during extreme darkness.

The second iteration notably improved the fitness value, achieving 61.2. However, since this score remained below the target, GPT produced another controller, depicted in Figure 8. Unlike the solution from the second iteration, this new approach concentrates on curtailing data transmissions by sending signals exclusively upon initial motion detection, determining listening necessity based on present light status, and integrating adaptive dimming.

The integration of environmental feedback into successive GPT iterations consistently led to performance improvements. Table III displays the fitness outcomes across three iterations for scenario 1, whereas Table IV presents the outcomes when the decision algorithms were implemented in a more intricate environment for scenario 2. In both scenarios, the third solution proposed by GPT outperformed its predecessors. When juxtaposed with the optimal outcome derived from the neuroevolution method, the solution offered by GPT achieved a superior fitness score in both scenarios. In scenario 1, one participant managed to devise a solution with a slightly better

Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.

Fig. 6. GPT's decision-making engine from the first iteration.

```
@Override
public double[] getOutput(double[] input) {
    // Input variables
    double lightingSensor = this.getLightingSensor();
    double motionSensor = this.getMotionSensor();
    double wirelessReceiver = (this.getPreviousListeningDecision() == 1.0) ? this.getWirelessReceiver() : 0.0;
// If not listening, ignore the value.
    // Decision variables
    double lightDecision = 0.0;  // By default, we keep it off
    double wirelessTransmit = 0.0;  // By default, we don't transmit anything
    double listeningDecision = 1.0;  // By default, we always listen to the surrounding posts
    // If it's day, we keep the light off
    if (lightingSensor == 1.0) {
        lightDecision = 0.0;
    } else {
        // If we detect movement
        if (motionSensor == 1.0) {
            lightDecision = 1.0;
            wirelessTransmit = 1.0;  // We alert the surrounding posts about the detected movement
        }
        // If the neighboring post detected movement and we are listening
        else if (wirelessReceiver > 0.0) {
            lightDecision = 0.5;  // We partially light up to prepare the path
        }
    }
    // Action decisions
    this.setLightDecision(lightDecision);
    this.setWirelessTransmitter(wirelessTransmit);
    this.setListeningDecision(listeningDecision);
    return this.actuador;
}
```

Fig. 7. GPT's decision-making engine from the second iteration (input and output settings excluded).

```
@Override
public double[] getOutput(double[] input) {
   /*...*/
    double lightDecision = 0.0;
    double wirelessTransmit = 0.0;
    double listeningDecision = 1.0;  // Always listening to optimize the person's path
    // If motion is detected
    if (motionSensor == 1.0) {
        lightDecision = 1.0;  // Turn on fully for the passerby
        wirelessTransmit = 1.0;  // Notify the nearby posts to prepare the path
    // If a neighboring post reports movement
    } else if (wirelessReceiver == 1.0) {
        lightDecision = 0.5;  // Partially light up to anticipate the passerby's arrival
    // Energy saving without compromising safety
    } else if (lightingSensor == 0.0) {
        lightDecision = 0.5;  // Keep partially lit if it's too dark, to ensure some level of safety
    }
    /*...*/
    return this.actuador;
}
```

fitness score than that of GPT's. However, this solution faltered in the second scenario. Evaluating GPT's performance against the most successful participant-driven solutions in the second scenario, GPT's solution was unrivaled.

In the quest for optimized streetlight controllers, GPT's iterative approach showcased notable adaptability and improvement. The model's ability to integrate environment feedback between iterations culminated in solutions competitive with human and neuroevolution strategies. Moreover, GPT's intrinsic explainability, as evident in its generated 'if-else statements' and accompanying rationale, offers valuable insights for users, bridging the gap between automation and human understanding.

C. Active MAS Evaluation

In the Active MAS setup, standard algorithms guide the streetlight agents, while GPT-4 helps explain their actions afterwards. We tested this method on the smart streetlights application scenario to see how well GPT-4 could clarify the reasoning behind the automated decisions made by these agents.

Originally, for the smart streetlight, [17] used an evolutionary approach with neural networks to evolve the best decision-making behaviors in agents. This involves an algorithm fine-tuning the neural network inside each agent to achieve optimal performance.

We used a simple diagram, shown in Figure 9, to explain how agents develop and adapt over time using this evolution-

Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.

Fig. 8. GPT's decision-making engine from the third iteration (input and output settings excluded).

```
@Override
    public double[] getOutput(double[] input) {
        /*...*/
        if (motionSensor == 1.0) {
            lightDecision = 1.0;
            if (this.getPreviousListeningDecision() != 1.0) {
                wirelessTransmit = 1.0; // Transmit motion detection once
                listeningDecision = 0.0; // No need to listen when we already detect motion
            }
        } else if (wirelessReceiver == 1.0) {
            lightDecision = 0.5;
            if (this.getPreviousListeningDecision() != 1.0) {
                wirelessTransmit = 0.5; // Inform neighbors about anticipatory lighting
            }
        } else if (lightingSensor == 0.0) {
            lightDecision = 0.5;
        }
        /*...*/
        return this.actuador;
    }
```

TABLE III
PERFORMANCE COMPARISON OF GPT ITERATIONS, BEST NEUROEVOLUTION SOLUTION, AND BEST PARTICIPANT'S SOLUTION IN THE FIRST SCENARIO.

Solution	Energy	People	TotalFTrip	Fitness
GPT (iteration 1)	4.03	66.66	59.25	29.49
GPT (iteration 2)	15.02	100	54.62	61.2
GPT (iteration 3)	11.92	100	54.62	62.44
Best neuroevolution's solution	8.1	100	62.03	59.53
Best participant's solution	9.46	100	55.55	62.88

TABLE IV
PERFORMANCE COMPARISON OF GPT ITERATIONS, BEST NEUROEVOLUTION SOLUTION, AND BEST PARTICIPANT'S SOLUTION IN THE SECOND SCENARIO.

Solution	Energy	People	TotalFTrip	Fitness
GPT (iteration 1)	2.08	66.66	48.51	36.72
GPT (iteration 2)	11.29	100	41.10	70.81
GPT (iteration 3)	9.76	100	41.10	71.42
Best neuroevolution's solution	8.46	100	46.29	68.83
Best participant's solution	50.52	100	38.14	56.9

ary method.

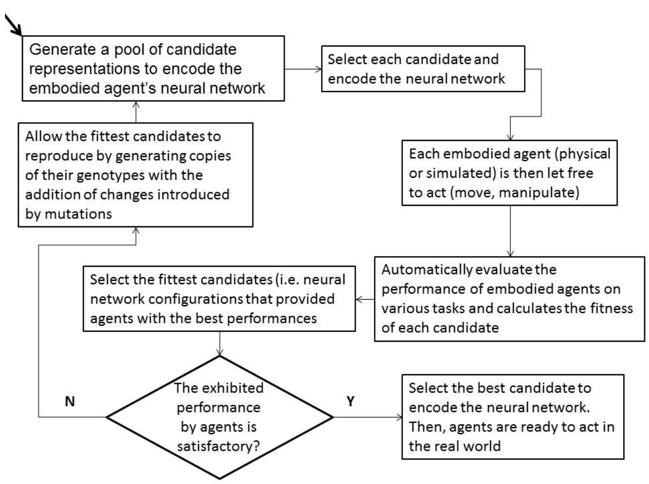

Fig. 9. Flowchart: Evolution of agents.

Such evolutionary approaches can lead to unexpected and beneficial behaviors like agent communication. Nevertheless, a significant challenge arises in interpreting these newly developed behaviors, as the traditional evaluation of these algorithms can be complex and opaque. To alleviate this, we aim to investigate the use of GPT-4 to elucidate the outcomes of the evolutionary process, adding a layer of interpretability to the traditionally cryptic evolutionary algorithms.

The evolutionary algorithm for this application scenario consisted on 20 generations, each generation with a population of 200 individuals. Each individual (neural network weights) was tested 10 times and its fitness was an average of these executions. During the training process, the algorithm evaluates the weight possibilities based on the energy consumption, the number of people that finished their routes after the simulation ends, and the total time spent by people to move during their trip. Therefore, each weights set trial is evaluated after the simulation ends.

Typically, the evolution of agent behavior is tracked by plotting the average fitness across generations, which highlights notable performance improvements or declines. From these plots, we can select the most promising individual neural networks and delve into their specific behaviors and

Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.

characteristics. The traditional approach involves scrutinizing these points on the plot for slight fitness variations, as depicted in Figure 10, to identify significant behavioral shifts.

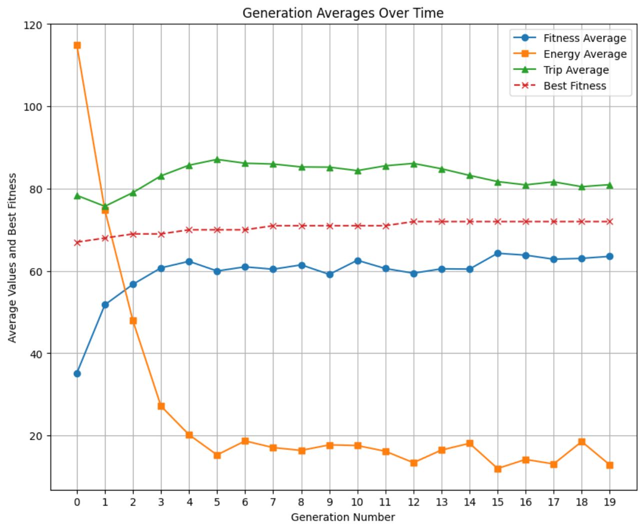

Fig. 10. Simulation results - Average fitness from each generation

To advance beyond traditional methods and enhance our understanding, we've employed GPT-4 to interpret the data from each generation, encompassing fitness scores, energy use, travel times, and the genetic makeup (neural network weights) of the top-performing individuals. We directed GPT-4 to analyze the evolution across generations, identify key genetic modifications, and elucidate the specific objectives targeted by each alteration. For this purpose, we used the following prompt: “Analyze the following generations, describing the changes that were made from one generation to the other, describing the main changes on the genetics of best individuals from one generation to the other. Describe the specific objectives that each change may address.”

1) Results and Discussion: Upon presenting the generation data to GPT-4, it generated explanations which were then compared to the chart in Figure 10 and the genes (weights) for the best individuals in Table V. GPT-4 highlighted a marked decrease in energy use from the second generation, linking this to the agents prioritizing light sensor data for decisions on lighting. By the third generation, a minor fitness improvement suggested a rebalancing of light sensor reliance, implying a search for an optimal balance between ambient light conditions and safety. The fourth generation saw a shift towards valuing motion sensor data, leading to further improvements. Post the fifth generation, GPT-4 identified a plateau in performance and behavioral changes, indicating a local optimization had been reached.

D. GPT Output

```
Changes from Generation 0 to Generation 1

Fitness Average: Increased from 35.16 to 51.78
Energy Average: Decreased from 114.99 to 74.78
Best Fitness: Increased from 67.0 to 68.0
Best Genes: Mostly unchanged except for
    lightSensor value, which increased from
    2.7 to 4.6.
Objectives Addressed: The increase in the
    fitness average indicates overall
    improvement. The reduced energy average
    suggests better energy efficiency. The
    genes show that more weight was given to
    the light sensor, potentially improving
    pedestrian safety by better adjusting to
    natural light conditions.
```

```
Changes from Generation 1 to Generation 2

Fitness Average: Slight increase
Energy Average: Significant decrease
Best Fitness: Increased to 69.0
Best Genes: Reduced weight for lightSensor
    from 4.6 to 2.7, while also reducing the
    fourth gene from 3.6 to 2.8.
Objectives Addressed: Again, the reduction in
    energy shows an emphasis on efficiency.
    The genes suggest a rebalancing, possibly
    to not overcompensate for ambient light
    and to maintain safety.
```

```
Changes from Generation 2 to Generation 3

Fitness Average: Increased
Energy Average: Dropped significantly
Best Genes: The fifth gene changed from -1.8
    to 3.8.
Objectives Addressed: A big drop in energy
    average shows excellent efficiency gains.
    The fifth gene's change likely influences
    the motion sensor, possibly optimizing for
     pedestrian safety.
```

```
Changes from Generation 3 to Generation 4

Fitness Average: Increased slightly
Energy Average: Decreased
Best Fitness: Increased to 70.0
Best Genes: Completely overhauled.
Objectives Addressed: A drastic change in
    genes suggests that a more efficient
    configuration was found, perhaps by
    allowing the lights to communicate more
    effectively or respond better to
    environmental variables.
```

```
Changes from Generation 4 to Generation 10

Fitness Average: Mostly stable with slight
    fluctuations
Energy Average: Generally declining
Best Fitness: Topped at 71.0 in Generation 7
    and stayed there
Best Genes: More stability and minor changes.
Objectives Addressed: After the drastic
    overhaul in Generation 4, subsequent
    generations seem to focus on fine-tuning.
    The stability in the best fitness and
    genes indicates that the algorithm is
    possibly reaching a local optimum.
```


Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.

TABLE V
BEST INDIVIDUALS FROM SIGNIFICANT GENERATIONS

Generation	Weight Values (W1 - W14)													
	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
0	4.9	4.2	2.7	3.6	-1.8	3.5	-0.4	4.8	3.2	3.1	-0.8	-1.6	-1.4	-0.2
1	4.9	4.2	4.6	3.6	-1.8	3.5	-0.4	4.8	3.2	3.1	-0.8	-1.6	-1.4	-0.2
2	4.9	4.2	2.7	2.8	-1.8	3.5	-0.4	4.8	3.2	3.1	-0.8	-1.6	-1.4	-0.2
3	4.9	4.2	4.6	3.6	3.8	3.5	-0.4	4.8	3.2	3.1	-0.8	-1.6	-1.4	-0.2
4	-2.1	4.2	-2.4	-1.5	-0.7	2.6	4.2	-1.6	4.7	3.8	-1.4	3.3	-0.8	-2.2
7	-2.4	4.2	-2.4	-1.5	-0.7	2.6	4.5	-1.6	4.7	3.8	-1.4	3.3	-0.8	-1.7
10	-2.4	4.2	-2.4	-1.5	-0.7	2.6	4.5	-1.6	4.7	3.8	-1.4	3.3	-0.8	-1.7

E. Prospective MAS Teaching Evaluation

We introduce a preliminary application for the MAS Teaching instance. In this prospective model, GPT-4 assumes a directive role, guiding the evolution and adaptation of the MAS. Although not yet tested, this approach aims to investigate GPT-4-driven learning and adaptation process for MAS, promising for future investigations into autonomous agent training and development.

Venturing beyond our preliminary framework, our ambition is to validate the GPT-in-the-loop approach in a spectrum of applications, especially when integrated with realistic robotics frameworks like Evorobot [23] and Webots [24]. Such platforms enable the deployment of neural networks sculpted by evolutionary techniques.

The domain of evolutionary robotics unravels complex challenges, a notable one being the food foraging task [25]. Here, agents are tasked with distinguishing nourishing food sources from harmful ones, adeptly navigating environmental intricacies for optimal survival. In this setup, agents traverse a dynamic landscape, reliant on a singular light sensor, to ascertain the edibility of proximate food. Represented in alternating colors of black and white, the safety of the food keeps shifting, mandating constant adaptability. Agents face a binary choice: to consume or avoid the food, within a given time frame.

Figure 11 depicts our conceptualization of GPT-in-the-loop within this distinct application setting. Here, the graphic portrays a MAS teaching interaction: while agents predominantly adhere to a conventional evolutionary path, GPT plays a supportive role in their evolution.

Embedding GPT within the evolutionary paradigm offers captivating prospects. GPT, transcending its observational role, can proactively shape the evolutionary trajectory. This encompasses guiding individual selection, fine-tuning genetic algorithms, and pinpointing ideal neural network configurations. Incorporating GPT's analytical prowess with evolutionary strategies could potentially evolve solutions that are not only optimal but also explainable.

V. CONCLUSION AND FUTURE HORIZONS

The synergy between Large Language Models (LLMs) such as GPT-4 with multiagent systems holds intriguing potential for transforming autonomous adaptability and interaction.

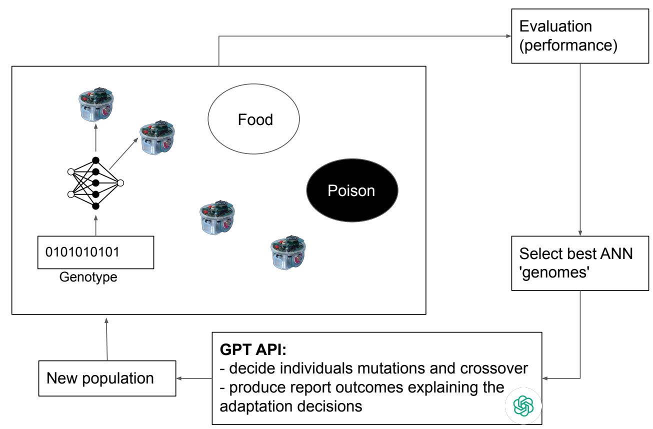

Fig. 11. GPT-in-the-loop: GPT supporting the evolutionary process.

Our study tentatively navigates the potential benefits this integration may offer. The 'GPT-in-the-loop' method is an exploratory step toward understanding how the incorporation of LLMs could potentially augment problem-solving capabilities within a dynamic environment. By embedding LLMs into agents, we anticipate a dual advantage: the potential for enhanced reasoning capabilities within individual agents and the prospect of streamlined communication within a complex multiagent system.

Moreover, the ability of GPT to articulate its decision-making rationale could introduce a layer of transparency, potentially boosting trust in the system's operations and facilitating a more nuanced grasp of its decision-making complexities.

Despite the potential benefits of integrating large language models (LLMs) with multiagent systems, several challenges persist. These include the high computational demands of LLMs, the complexity of their decision-making processes, and ethical implications.

A. Enhancing Human Engagement in the Loop

While the human element remains foundational, especially in shaping the initial system prompt or documentation, the potential for a more intertwined human-machine partnership exists.

1) Direct Influence: Encouraging humans to directly shape agent behaviors is key. An intuitive interface could enable users to propose behaviors, pinpoint overarching

Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.

goals, or lay out specific parameters. This merges human intuition with technological prowess, targeting the best results for agents.

2) Feedback Mechanism: It's beneficial when agents offer clear summaries of their decisions, from data analysis to behavioral tweaks. Such transparency strengthens trust, offers clarity, and provides avenues for system enhancements based on human feedback.
3) Making Sense of Complexity: Even though adaptive systems are complex by design, demystifying their workings is essential. Translating intricate operations into comprehensible language paves the way for enhanced human-machine interactions.

B. Diversifying LLM Choices:

While we centered on GPT-4, many other LLMs exist with unique capabilities. Exploring these options and creating clear evaluation standards might lead to even more effective multiagent strategies.

C. Addressing the Black-Box Concern and Hallucination Phenomenon:

While GPT-4's ability to provide explanations is notable, the model remains a proprietary and non-transparent system. This opacity raises significant trust and safety concerns that necessitate a deeper understanding of its internal decision-making processes. Furthermore, GPT models, including GPT-4, are sometimes prone to 'hallucinations' – instances where they produce responses that are not just contextually irrelevant but also factually incorrect or disconnected from reality [26].

Acknowledgment

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Centre for Community Mapping (COMAP).

References

[1] T. Webb, K. J. Holyoak, and H. Lu, "Emergent analogical reasoning in large language models," *Nature Human Behaviour*, pp. 1–16, 2023.

[2] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[3] N. Nascimento, P. Alencar, and D. Cowan, "Self-adaptive large language model (llm)-based multiagent systems," in *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. IEEE, 2023, pp. 104–109.

[4] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, "Let's verify step by step," *arXiv preprint arXiv:2305.20050*, 2023.

[5] J. Huang and K. C.-C. Chang, "Towards reasoning in large language models: A survey," 2023.

[6] I. Dasgupta, A. K. Lampinen, S. C. Chan, A. Creswell, D. Kumaran, J. L. McClelland, and F. Hill, "Language models show human-like content effects on reasoning," *arXiv preprint arXiv:2207.07051*, 2022.

[7] G. V. Aher, R. I. Arriaga, and A. T. Kalai, "Using large language models to simulate multiple humans and replicate human subject studies," in *International Conference on Machine Learning*. PMLR, 2023, pp. 337–371.

[8] M. Kosinski, "Theory of mind may have spontaneously emerged in large language models," *arXiv preprint arXiv:2302.02083*, 2023.

[9] S. Nolfi, "On the unexpected abilities of large language models," *arXiv preprint arXiv:2308.09720*, 2023.

[10] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv:2310.12931*, 2023.

[11] S. Nolfi, "Progress and challenges in adaptive robotics," *Frontiers in Robotics and AI*, vol. 9, p. 1020462, 2022.

[12] A. Almansoori, M. Alkilabi, and E. Tuci, "On the evolution of mechanisms for three-option collective decision-making in a swarm of simulated robots," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 4–12.

[13] N. M. do Nascimento and C. J. P. de Lucena, "Fiot: An agent-based framework for self-adaptive and self-organizing applications based on the internet of things," *Information Sciences*, vol. 378, pp. 161–176, 2017.

[14] G. Lan, J. Chen, and A. Eiben, "Evolutionary predator-prey robot systems: From simulation to real world," in *Proceedings of the genetic and evolutionary computation conference companion*, 2019, pp. 123–124.

[15] F. Sado, C. K. Loo, W. S. Liew, M. Kerzel, and S. Wermter, "Explainable goal-driven agents and robots - a comprehensive review," *ACM Comput. Surv.*, vol. 55, no. 10, feb 2023. [Online]. Available: https://doi.org/10.1145/3564240

[16] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, and Á. Fernández-Leal, "Human-in-the-loop machine learning: A state of the art," *Artificial Intelligence Review*, vol. 56, no. 4, pp. 3005–3054, 2023.

[17] N. Nascimento, P. Alencar, C. Lucena, and D. Cowan, "Toward human-in-the-loop collaboration between software engineers and machine learning algorithms," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 3534–3540.

[18] OpenAI, "Gpt models api," 2023, accessed: 2023-08-07. [Online]. Available: https://platform.openai.com/docs/guides/gpt

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[21] N. Nascimento, "A software framework for developing self-adaptive iot agents," https://github.com/nathyecomp/fiot, 2023.

[22] I. Redbooks and I. B. M. C. I. T. S. Organization, *A Practical Guide to the IBM Autonomic Computing Toolkit*, ser. IBM redbooks. IBM, International Support Organization, 2004. [Online]. Available: https://books.google.com.au/books?id=XHeoSgAACAAJ

[23] S. Nolfi, "A tool for training robots through evolutionary and reinforcement learning methods," https://github.com/snolfi/evorobotpy2, 2020.

[24] O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[25] S. Pontes-Filho, K. Olsen, A. Yazidi, M. A. Riegler, P. Halvorsen, and S. Nichele, "Towards the neuroevolution of low-level artificial general intelligence," *Frontiers in Robotics and AI*, vol. 9, p. 1007547, 2022.

[26] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

Authorized licensed use limited to: University of Central Florida. Downloaded on February 04,2024 at 00:06:51 UTC from IEEE Xplore. Restrictions apply.