Wesaac
2024

Anais do XVIII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações

Organizado por

Bruno Werneck Pinto Hoelz
Conceição de Maria Albuquerque Alves

Brasília/DF
14 a 16 de agosto de 2024

UnB

Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – WESAAC (18. : 2024 : Brasília, DF)
Anais do XVIII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – WESAAC 2024 / organizado por Bruno Werneck Pinto Hoelz, Conceição Maria de Albuquerque Alves. — Brasília, DF, 2024.
188 p. : il.
E-book (PDF)
ISBN 978-65-00-98256-5
DOI: 10.5281/zenodo.13126686
1. Agentes inteligentes. 2. Sistemas multiagente. 3. Ambientes computacionais. 4. Aplicações de sistemas multiagente. I. Hoelz, Bruno Werneck Pinto, org. II. Alves, Conceição Maria de Albuquerque, org. III. Título.

PREFÁCIO

Este documento reúne os trabalhos apresentados nas sessões técnicas da 18ª edição do Workshop-Escola de Sistemas de Agentes, Seus Ambientes e Aplicações – Wesaac 2024, realizado na Universidade de Brasília, entre 14 e 16 de agosto de 2024.

Durante as sessões técnicas, foram apresentados 21 artigos que abarcaram uma grande diversidade temática e interdisciplinaridade. Agradecemos ao comitê de programa pelo rigoroso processo de revisão e pela garantia da qualidade dos trabalhos aceitos e apresentados.

Nessa edição, contamos também com três palestras: o Prof. Jaime Sichman (USP) ofereceu uma retrospectiva dos mais de vinte anos da área de pesquisa de agentes autônomos e sistemas multiagentes; o Prof. Brian Logan (University of Aberdeen, UK & Universiteit Utrecht, NL) abordou o desafio do reconhecimento de intenções em sistemas multiagente; e o Prof. Rafael Cardoso (University of Aberdeen, UK) apresentou um histórico de sua pesquisa em Inteligência Artificial (IA) simbólica e uma nova direção com a IA neurosimbólica. Aos palestrantes, nosso grande agradecimento pela inestimável contribuição.

Foram ministrados ainda três minicursos: o primeiro, pelo Prof. Dr. Maiquel de Brito (UFSC), sobre o desenvolvimento de robôs autônomos utilizando agentes BDI e o *Robot Operating System* (ROS); o segundo, pelos Profs. MSc. Nilson Mori Lazarin (CEFET/RJ) e Dr. Carlos Eduardo Pantoja (CEFET/RJ), com uma introdução a sistemas multiagente distribuídos e embarcados; e o terceiro, pelos Profs. Dra. Célia Ghedini Ralha (UnB) e MSc. Tiago Henrique Faccio Segato (IFB), que apresentaram um processo de desenvolvimento de sistemas multiagente. A todos, agradecemos a preparação e o empenho na realização dos minicursos dessa edição.

Por fim, expressamos nossa gratidão a todos os participantes, bem como às suas respectivas universidades e instituições, por sua presença e dedicação fundamentais para o sucesso dessa edição.

Brasília/DF, agosto de 2024.

Dr. Bruno Werneck Pinto Hoelz

Profa. Dra. Conceição de Maria Albuquerque Alves
Universidade de Brasília

ORGANIZAÇÃO

Organização Geral:
Célia Ghedini Ralha (UnB)

Coordenação do Comitê de Programa:
Conceição de Maria Albuquerque Alves (UnB)
Bruno Werneck Pinto Hoelz (PF)

Comitê de Programa:
Aldo Henrique Dias Mendes (UNIEURO)
Ana Regia de Mendonça Neves (IFB)
Antonio Carlos da Rocha Costa (PUCRS)
Aurelio Ribeiro Costa (STF)
Bruno Werneck Pinto Hoelz (PF)
Carlos Eduardo Pantoja (CEFET RJ)
Carolina Gonçalves Abreu (CJF)
Célia Ghedini Ralha (UnB)
Conceição de Maria Albuquerque Alves (UnB)
Diana Francisca Adamatti (UFRG)
Felipe Rech Meneguzzi (University of Aberdeen, UK)
Fernando Santos (UDESC)
Fernando Szimanski (INEP)
Gleifer Vaz Alves (UTFPR)
Gustavo Alberto Giménez-Lugo (UTFPR)
Jerusa Marchi (UFSC)
Jomi Fred Hübner (UFSC)
José Pergentino de Araújo Neto (European Commission)
Leonardo Henrique Moreira (ECEME)
Maicon Rafael Zatelli (UFSC)
Maiquel de Brito (UFSC)
Marilton Snachotene de Aguiar (UFPEL)
Rafael Cauê Cardoso (University of Aberdeen, UK)
Rafael Heitor Bordini (PUCRS)
Rejane Frozza (UNISC)
Ricardo Choren (IME)

Participação

UFSC · PUCRS · USP · CEFET/RJ

IME · UNIEURO Centro Universitário · FURG · Universidade Federal de Pelotas RS-Brasil

UDESC · UTFPR Universidade Tecnológica Federal do Paraná · UNISC Universidade de Santa Cruz do Sul · Instituto Federal Brasília Campus Brasília

Justiça Federal Conselho da Justiça Federal · STF Supremo Tribunal Federal · INEP · ECEME

University of Aberdeen · Universiteit Utrecht · European Commission

Patrocínio

FAAMAS
International Foundation for Autonomous Agents and Multiagent Systems

Contato
wesaac2024@gmail.com

SUMÁRIO

Multi-agent Perspective of Fake Feedback Attacks on Stochastic Multi-armed Bandits

Charles A. N. Costa[1], **Célia Ghedini Ralha**[1]

[1]Computer Science Department – University of Brasília (UnB)
Campus Dary Ribeiro – Asa Norte – 70.910-900 – Brasília – DF – Brazil

charles.costa@aluno.unb.br, ghedini@unb.br

Abstract. *The problem of false feedback attacks on stochastic Multi-Armed Bandits (MAB) algorithms is the focus of this article from the perspective of Multi-Agent Systems (MAS). We present the roles, beliefs, interactions, and goals of the agents involved in false feedback attacks according to performance metrics to evaluate the MAB algorithms' performance. Three types of attacks illustrate the problem: constant, adaptive, and Jun's adversarial attacks relaxed version, specifically built to exploit vulnerabilities in the e-Greedy and UCB1 algorithms. Experiments exploiting the vulnerabilities inherent in stochastic MAB algorithms present results revealing the useful characteristics for those who design MAS requiring defenses against false feedback attacks.*

1. Introduction

Multi-Armed Bandits (MAB) are online machine-learning algorithms designed to deal with the trade-off between exploration and exploitation. On the metaphorical version of the problem, a player faces a set of levers, usually called arms, each potentially returning a different reward. The player has no previous knowledge about the arms. With a limited number of tries at her disposal, the player has to strategically decide which arm to pull at each step of the game to maximize the accumulated reward while acquiring knowledge about arms' reward functions. The problem appears in many real-world situations, such as the administration of drugs to patients [Bastani and Bayati 2020], showing advertising to website visitors [Shen et al. 2015], and even deciding which stock to buy [Schwartz et al. 2017]. Overall, [Bouneffouf et al. 2020] provides a comprehensive overview of practical applications of MAB algorithms, also discussing their potential to help advance machine learning in many domains.

A way to classify MAB algorithms is by the set of assumptions they make about the reward functions. The stochastic MAB algorithms assume that rewards are sampled from stationary distributions [Slivkins 2019]. There exist many stochastic MAB algorithms that implement different strategies to balance between the exploration and exploitation phases. In this work, we tested two commonly used ones, the ϵ-Greedy and the UCB1. The ϵ-Greedy algorithm uses a constant ϵ in the range [0,1], which is the probability of using exploration. At each round, there is a probability of $\epsilon - 1$ that the learner will choose an option by chance and a ϵ that it will make the greedy decision, i.e., picking the option with the highest average reward [Vermorel and Mohri 2005]. The UCB1

estimates arm rewards' confidence intervals and selects the one with the highest upper bound. Since confidence intervals' lengths are inversely proportional to the sample set length, with sufficient time, UCB1 will select every arm at least once [Auer et al. 2002].

When applicable, stochastic MAB algorithms provide robust results hardly surpassed by other approaches. However, it is well-known that stochastic MABs are vulnerable to data poisoning attacks. Data poisoning attacks are the corruption of reward information for MAB choices. Some works have approached the problem of poisoned stochastic rewards. [Lykouris et al. 2018] cited e-commerce fake reviews patronized by competitors and other non-malicious sources of corruption. [Niss and Tewari 2020] is motivated by the application of bandits in education, where reward information derives from human opinion, thus prone to deviation. [Jun et al. 2018] is motivated by the industrial application of contextual bandits [Li et al. 2010] usually employed in recommender systems. [Vallée et al. 2014] noted that some algorithms robust to one-source corruption are vulnerable to collusion of agents. [Guan et al. 2020] claims that some popular algorithms fail even when attacks are not large all the time. We argue that a fake feedback attack study helps design adequate defenses, especially for Multi-Agent Systems (MAS) design and development.

Although the sets for studying those attacks and defenses are populated by agents performing diverse roles, very few works approach the problem of fake feedback manipulative attacks to stochastic bandits as a multi-agent problem. A multi-agent definition of the problem would help researchers respond to some of the questions related to this scenario, i.e., which roles do agents perform, what capacities must the agents involved have, which information must they pursue, and what are their motivations and commitments, among other questions. The main contribution of this paper is the experimental exploration and analysis of attacks on stochastic MAB from a multi-agent perspective. It presents practical but effective attacks on stochastic MAB, followed by their experimental evaluation. It sheds light on challenges and opportunities related to weak attacks on stochastic MAB within an agent framework.

The rest of this article is as follows: Section 2 presents a multi-agent definition of the problem, Section 3 presents an overview of attacks on stochastic MAB, Section 4 presents the performance measures used on three stochastic MAB attacks, Section 5 brings our experimental outcome and analysis, and Section 6 the conclusion.

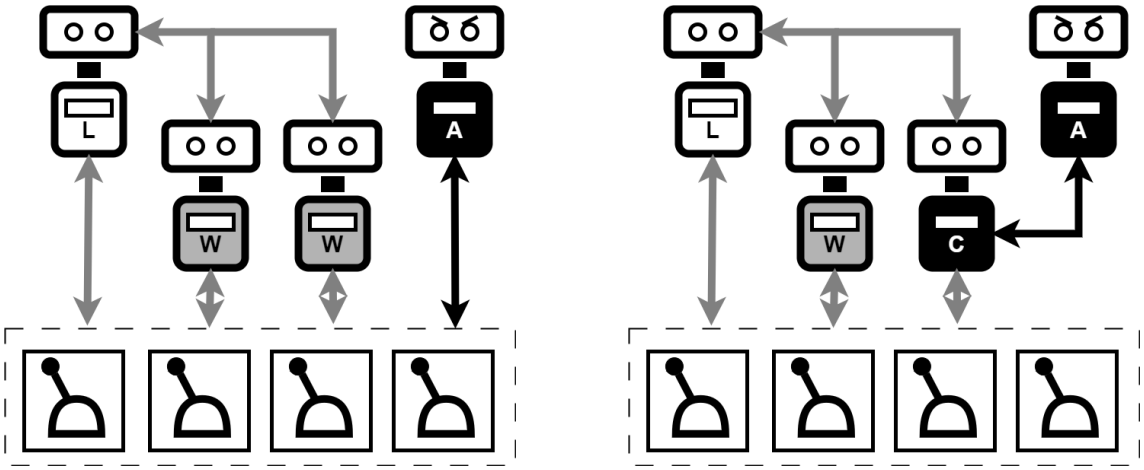
2. Multi-agent Perspective

MAS are systems where agents sharing the same environment act autonomously to achieve common or conflicting goals, in most practical cases, on behalf of users [Wooldridge 2009]. To provide a MAS perspective on the problem, we define the roles, goals and beliefs database, required to reach an understanding.

Following the classification system described in [Wooldridge 2009], the stochastic MAB environment is non-deterministic, episodic, static, and discrete. In this work, we assume no noise or communication cost, which means that any agent can freely communicate, and received messages are equal to the sent ones, despite the communication cost and noise can influence attack efficiency in the real world. We also take that more than one agent can draw the same arm in a round without blocking.

We represent our problem as a game played by a set of agents. As we can see in

Figure 1b, the agents are the Learner (l), the Attacker (a), and a set of Witnesses (W). The set of all agents AG is $\{l, a\} \cup W$. We represent roles and individual agents in lowercase and sets by capital letters, except in Figure 1 for readability. AG's possible actions are drawing arms to collect rewards, communicate, and read information from the environment. The set of arms K is $\{k_1, k_2, ..., k_n\}$. The witnesses are other agents playing in the same environment from whom the Learner can obtain feedback reports about past arm trials. Witnesses' feedback reports help the Learner to evaluate the arms. The co-opted witnesses (C), a subset of W, respond to a's instructions to corrupt the information sent to l. The members of C receive messages from a about the level of corruption they should introduce in the rewards. It is worth noting that $|C| < |W|$, i.e., at least one witness is honest.



(a) Adversarial attack. (b) Fake feedback attack.

Figure 1. Comparing adversarial attack (1a), and fake feedback attack scenarios (1b). Agents are: Learner (L), Attacker (A), Witnesses (W), and Co-opted witness (C).

Learner

The Learner's goal is to collect rewards from the arms while minimizing regret. We define regret measure in Section 4 (Performance measures). The Learner maintains a belief database with all reward information, including rewards obtained by itself and that informed by the witnesses. Equation 1 describes the Learner's belief database (B_l), where t is the round time, α is the agent informing its collected reward, k is the pulled arm, r is the reward.

$$B_l = \{(t, \alpha, k, r) | t \geq 1, \alpha \in AG, k \in K, r \in [0, 1]\} \tag{1}$$

Based on B_l state in t, the Learner employs a policy π which returns the arm that l must draw. The policy π must be defined in $t = 0$, when $B_p = \emptyset$.

$$\pi(B_l, t) = k_t \tag{2}$$

In experiments, π is to be substituted by an MAB policy.

Attacker

The Attacker's goal is to maximize the number of times the Learner pulls a sub-optimal target arm. The Attacker can observe each arm's estimated average reward and Learner's pull count. Equation 3 describes the Attacker's belief database, where $\hat{\mu}$ is the estimated average reward of $k \in K$, and n is the pull count of k.

$$B_a = \{(t, k, \mu, n) | t \geq 1, k \in K, \mu \in [0, 1], n \geq 0\} \tag{3}$$

Thus, based on its belief database, the Attacker applies a policy ρ to calculate the level of corruption that the Co-opted Witnesses will insert into the feedback provided to the Learner, where k_t is the target arm ($k_t \in K$). The ρ function is the fake feedback attack policy.

$$\rho(B_a, k_t, t) = \{(k, c_{t+1}) | k \in K\} \tag{4}$$

Witnesses and Co-opted Witnesses

For generality, we refrain from clearly defining Witnesses' goals and policies. Instead, we describe the Witnesses' behavior in three steps. During each round, a witness will draw an arm, collect the reward, and respond to the Learner with the required information. Witnesses' belief database only stores information about their collected rewards, as described in Equation 5.

$$B_w = \{(t, k, r) | t \geq 1, k \in K, r \in [0, 1]\} \tag{5}$$

However, Co-opted Witnesses have secretly the goal of helping the attacker to manipulate the Learner. They execute an additional step before responding to the Learner, which is receiving the new corruption level to insert in the information provided to the Learner. Thus, Co-opted Witnesses must maintain the current corruption level informed by the Attacker in their database, like in Equation 6.

$$B_c = \{c | c \in [0, 1]\} \cup \{(t, k, r) | t \geq 1, k \in K, r \in [0, 1]\} \tag{6}$$

Interaction Protocol

Here, the protocol of the T-round game is described. The protocol is executed at each $1 \leq t \leq T$. We omit the steps for receiving and sending messages and storing information in the belief databases for brevity. All databases are initialized as empty sets.

1. l applies $\pi(B_l, t)$ to obtain k_t.
2. l draws k_t and collects r_{K_t}.
3. l requires from W feedback reports.
4. a collects the information about K.
5. a applies $\rho(B_a, k_t, t)$ to calculate c_{t+1}.
6. a informs C about c_{t+1}.
7. each w in W draws an arm at random and collects the reward.
8. each c in C add the informed c_{t+1}.

3. Attacks on Stochastic MAB

Manipulative attacks on MAB refer to strategies to manipulate them into increasing the counting pull of a specific arm, frequently a sub-optimal arm, in a way that affects the system's performance or justice. In adversarial attacks, an attacker has control of the reward perceived by the Learner agent [Rangi et al. 2022]. Let us use the slot machine metaphor to explain how adversarial attacks work. When a player pulls a chosen machine, a mechanism informs the adversary of the player and the machine's identities. Then, the adversary will select in a control board which prizes the player will receive, if any.

Compared to adversarial attackers, weak attackers have two flaws: they cannot corrupt all reward reports since some are secured; they have to decide about the corruption before the actual reward is revealed [Rangi et al. 2022]. The weak attacker cannot access the machine prizing mechanism of the slot machines. However, it can co-opt witnesses and instruct them in advance to poison the report provided to the Learner.

Poisoning attacks to stochastic MAB are those in which the attacker corrupts the reward information by adding a carefully calculated value to the actual reward to manipulate the Learner to make decisions in a specific way [Liu and Shroff 2019]. When the weak attack employs Co-opted Witnesses to corrupt the reports provided to the Learner, it is called a fake feedback attack. Figure 1a shows information flow on adversarial attacks and fake-feedback attacks (Figure 1b), where the Attacker (A) counts on Co-opted Witnesses (C) to corrupt the information provided to the Learner (L).

Fake feedback presents a problem for many online, open, and dynamic real-world environments. Let us take the reputation systems employed in many e-commerce platforms. Some sellers contract counterfeiters to insert manipulative reviews of products and sellers, whether to improve their reputation or harm competitors. The work on [Zhang et al. 2013] illustrates a history of generations of fake feedback attacks on the trust system of the Chinese e-commerce platform Taobao.

4. Weak Attacks on Stochastic MAB

This Section presents the performance measures used to evaluate the MAB algorithms' performance, followed by weak attacks on stochastic bandits examples that can succeed when no defense is present in the system.

Performance Measures

MAB algorithms' performance is evaluated using a metric called regret. Regret is the difference between the reward accumulated by pulling only the best arm overall and the real accumulated reward. According to [Vermorel and Mohri 2005], the regret at the round T is defined by Equation 7, where $\mu(k)$ is the mean of the rewards obtained from the arm $k \in K$, K the set of all available arms. Exists an k^* for which $\mu(k^*) \geq max_{k \in K}(\mu(k))$, that, for simplicity, we call $\mu(k^*)$, μ^*. Considering that there were T rounds, and $r_{k,L}(t)$ is the reward the Learner collected from the arm k at the round $t \leq T$.

$$R_L(T) = \mu^* \cdot T - \sum_{t=1}^{T} r_{k,L}(t) \tag{7}$$

Take $r^i_{k,w}(t)$ as the reward that a witness w provided to the Learner in round t. The corruption that the witness w inserted in the system is the difference between the reward it

collected and the reward it informed. If the Attacker did not co-opt w, then the corruption it inserts is always zero. The total corruption level inserted into the game until the final round T is expressed by Equation 8.

$$C(T) = \sum_{w \in W} \sum_{t=0}^{T} |r^i_{k,w}(t) - r_{k,w}(t)| \tag{8}$$

Regarding the agent's objectives in the game, we define success by comparing scenarios when corruption exists or not. Take $N(k, t, c)$ as the counting of times an arm k was pulled by the Learner until some t when the Attacker submitted the game to a level of corruption c. We call Achieved Pulls (AP) the increase that the Attacker could achieve in the pull counting of the target arm using the policy ρ until the horizon T, as described in Equation 9, where k_t is the target arm.

$$AP(T) = N(k_t, T, C(T)) - N(k_t, T, 0) \tag{9}$$

The Attacker's cost for each additional pull is defined by Equation 10.

$$CP(T) = C(T)/AP(T) \tag{10}$$

We say that the policy ρ was successful in a horizon T with a level of corruption $C(T)$ if $AP(T) > \theta$, where θ is a constant greater than zero, arbitrarily defined. This criterion is adequate since in [Jun et al. 2018] authors analyze that if the Oracle attack succeeds, then $\mathbb{E}[N(k_t, T, C(T))] = T - o(T)$, which is coherent with our definition. A cost-efficient attack policy is if it can succeed with a low $CP(T)$.

In the sequence, we present the three types of attacks on stochastic MAB used in this article to illustrate the problem.

Constant Attack

The constant attack policy idea is that all the co-opted witnesses add a fixed corruption value in all reports provided to the learner. Equation 11 illustrates this idea. If the arm i is the target arm k, the co-opted witness will add a value c to reward values. For all other arms, rewards are subtracted by c. This policy is agnostic concerning the MAB policy, requires only one message to inform the appropriate c to the witnesses, and presents a fixed cost budget. However, the c value must be the same during the game.

$$\alpha(k, t) = \begin{cases} c & \text{if } k = k_t \\ -c & \text{otherwise} \end{cases} \tag{11}$$

Adaptive Attack

The adaptive attack goal is to try to maintain the pull count of the target arm in a range. We can see this attack as an improvement of the Constant attack defined in Section 4 where the corruption is a function of the time and the target arm pull counting (Equation 12). If the target arm pulls counting in the previous step ($n_{t-1} = N(k_t, t-1, C(t-1))$) is below a threshold, the attacker raises the corruption level by a small constant amount of

c. Then, the corruption level will rise until it forces the pull count above the lower bound. However, if the pull count of the target arm surpasses a higher threshold, the corruption level decreases by the same small amount. Only note that $\alpha_{i=k}(t) = -1 \cdot \alpha_{i \neq k}(t)$, as in Equation 11. The higher bound helps prevent the total cost of corruption from growing without a limit. The parameters are c_I, the inferior corruption level, c_S, the superior corruption level, c, the increase/decrease factor, and (R^S, R^I) are the superior and the inferior ratio bounds, respectively. R^S and R^I are in the $[0, 1]$ interval and $R^S > R^I$.

$$c(t, n_{t-1}, c_{t-1}) = \begin{cases} c_I & \text{if } t \leq 1 \\ min(c_S, c_{t-1} + c) & \text{if } t > 1 \text{ and } n_{t-1} < t \cdot R^I \\ max(c_I, c_{t-1} - c) & \text{if } t > 1 \text{ and } n_{t-1} > t \cdot R^S \\ c_{t-1} & \text{otherwise} \end{cases} \tag{12}$$

This attack has the advantage of being agnostic to the MAB policy, and the level of corruption will adapt to a desired target arm pull. The cost budget has the upper bound $CP(T) \leq T \cdot c_S$. However, the Attacker agent cannot know the cost budget in advance, although it might be possible by analyzing the stochastic MAB policy, which hurts agnosticism.

Jun's Adversarial Relaxed Attack

The attacks to ϵ-Greedy and UCB1 defined in [Jun et al. 2018] are adversarial. The general idea is to carefully craft the corruption value to subtract from the reward of other arms but the target arm to minimize the total corruption. We can derive a weaker attack from [Jun et al. 2018] by relaxing the assumption that the attacker can know in advance which reward an arm will give.

Let us substitute the reward in $t + 1$, $r_k(t + 1)$ by $\mathbb{E}[r_k(t + 1)] = \mu_k(t)$, since the sample mean is an unbiased estimator for $\mathbb{E}[r_k(t + 1)]$ in the stochastic MAB problem. Making this relaxation and rearranging the equations from [Jun et al. 2018], Jun's relaxed attack to ϵ-Greedy is presented in Equation 15. $N_i(t)$ is the number of times that the Learner has pulled the arm i until the round t, $\hat{\mu}_i$ is the sample mean of the arm i, being that k is the index of the target arm and σ and Δ_0 are parameters.

$$\beta(N) = \sqrt{\frac{2\sigma^2}{N} log \frac{\pi^2 K N^2}{3\delta}} \tag{13}$$

$$r_k(t) = \hat{\mu}_K(t - 1) - 2\beta(N_k(t - 1)) \tag{14}$$

$$\alpha_i(t) = (N_i(t - 1) + 1) \cdot (\hat{\mu}_i(t - 1) - r_k(t - 1)) \tag{15}$$

By its turn, Jun's relaxed attack to UCB1 is presented in Equation 18.

$$A_i(t) = \sum_{s=1}^{t} \alpha_i(s) \tag{16}$$

$$r'_k(t) = r_k(t) - \Delta_0 \tag{17}$$

$$\alpha'_i(t) = (N_i(t - 1) + 1) \cdot (\hat{\mu}_i(t - 1) - r'_k(t - 1)) - A_i(t - 1) \tag{18}$$

Our relaxed version of Jun's attacks presented in [Jun et al. 2018] produces a higher corruption level when compared to the original ones since it attacks every arm that is not the target one in every round. The attacks must be indiscriminate since the attacker cannot know each one the Learner will choose. Because of this, we can no longer maintain the guarantee that cost will be limited by $O(log(T))$, as claimed in [Jun et al. 2018]. The cost tends to grow faster than $O(Klog(T))$, as in the experiments described in Section 5.

5. Experiments

This section presents experiments with the weak attacks proposed in Section 4 on manipulating the Learner agent employing UCB1 and ϵ-Greedy strategies, using the metrics defined in that section. The objective is to provide insights into the strengths and limitations of each approach, facilitating their application in the development of defenses against this kind of manipulation in MAS.

We implemented a multi-agent framework using the Java Agent Development Framework (JADE) middle-ware [Bellifemine et al. 2007]. The framework code is available at `github.com/charlesANC/BanditsExperiment`. The agent population is the Learner l, the Attacker a, nine honest witnesses, and five co-opted ones. There are more agents than arms to guarantee overlapping evaluations. For the experimental setting, $K = \{A1, B1, B2, C1, C2\}$. We set up the arm's reward functions as stochastic, stationary, sampled from a Gaussian distribution accordingly with the following profiles: A1 reward distribution is $\mathcal{N}(0.9, 0.1)$; B1 and B2's are $\mathcal{N}(0.85, 0.30)$; and C1 and C2's are $\mathcal{N}(0.75, 0.50)$. In all runs, the target arm is C2. We run games of 2,000 rounds varying learner's and attacker's policies. The Attacker can use Constant, Adaptive, and Jun's relaxed attacks described in Section 4. The Learner can use ϵ-Greedy and UCB1. For calculating the achieved pulls ($AP(T)$), we also ran games where the Learner employed the stochastic MAB with no Attacker agent. We repeated each combination 30 times. Table 1 shows the general parameters of our experiments.

Table 1. General experimental parameters

Parameter	**Value**
Number of rounds	2,000
Number of repetitions	30
ϵ value	0.80
Evaluation values range	$[0, 1]$
Honest witnesses	9
co-opted witnesses	5
c (Constant)	1.00
c_I	0
c_S	1.00
c (Adaptive)	0.20
Adaptive 1 (R^I, R^S)	(0.40, 0.60)
Adaptive 2 (R^I, R^S)	(0,80, 0.90)
δ	0.025
σ	0.001
Δ_0	0.10

For the Constant attack, we set the parameter c as 1 since the rewards' range is $[0, 1]$. For the Adaptive attack, the initial corruption value C_I was 0, and we experimented with two ranges of $[R^I, R^S]$ which were $[0.4, 0.6]$ and $[0.8, 0.9]$, which we differentiate as Adaptive 1 and 2. For Jun's relaxed attack, we set the value of parameter δ as 0.025, and Δ_0 as 0.10, such as in experiments described in [Jun et al. 2018], and σ as 0.001. It is worth noting that authors in [Jun et al. 2018] ran experiments with σ values varying from 0.05 to 0.5, depending on the scenario. However, we observed that even 0.05 was too high for our set-up, resulting in higher costs and lower efficiency. Co-opted witnesses cropped informed corrupted rewards into the $[0, 1]$ range to avoid providing out-of-scale values.

Table 2 resumes the experimental outcome. The columns show the MAB algorithm, the employed attack, regret, target arm pulls, total cost, and cost per additional target pull, respectively, as defined in Section 4. When there is no employed attack, $C(T) = 0$, thus in lines 1 and 6 $N(k_t, T, C(T)) = N(k_t, T, 0)$. However, we need $N(k_t, T, C(T))$ in lines 1 and 6 for calculating $AP(T)$ and $CP(T)$.

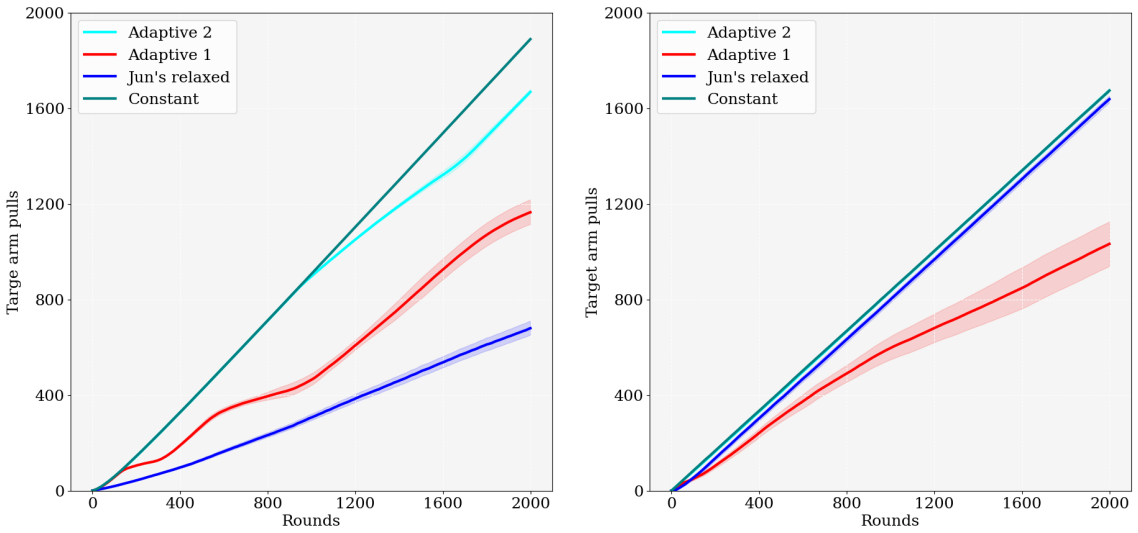
Table 2. Resumed measures over MAB algorithms and attacks. The values represent the mean with the standard variation in parentheses.

MAB	**Attack**	$\mathbf{R_L(T)}$	$\mathbf{N(k_t, T, C(T))}$	$\mathbf{AP(T)}$	$\mathbf{C(T)}$	$\mathbf{CP(T)}$
UCB1	-	70.93 (8.77)	131.50 (9.70)	-	-	-
UCB1	Constant	297.20 (21.42)	1,889.07 (1.69)	1,757.57 (10.03)	51,549.73 (68.40)	29.33 (0.18)
UCB1	Adaptive 1	216.34 (24.91)	1,164.87 (134.40)	1,033.37 (135.04)	19,769.67 (3,620.93)	19.11 (2.51)
UCB1	Adaptive 2	275.42 (24.39)	1,668.43 (24.99)	1,536.93 (27.41)	30,803.83 (1,565.36)	20.04 (0.89)
UCB1	Jun's relaxed	181.52 (19.12)	679.83 (77.59)	548.33 (78.79)	13,601.23 (1,271.82)	25.04 (2.18)
ϵ-Greedy	-	31.45 (8.96)	79.67 (8.83)	-	-	-
ϵ-Greedy	Constant	274.45 (23.26)	1,673.80 (13.08)	1,594.13 (15.41)	51,581.90 (68.72)	33.45 (0.30)
ϵ-Greedy	Adaptive 1	180.72 (38.44)	1,032.57 (246.09)	952.90 (244.51)	17,421.66 (8,252.80)	20.12 (9.58)
ϵ-Greedy	Adaptive 2	269.08 (21.88)	1,594.90 (18.22)	1,674.57 (18.74)	50,468.31 (3,910.64)	31.65 (2.48)
ϵ-Greedy	Jun's relaxed	268.87 (20.86)	1,638.10 (37.38)	1,594.90 (38.61)	185,662.13 (26,423.06)	123.55 (19.79)

Note that all attacks successfully increased the target arm pull count. However, we can also observe significant differences in cost values. Due to the used parameters, the cost of constant attack dominated the two tested adaptive attack ranges, which resulted in a lower cost per additional pull.

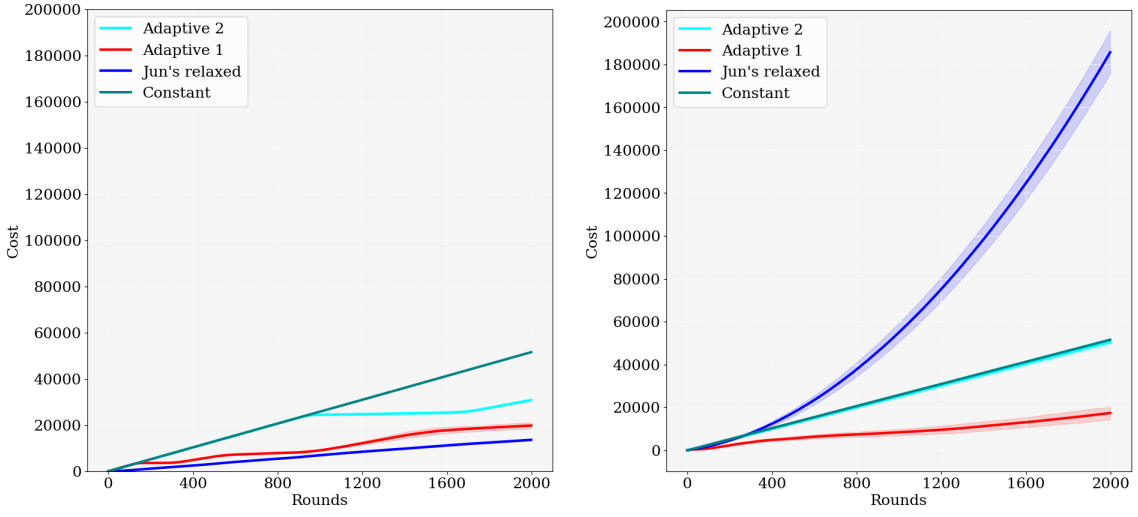
With the same parameters, Jun's relaxed attacks had very different performances depending on the MAB algorithms, despite both versions presenting the highest cost per additional pull compared to the other attack strategies. Note in line 5 of Table 2, against the UCB1 algorithm, the achieved pulls $AP(T)$ and the total cost $C(T)$ were significantly

lower than other strategies' values. However, $CP(T)$ was higher. Against ϵ-Greedy, Line 10, the achieved pulls of Jun's relaxed attack were comparable to those obtained from Constant and Adaptive attacks. However, the cost measures were much higher than any other tested strategy. Also, note in Figure 2 that the target arm pulls over rounds in ϵ-Greedy experiments, that Jun's relaxed attack did not surpass the Constant attack at any moment. Those results allow us to conclude that Jun's relaxed attacks needed more work on tunning parameters, which is a disadvantage compared to Constant and Adaptive strategies. However, when testing a possible defense against data corruption, pursuing an attack that presents an exponentially growing cost curve is desirable.



(a) UCB1 - Target arm pulls. **(b) ϵ-Greedy - Target arm pulls.**

Figure 2. Target arm pulls over MAB algorithms and attacks.



(a) UCB1 - Total Cost. **(b) ϵ-Greedy - Total Cost.**

Figure 3. Cost of corruption over MAB algorithms and attacks.

In summary, the experimental results offer valuable insights into the behavior and cost-effectiveness of the weak attacks presented in Section 4. One can observe that while all attacks were successful in manipulating the learner choices, performance measures

varied significantly. Specifically, the Adaptive attack has the advantages of being agnostic regarding the MAB algorithm, easily configurable, and cost-efficient. Jun's relaxed attacks present advantages in scenarios for testing algorithmic defenses.

6. Conclusion

This article presents the problem of fake feedback attacks on stochastic MAB algorithms from a multi-agent perspective. Our work highlighted the vulnerability of stochastic MAB with two commonly used algorithms – ϵ-Greedy and UCB1 – illustrating the problem with three types of attacks – Constant, Adaptive, and Jun's Adversarial Relaxed. By considering the roles, beliefs, interactions, goals, and motivations behind these attacks, we contributed to the problem understanding within the MAS perspective, which facilitates the development of defenses against this kind of manipulative attack. Our outcome pointed to the cost-efficiency of Constant and Adaptive attacks, besides the advantages of Jun's relaxed attacks on testing potential defenses.

Our vision is that other works that studied data poisoning attacks on MAB without a strong MAS perspective, such as [Lykouris et al. 2018], [Liu and Shroff 2019], [Niss and Tewari 2020], and [Rangi et al. 2022], could benefit from comparison using the performance measures presented in Section 4 (Equations 7-10). Future work will compare them using the performance metrics proposed in Section 4. Another possible direction to further work should be the effective defense development against fake feedback attacks in MAS requirements design, which is useful for diverse real-world problems such as e-commerce marketing, the stock market, and drug administration.

References

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3):235–256.

Bastani, H. and Bayati, M. (2020). Online decision making with high-dimensional covariates. *Operations Research*, 68(1):276–294.

Bellifemine, F., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, Ltd, Chichester, West Sussex, England.

Bouneffouf, D., Rish, I., and Aggarwal, C. (2020). Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, page 1–8. IEEE Press.

Guan, Z., Ji, K., Bucci Jr, D. J., Hu, T. Y., Palombo, J., Liston, M., and Liang, Y. (2020). Robust stochastic bandit algorithms under probabilistic unbounded adversarial attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4036–4043, New York, USA. AAAI Press.

Jun, K.-S., Li, L., Ma, Y., and Zhu, J. (2018). Adversarial attacks on stochastic bandits. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31, Montréal, Canada. Curran Associates, Inc.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670.

Liu, F. and Shroff, N. (2019). Data poisoning attacks on stochastic bandits. In *International Conference on Machine Learning*, pages 4042–4050. PMLR.

Lykouris, T., Mirrokni, V., and Leme, R. P. (2018). Stochastic bandits robust to adversarial corruptions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, page 114–122, Los Angeles, USA. ACM.

Niss, L. and Tewari, A. (2020). What you see may not be what you get: Ucb bandit algorithms robust to ε-contamination. In Peters, J. and Sontag, D., editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 450–459, Virtual. PMLR.

Rangi, A., Tran-Thanh, L., Xu, H., and Franceschetti, M. (2022). Saving stochastic bandits from poisoning attacks via limited data verification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36-7, pages 8054–8061, New York, USA. AAAI Press.

Schwartz, E. M., Bradlow, E. T., and Fader, P. S. (2017). Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science*, 36(4):500–522.

Shen, W., Wang, J., Jiang, Y.-G., and Zha, H. (2015). Portfolio choices with orthogonal bandit learning. In *Twenty-fourth international joint conference on artificial intelligence*.

Slivkins, A. (2019). Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286.

Vallée, T., Bonnet, G., and Bourdon, F. (2014). Multi-armed bandit policies for reputation systems. In Y., D., F., Z., J.M., C., and J., B., editors, *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, volume 8473 of *Lecture Notes in Computer Science*. Springer, Cham, Salamanca, Spain.

Vermorel, J. and Mohri, M. (2005). Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European Conference on Machine Learning*, ECML'05, page 437–448, Berlin, Heidelberg. Springer-Verlag.

Wooldridge, M. (2009). *An Introduction to Multiagent Systems*. Wiley, Chichester, UK, 2 edition.

Zhang, Y., Bian, J., and Zhu, W. (2013). Trust fraud: A crucial challenge for china's e-commerce market. *Electronic Commerce Research and Applications*, 12(5):299–308.

Direção assistida por agentes cognitivos: Uma análise de viabilidade para o controle de velocidade

Ricardo de Monteiro e Tavares [1], Carlos Eduardo Pantoja [2], Nilson Mori Lazarin [1,2], Flávia Cristina Bernadini [1]

[1]Instituto de Computação - Universidade Federal Fluminense - Niterói, RJ – Brasil

[2]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ) Rio de Janeiro, RJ – Brasil

ricardotavares@id.uff.br

Abstract. *This paper presents a study on the feasibility of using Embedded Multi-Agent Systems for vehicle autonomous speed control. The system is implemented in a prototype equipped with Raspberry Pi and Arduino boards, using RFID tags to read traffic signs. The prototype reads the tags and adjusts the speed according to the data received. The experimental results demonstrate the viability of the proposed model despite some technical limitations, suggesting that future improvements may include more powerful RFID readers and the incorporation of encryption for greater security.*

Resumo. *Este trabalho apresenta um estudo sobre a viabilidade do uso de Sistemas Multiagente Embarcados para o controle autônomo de velocidade em veículos. Utilizando tags RFID para leitura de sinalizações de trânsito, o sistema é implementado em um protótipo equipado com um Raspberry Pi e placas Arduino. O protótipo lê as tags e ajusta a velocidade conforme os dados recebidos. Os resultados experimentais demonstram a viabilidade do modelo proposto, apesar de algumas limitações técnicas, sugerindo que futuras melhorias podem incluir leitores RFID mais potentes e a incorporação de criptografia para maior segurança.*

1. Introdução

Atualmente a fiscalização da velocidade veicular é feita por radares, seja de forma fixa ou móvel. Os radares fixos ficam instalados em um determinado ponto, para medir a velocidade em um trecho da via, para medir a velocidade média no trecho. Já os radares móveis, são agentes de trânsito ou policiais rodoviários que vão para um determinado ponto da via e realizam o monitoramento por meio de ondas eletromagnéticas por meio de uma pistola de medição [Lara 2020]. Porém, a fiscalização de velocidade veicular por meio de radares torna-se ineficiente quando há uma grande quilometragem de vias para fiscalizar, porque necessitaria de um número considerável de radares e isto tem impacto nos custos de operacionalização [Redação Mobilidade 2022]. Diante deste cenário, este trabalho visa analisar a viabilidade do uso de tecnologias baseadas em agentes, para permitir que o próprio veículo faça, de forma autônoma e/ou colaborativa, este controle de velocidade a partir da leitura das sinalizações presentes nas vias.

Agentes são entidades autônomas da inteligência artificial distribuída, capazes de perceberem seu ambiente ao redor e, de forma autônoma e proativa, tomarem decisões baseados em seus próprios objetivos e percepções, que coexistem em um sistema multiagente (SMA), podendo ter objetivos comuns, conflitantes ou sistêmicos [Wooldridge 2009]. Agentes e SMAs vem sendo utilizando em soluções embarcadas para prover autonomia a protótipos de veículos autônomos terrestres [Barros. et al. 2014, Souza de Castro et al. 2018] através do uso de extensões do Jason [Bordini et al. 2007] para programação de agentes em sistemas embarcados [Brandão et al. 2021, Lazarin et al. 2024] e simulados [Heijmeijer and Vaz Alves 2018, Souza de Castro et al. 2022]. O JasonEmbedded [Pantoja et al. 2023] é uma versão do Jason com potencial de uso em sistemas embarcados, permitindo o uso de arquitetura específicas de agentes embarcados capazes de gerenciar microcontroladores, que adota o modelo cognitivo Belief-Desire-Intention (BDI) [Bratman et al. 1988], adicionando uma camada de cognição próxima ao pensamento humano em sistemas embarcados.

O objetivo deste trabalho é analisar a viabilidade do uso de SMA embarcados para no controle de velocidade de forma que um veículo, de forma autônoma, consiga ler os dados das sinalizações presentes nas vias e realize o controle da velocidade veicular de forma condizente com os dados lidos da sinalização. Para tal, será criado um protótipo de um veículo terrestre autônomo utilizando-se componentes eletrônicos, um computador de placa única Raspberry Pi e duas placas Arduino. O protótipo terá um SMA embarcado utilizando o framework JasonEmbedded e suas extensões para sistemas embarcados. As placas de trânsito serão representadas por tags RFID (*Radio Frequency Identification*) indicando a velocidade permitida pela via. Serão realizados experimentos para garantir a leitura da tag RFID, a comunicação entre os agentes e o processo de controle de velocidade. Os experimentos buscam mostrar a viabilidade, em ambiente experimental, do potencial da abordagem.

Este trabalho está dividido da seguinte forma: na Seção 2 será a presentada a metodologia proposta do trabalho; na Seção 3 serão apresentados o estudo de caso e os experimentos realizados; por fim, na Seção 4 será apresentada a conclusão.

2. Metodologia proposta para controle de velocidade usando RFID

O controle de velocidade em estradas é um procedimento adotado em diversos países. Atualmente, os radares de velocidade identificam em pontos estáticos ao longo da estrada se um determinado veículo passou ou não de uma velocidade pré-determinada, a fim de regular a velocidade praticada. Contudo, a aderência ou não a velocidade pré-determinada depende da disponibilidade do condutor e da visibilidade da sinalização. Além disso, a identificação do veículo pode ser burlada como a troca da placa ou a partir da inserção de objeto que dificultem a identificação da placa.

Ao adotar um mecanismo de direção assistida, a transferência da responsabilidade de controle de velocidade é transferida para o sistema veicular. Neste trabalho, visamos aplicar agentes cognitivos embarcados, que através de suas percepções do mundo físico, onde podem ser utilizadas etiquetas RFID, por exemplo, o agente pode auxiliar o motorista, emitindo alerta, ou até mesmo, reduzir a velocidade do veículo de forma autônoma.

A tecnologia RFID já é utilizada em diversas aplicações, tais como controle Pa-

trimonial e logístico, utilização em pedágios, e autenticação de funcionários. As sinalizações de trânsito serão confeccionadas respeitando as regras atuais do Código de Trânsito Brasileiro (CTB). As tags RFID conterão as mesmas informações que a placa apresenta visualmente e a instalação poderia ser feita no asfalto da via. As tags serão do tipo passivo, o que permite com que os sistemas, inicialmente, não dependam de baterias ou acesso à energia elétrica, diminuindo assim o custo de implantação. A cada vez que um veículo identificar uma sinalização, ela encaminhará a percepção ao SMA, que ficará responsável por atender ao especificado pela sinalização através do controle de velocidade e do limitador de frenagem. O leitor de RFID, faz a captura dos dados contidos nas tags através das ondas de radiofrequência, permitindo a leitura em qualquer posição, desde que, esteja no raio de leitura [Alecrim and Marques 2023]. A Figura 1 exemplifica como ficaria no mundo físico.

Figura 1. Ilustração de instalação das tags RFID nas vias urbanas.

Existem dois tipos de chip RFID, o passivo e o ativo. O chip passivo, só irá transmitir os dados quando estiver no raio do leitor, pois o mesmo se aproveita das ondas de rádio frequência, gerado pela antena do leitor, para se energizar. Já os chips ativos, funcionam semelhantemente, porém possui uma bateria acoplada neles para haver uma transmissão constante e unidirecionalmente. A distância para a leitura varia conforme o tipo e a frequência utilizada. O modelo passivo, a distância da leitura, pode chegar a 15 metros, enquanto o ativo, pode chegar a 300 metros. Para o desenvolvimento deste sistema, utilizamos o chip RFID do tipo passivo, devido ao não uso de baterias. O que removeria os custos necessários para manter a bateria com carga.

Na Figura 2 é apresentada a arquitetura do sistema embarcado no veículo. A arquitetura física do sistema proposto é composto pelas seguintes unidades: Unidade Leitora (UL); Unidades Externas (UEx) e Unidade Controladora (UC).

- A UL é composta pelo leitor de RFID, que faz a leitura dos dados presentes nas tags RFID instaladas no asfalto.
- A UC é composta por um Arduino e um Raspberry Pi. O Arduino fará o papel de comunicação entre as UEx, a UL e o Raspberry Pi. A comunicação feita entre o Raspberry Pi e o Arduino é realizado por meio de comunicação serial usando o Javino [Lazarin and Pantoja 2015], que envia as informações para o SMA em execução pelo JasonEmbedded, localizado no Raspberry Pi.
- As UEx é composta por uma ou mais dispositivos, sensores e atuadores, que estão instalados nos veículos que percebem o ambiente e enviam para os agentes embarcados na UC, ou atuam no ambiente, conforme deliberação dos agentes na UC.

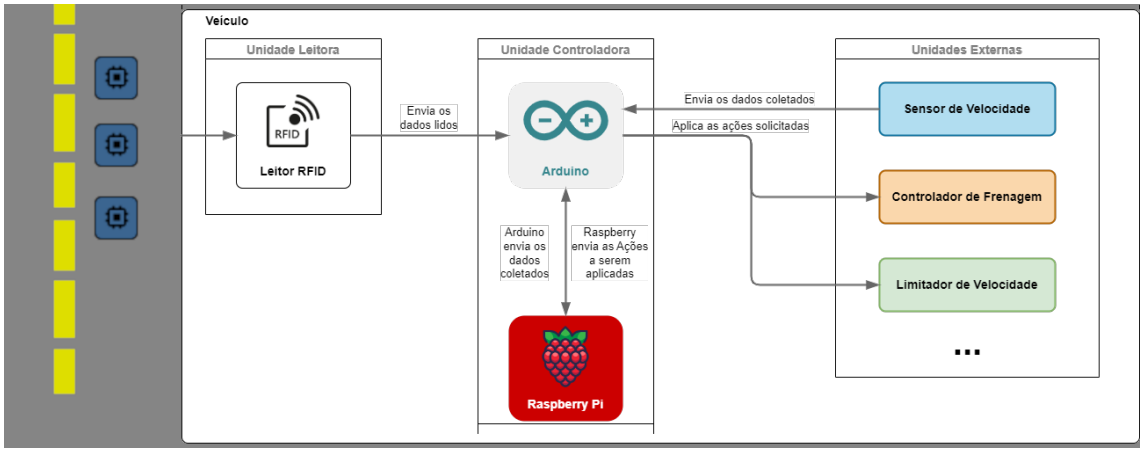


Figura 2. A arquitetura física de um veículo considerando a necessidade do controle de velocidade autônoma.

3. Estudo de Caso

O estudo de caso consiste em reproduzir em um ambiente controlado uma estrada com diferentes limites de velocidades (Baixa, Média e Alta) e conforme o veículo for trafegando, este deve obedecer à sinalização. Para isso foram realizados três experiências. A primeira experiência compreende a validação da leitura dos dados do chip RFID pelo agente. A segunda experiência, compreende a comunicação entre agentes e a realização de ações do agente para o protótipo após a leitura do chip RFID. A terceira, e última, experiência compreende do protótipo ler vários chips RFID ao longo de um circuito e realizar as ações de adequação de velocidade.

Para a execução do estudo de caso foi utilizado um *Cognitive Hardware On Network Basic Prototype (ChonBot) 2WD* [Lazarin et al. 2023], o qual foi adaptado, adicionando um Arduino Uno R3, um *Hub* USB, um leitor de RFID modelo MFRC522. O protótipo é apresentado na Figura 3.

Figura 3. O Protótipo (ChonBot 2WD - adaptado).

O leitor RFID modelo MFRC522 foi escolhido por ser de baixo custo e dimensões pequenas. O leitor possui frequência de 13.56MHz e pode ler tags RFID em até 4 centímetros de distância. Um dos Arduinos é dedicado ao leitor RFID e o outro para o circuito de controle das funções básicas de movimentação, iluminação e sensoriamento de distância e luminosidade. O *Hub* USB permite a comunicação entre o RaspberryPi e os Arduinos. O circuito é descrito na Figura 4.

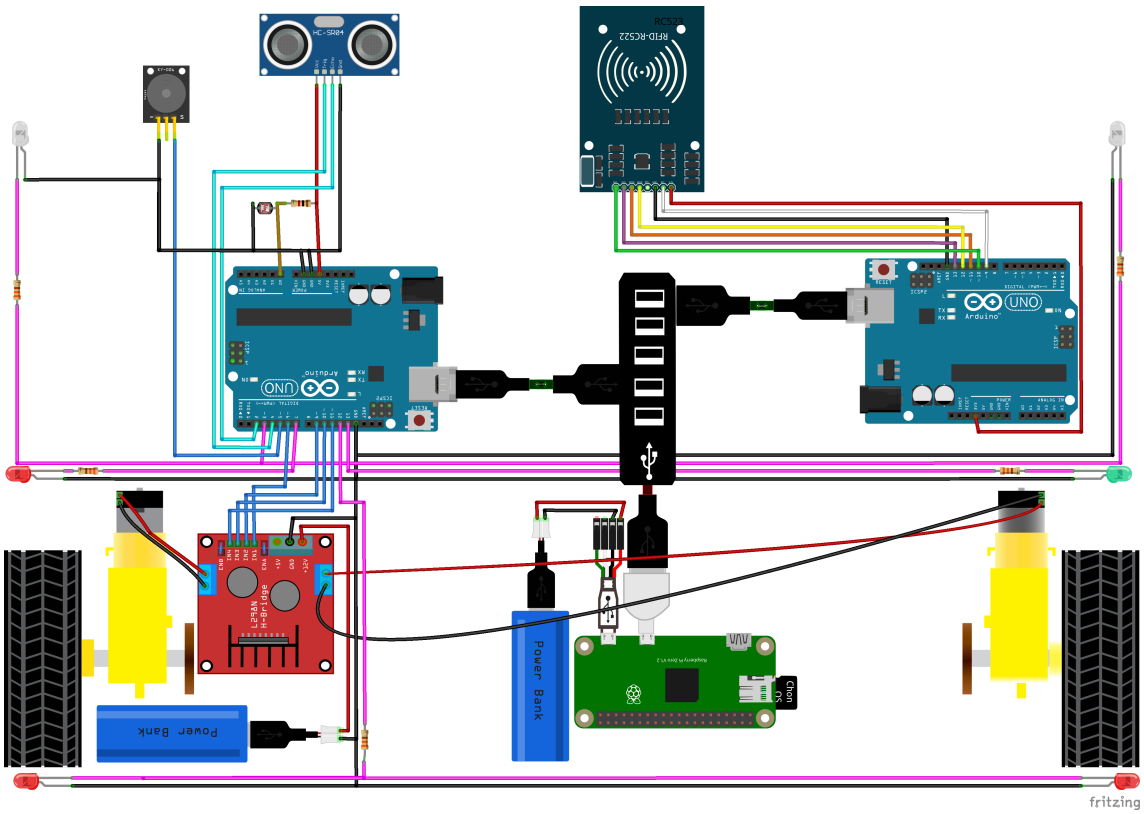


Figura 4. O Circuito do protótipo.

O Sistema multiagente desenvolvido contém dois agentes do tipo ARGO [Pantoja et al. 2016] — o agente *pilot* e o *copilot*. O *pilot* é responsável por enviar os comandos para serem executados pelas UEx, já o agente *copilot* faz as solicitações de informações provindas das UEx e da UL e informa para o agente *pilot* a informação atual do ambiente. Os códigos do raciocínio dos agentes é apresentado abaixo:

```
!start.

+!start <- .port(ttyACM0);  .percepts(open).

+!stopAction[source(copilot)]<-
    .act(stop); .act(breakLOn); .act(lightOff).

+!speedLimiterAction(X)[source(copilot)]
: X == med <-
    .act(speedM); .act(lightOn); .act(goAhead); .act(breakLOff).

+!speedLimiterAction(X)[source(copilot)]
: X == low <-
    .act(speedL); .act(lightOnL); .act(goAhead);  .act(breakLOff).

+!speedLimiterAction(X)[source(copilot)]
: X == high <-
    .act(speedH); .act(lightOnH); .act(goAhead); .act(breakLOff).
```

Código-fonte 1. Raciocínio do agente *pilot*.

```
!start.

+!start <- .port(ttyACM1); .percepts(open); .send(pilot, achieve, speedLimiterAction(high)).

+roadAction(spdhigh)[source(percept)] <-  .send(pilot, achieve, speedLimiterAction(high)).

+roadAction(spdmed)[source(percept)] <- .send(pilot, achieve, speedLimiterAction(med)).

+roadAction(spdlow)[source(percept)] <- .send(pilot, achieve, speedLimiterAction(low)).

+roadAction(stop)[source(percept)] <- .send(pilot, achieve, stopAction).
```

Código-fonte 2. Raciocínio do agente *copilot*.

O código do firmware da unidade leitora de RFID adicionada ao protótipo é apresentado abaixo:

```
#include <Javino.h>           /* https://github.com/chon-group/javino2arduino/releases/latest */
#include <SPI.h>              /* https://www.arduino.cc/reference/en/language/functions/communication/spi/ */
#include <MFRC522.h>          /* https://github.com/miguelbalboa/rfid/releases/latest */
#define RFIDresetPin  9       /* https://github.com/miguelbalboa/rfid#pin-layout */
#define RFIDsdaPin   10       /* https://github.com/miguelbalboa/rfid#pin-layout */
#define BLOCKREAD     2

Javino javino;
MFRC522 rfid(RFIDsdaPin, RFIDresetPin); // Instancia da classe do leitor
MFRC522::MIFARE_Key key={0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

void setup() {
  javino.start(9600);   /* Inicia a comunicacao Serial */
  SPI.begin();          /* Inicia a comunicacao SPI    */
  rfid.PCD_Init();      /* Inicia o MFRC522            */
}

void serialEvent(){  javino.readSerial(); }

void loop(){
  if(javino.availableMsg() && javino.requestPercepts()){javino.sendPercepts();}
  lerCartao();
}

void lerCartao(){
  if(rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()){ /*Funcao que verifica se alguma tag foi lida*/
    MFRC522::PICC_Type tag = rfid.PICC_GetType(rfid.uid.sak);
    if(tag == MFRC522::PICC_TYPE_MIFARE_MINI ||  tag == MFRC522::PICC_TYPE_MIFARE_1K || tag == MFRC522::
      PICC_TYPE_MIFARE_4K){
      byte bufferLen = 18;
      byte readBlockData[18];
      MFRC522::StatusCode auth = rfid.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, BLOCKREAD, &key, &(rfid.uid));
      MFRC522::StatusCode read = rfid.MIFARE_Read(BLOCKREAD, readBlockData, &bufferLen);
      if(auth == MFRC522::STATUS_OK && read == MFRC522::STATUS_OK){
        String strValue = String((char *)readBlockData);
        javino.addPercept("roadAction("+strValue+")");
      }
      rfid.PICC_HaltA();
      rfid.PCD_StopCrypto1();
    }
  }
}
```

Código-fonte 3. Firmware do leitor RFID.

3.1. Experimentos

O objetivo do primeiro experimento é verificar se o agente recebe os dados lidos de uma tag. Para realizar o experimento, foram desenvolvidos o código para a leitura das tags RFID, e um agente ARGO, programado na ChonIDE [Souza de Jesus et al. 2023], que solicita ao Arduino, por meio do Javino, a leitura da placa e informa uma mensagem na tela informando qual placa foi lida. Neste experimento, todas as leituras realizadas e mensagens informadas pelo agente, por intermédio do Arduino, corresponderam, fielmente, aos dados presentes nos chips.

O objetivo do segundo experimento é verificar se é possível realizar a comunicação entre os agentes *pilot* e *copilot*, e se o agente *pilot* consegue controlar o protótipo. Para realizar este experimento, será adicionado um novo agente ao código do experimento anterior, que passará a receber as informações do agente já desenvolvido, realizando ações

de controle de intensidade do motor, ajustando a intensidade de luminosidade e desligando os faróis dianteiros e ligando e desligando as luzes de freio. O agente *copilot*, desenvolvido na experiência anterior, sofrerá modificações para permitir a comunicação com o novo agente desenvolvido. O agente *pilot* conseguia controlar o protótipo por meio das ações, já previamente configuradas. Neste experimento, a comunicação entre os agentes foi realizado com sucesso, sendo necessário apenas algumas correções no código do agente *copilot* para o uso do método adequado de comunicação para o *pilot*, e todas as ações feitas pelo *pilot* foram executadas com sucesso.

Para o terceiro experimento, o protótipo passa a trafegar em um circuito em que consiste em uma via reta com algumas tags RFID dispostos uniformemente ao longo da via. As tags foram inseridas na superfície da via e a sequência de leitura foi proposto da seguinte forma: lê-se primeiramente a placa de Velocidade Média, depois lê-se as placas de Velocidade Alta, Velocidade Baixa e de Pare respectivamente. Neste experimento, o sistema conseguiu ler e aplicar as ações determinadas no programa, porém houve alguns problemas durante as execuções destes experimentos. Os problemas foram causados devido a uma inconsistência nas rotações dos motores do protótipo. Devido a isto, o carrinho não andava em linha reta, mas levemente inclinado à esquerda, causando problemas de leitura das tags RFID instalados no circuito. Pois quando o protótipo passa pelo chip RFID a alguns milímetros ao lado do leitor, devido à baixa potência do leitor RFID, o sistema não conseguia realizar a leitura do chip e, consequentemente, a não aplicação das ações configuradas para o trecho do circuito.

Na Figura 5 são apresentadas algumas imagens da realização dos experimentos, também disponíveis[1] em vídeo.

Figura 5. Imagens da realização dos experimentos.

[1] https://papers.chon.group/WESAAC/2024/direcaoAssistida/

4. Conclusão

Este trabalho apresentou um estudo de viabilidade do uso de SMA Embarcado no controle de velocidade veicular por meio de um protótipo e tags RFID. As experiências realizadas mostraram que o modelo apresentado, apesar dos problemas ocorridos durante a experiência número 3, produziu resultados gerais satisfatórios. Logo, é possível que um veículo, de forma autônoma, consiga ler os dados das sinalizações presentes nas vias e realize o controle da velocidade veicular de forma condizente com os dados lidos da sinalização.

Para trabalhos futuros, se faz necessário a utilização de um leitor de RFID mais potente ou um chip RFID com antena que permita com que os veículos realizem a leitura correta dos dados inseridos, independentemente da posição do veículo na via. Também deve-se atentar para a questão da segurança, com a aplicação de criptografia no chip RFID, para permitir apenas os agentes autorizados façam a reescrita dos dados presentes nos cartões.

Pode-se avaliar também a incorporação de sistemas colaborativos no sistema atual mediante interações entre os veículos conectados e as sinalizações de trânsito, por meio de um sistema de contratos e recompensas para informar aos motoristas sobre algum incidente que tenha ocorrido a metros ou quilômetros à frente, deste modo evitando acidentes e/ou engarrafamentos indesejados. Outra linha de pesquisa seria utilizar o JaCaMo [Boissier et al. 2013] e adotar o modelo organizacional Moise para fazer o gerenciamento das normativas de velocidade de trânsito.

Referências

Alecrim, E. and Marques, A. (2023). O que é RFID? Saiba como funciona essa tecnologia de conexão | Tecnoblog. Disponível em https://tecnoblog.net/responde/o-que-e-rfid-entenda-como-funciona-essa-tecnologia/. Acesso em 08/07/2024.

Barros., R. S., Heringer., V. H., Pantoja., C. E., Lazarin., N. M., and de Moraes., L. M. (2014). An Agent-oriented Ground Vehicle's Automation using Jason Framework. In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*, pages 261–266. INSTICC, SciTePress. DOI: https://doi.org/10.5220/0004917102610266.

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2013). Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78(6):747–761. Special section: The Programming Languages track at the 26th ACM Symposium on Applied Computing (SAC 2011) Special section on Agent-oriented Design Methods and Programming Techniques for Distributed Computing in Dynamic and Complex Environments. DOI: https://doi.org/10.1016/j.scico.2011.10.004.

Bordini, R., Hübner, J., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. Wiley.

Brandão, F. C., Lima, M. A. T., Pantoja, C. E., Zahn, J., and Viterbo, J. (2021). Engineering Approaches for Programming Agent-Based IoT Objects Using the Resource Management Architecture. *Sensors*, 21(23). DOI: https://doi.org/10.3390/s21238110.

Bratman, M. E., Israel, D. J., and Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355. DOI: https://doi.org/10.1111/j.1467-8640.1988.tb00284.x.

Heijmeijer, A. V. H. and Vaz Alves, G. (2018). Development of a Middleware between SUMO simulation tool and JaCaMo framework. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 7(2):5–15. DOI: https://doi.org/10.14201/ADCAIJ201872515.

Lara, R. (2020). Fiscais das ruas: radares de velocidade são mesmo precisos? | Tilt UOL. Disponível em https://www.uol.com.br/tilt/noticias/redacao/2020/11/19/fiscais-das-ruas-como-funcionam-os-radares-de-velocidade.htm. Acessado em 22/07/2023.

Lazarin, N., Pantoja, C., and Viterbo, J. (2023). Towards a Toolkit for Teaching AI Supported by Robotic-agents: Proposal and First Impressions. In *Anais do XXXI Workshop sobre Educação em Computação*, pages 20–29, Porto Alegre, RS, Brasil. SBC. DOI: https://doi.org/10.5753/wei.2023.229753.

Lazarin, N. M. and Pantoja, C. E. (2015). A robotic-agent platform for embedding software agents using Raspberry Pi and Arduino Boards. In *Proceedings of the 9th Workshop-School on Agents, Environments, and Applications (WESAAC 2015)*, pages 13–20, Niteroi. UFF. http://www2.ic.uff.br/~wesaac2015/Proceedings-WESAAC-2015.pdf.

Lazarin, N. M., Pantoja, C. E., and Viterbo, J. (2024). Dealing with the unpredictability of physical resources in real-world multi-agent systems. In Rocha, A. P., Steels, L., and van den Herik, J., editors, *Agents and Artificial Intelligence*, pages 48–71, Cham. Springer Nature Switzerland. DOI: https://doi.org/10.1007/978-3-031-55326-4_3.

Pantoja, C. E., Jesus, V. S. d., Lazarin, N. M., and Viterbo, J. (2023). A Spin-off Version of Jason for IoT and Embedded Multi-Agent Systems. In Naldi, M. C. and Bianchi, R. A. C., editors, *Intelligent Systems*, Lecture Notes in Computer Science, pages 382–396, Cham. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-45368-7_25.

Pantoja, C. E., Stabile, M. F., Lazarin, N. M., and Sichman, J. S. (2016). ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In Baldoni, M., Müller, J. P., Nunes, I., and Zalila-Wenkstern, R., editors, *Engineering Multi-Agent Systems*, pages 136–155, Cham. Springer International Publishing. https://doi.org/10.1007/978-3-319-50983-9_8.

Redação Mobilidade (2022). Reduzir a velocidade dos carros piora o congestionamento? | Mobilidade Estadão. Disponível em: https://mobilidade.estadao.com.br/mobilidade-com-seguranca/reduzir-a-velocidade-dos-carros-piora-o-congestionamento/ Acessado em: 22/07/2023.

Souza de Castro, L., Borges, A., and Vaz Alves, G. (2018). Developing a smart parking solution based on a Holonic Multiagent System using JaCaMo Framework. In *Proceedings of the 12th Workshop-School on Agents, Environments, and Applications (WESAAC 2018)*, Fortaleza. UFC. https://scholar.google.com.br/scholar?cluster=16564013293278034919.

Souza de Castro, L. F., Manoel, F. C. P. B., Souza de Jesus, V., Pantoja, C. E., Pinz Borges, A., and Vaz Alves, G. (2022). Integrating Embedded Multiagent Systems with Urban Simulation Tools and IoT Applications. *Revista de Informática Teórica e Aplicada*, 29(1):81–90. DOI: https://doi.org/10.22456/2175-2745.110837.

Souza de Jesus, V., Mori Lazarin, N., Pantoja, C. E., Vaz Alves, G., Ramos Alves de Lima, G., and Viterbo, J. (2023). An IDE to Support the Development of Embedded Multi-Agent Systems. In Mathieu, P., Dignum, F., Novais, P., and De la Prieta, F., editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*, pages 346–358, Cham. Springer Nature Switzerland. DOI: https://doi.org./10.1007/978-3-031-37616-0_29.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John wiley & sons.

Abordagens multiagentes para resiliência e reestabelecimento do fornecimento de energia elétrica: Uma perspectiva logística na colaboração entre bases de atendimento durante eventos climáticos severos

James Gustavo Black Rebelato[1], Cesar Augusto Tacla[1], Gustavo Giménez Lugo[1]

[1]Departamento Acadêmico de Informática – Programa de Pós-Graduação em Computação Aplicada - Universidade Tecnologica Federal do Paraná (UTFPR)
Av. Sete de Setembro, 3165 – Rebouças – 80230-901 – Curitiba – PR – Brazil

james.rebelato@copel.com, tacla@utfpr.edu.br, gustavogl@utfpr.edu.br

Abstract. *This article highlights the importance of multi-agent approaches in improving the resilience and rapid restoration of electricity supply in adverse weather conditions scenarios. Using real data from Companhia Paranaense de Energia (COPEL), the study presents simulations considering geolocation, affinities, and technical team overload. As a result, the collaboration between service centers promoted through multi-agent system modeling is emphasized, aiming to contribute to crisis management to address increasingly frequent climatic challenges.*[*]

Resumo. *Este artigo destaca a importância das abordagens multiagentes na melhoria da resiliência e na rápida restauração do fornecimento de energia elétrica em cenários de eventos climáticos adversos. Utilizando dados reais da Companhia Paranaense de energia (COPEL), o estudo apresenta simulações, considerando geolocalização, afinidades e sobrecarga das equipes técnicas. Como resultado é enfatizanda a colaboração entre as bases de atendimento que é promovida através da modelagem de sistemas multiagentes, buscando contribuir para a gestão de crises para enfrentar desafios climáticos cada vez mais frequentes.*

1. Introdução

Eventos climáticos severos (ECS) afetam a rede de distribuição de energia elétrica, provocando interrupções no fornecimento que impactam a população e a economia [Sales, 2024].

Durante os temporais, a quantidade de emergências e solicitações de atendimento aumenta exponencialmente, sobrecarregando as equipes de campo responsáveis pelo restabelecimento do serviço. Com o objetivo de reduzir o tempo de restabelecimento e minimizar os transtornos causados por tais eventos, é imperativo desenvolver estratégias eficazes de gestão e resposta [Simple, 2024].

Na literatura, o tema do restabelecimento do fornecimento de energia elétrica durante ECS é abordado de maneira ampla, destacando-se a importância do restabelecimento automático e da colaboração entre as equipes de campo. No entanto, há uma lacuna na compreensão das melhores práticas de coordenação entre as bases de

atendimento e as equipes de restauração de sistemas de distribuição de energia.

Neste contexto, este trabalho propõe a simulação da colaboração entre bases de atendimento e equipes de restauração de sistemas de distribuição de energia durante ECS. Através de simulações baseadas em Multiagentes, buscamos identificar estratégias eficazes de alocação de recursos e coordenação entre as equipes, visando otimizar o tempo de restabelecimento e minimizar os impactos sobre os consumidores.

Este artigo está organizado da seguinte forma: primeiramente, revisamos a literatura existente sobre o restabelecimento do fornecimento de energia elétrica dutante ECS; em seguida, apresentaremos a metodologia de simulação baseada em Multiagentes; depois, discutiremos os resultados obtidos e suas implicações para a gestão de crises; por fim, apresentaremos as conclusões e sugestões para pesquisas futuras.

2. Fundamentação Teórica

Nesta seção são apresentados os principais conceitos relacionados ao trabalho.

Dados georreferenciados - A disponibilidade e o uso de dados cartográficos e georreferenciados têm se tornado essenciais em uma variedade de campos, incluindo planejamento urbano, gestão ambiental, agricultura de precisão e monitoramento de desastres naturais. Esses dados, que podem incluir informações topográficas, imagens de satélite de alta resolução, dados de sensoriamento remoto e sistemas de informações geográficas (SIG), fornecem uma base sólida para a análise espacial e a tomada de decisões baseada em evidências [Oliveira, 2022].

Rede de Distribuição de energia elétrica - Constitui um sistema complexo que facilita o fornecimento de eletricidade aos consumidores finais. Essa rede inicia-se nas subestações, onde a tensão é reduzida para níveis adequados ao transporte seguro e eficiente. As subestações desempenham um papel crucial na distribuição de energia, permitindo a interligação entre diferentes circuitos e a manutenção da estabilidade do sistema. Equipamentos de proteção automáticos e manuais são essenciais para garantir a segurança e a confiabilidade da rede, detectando e isolando falhas e sobrecargas que possam ocorrer [IEEE, 2018]. Transformadores desempenham um papel fundamental na adaptação dos níveis de tensão, permitindo a transmissão eficiente da eletricidade ao longo da rede. Postes e cabos são os componentes físicos visíveis da rede, responsáveis por transportar a eletricidade até os consumidores finais, residenciais, comerciais e industriais [Gonen, 2014]. Esses elementos, em conjunto, formam uma infraestrutura vital para o funcionamento do sistema elétrico, garantindo o acesso confiável à energia elétrica em todo o território brasileiro.

Eventos Climáticos severos - Representam uma preocupação cada vez maior devido ao seu impacto significativo nas infraestruturas e na sociedade em geral. Esses eventos, como furacões, tempestades intensas, enchentes e secas prolongadas, têm sido objeto de estudo devido à sua frequência e intensidade crescentes, muitas vezes atribuídas às mudanças climáticas globais [IPCC, 2023]. O aumento da ocorrência e da intensidade desses eventos tem implicações diretas na segurança e na resiliência das comunidades, na agricultura, na segurança alimentar e na infraestrutura crítica, incluindo sistemas de distribuição de energia elétrica e redes de abastecimento de água [Fernandes, 2021]. A compreensão desses eventos e a implementação de medidas de adaptação e mitigação são essenciais para minimizar os impactos adversos e promover a sustentabilidade ambiental e socioeconômica.

Falta de energia e risco de vida - Durante ECS são geradas uma quantidade atípica de serviços emergenciais de falta de energia e risco de vida [CREA-PR, 2024]. De acordo com a Agência Nacional de Energia Elétrica (ANEEL), a ocorrência desses eventos tem impacto direto na qualidade do serviço de distribuição de energia, refletindo-se nos indicadores de Duração Equivalente de Interrupção por Unidade Consumidora (DEC) e Frequência Equivalente de Interrupção por Unidade Consumidora (FEC) [Schardong, 2020]. A resposta eficaz a essas situações de emergência é essencial para garantir a segurança dos cidadãos e a resiliência do sistema elétrico diante dos desafios impostos pelo clima extremo [ANEEL, 2024].

Resiliência do sistema de distribuição - É a capacidade do sistema de distribuição resistir, utilizando uma gama de estratégias destinadas a mitigar os impactos causados por ECS. Essa abordagem busca reduzir a probabilidade de falhas físicas nos elementos da rede, tornando-a mais resistente e capaz de absorver os efeitos adversos desses eventos. Esse conceito de resiliência é ilustrado pelo "*Trapézio da Resiliência*", demonstrado na Figura 1, que engloba as etapas de Preparação, Adaptação, Restauração e Recuperação. Esse modelo, fornece uma medida para avaliar o impacto temporal de um evento climático extremo, destacando a importância de medidas de aprimoramento para reduzir tanto o impacto inicial quanto o tempo de recuperação subsequente [Tierney e Bruneau, 2007].

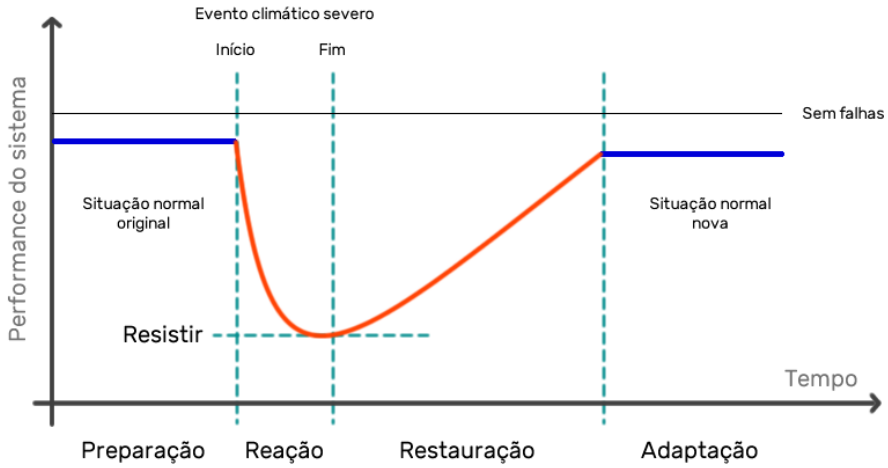


Figura 1. Trapézio da Resiliência
Fonte: Autoria própria (2024)

Restabelecimento do fornecimento de energia - Restabelecer a energia após uma interrupção é uma operação crucial que demanda a atuação ágil e coordenada de equipes de campo compostas por técnicos eletricistas [IBRE, 2024]. Esses profissionais desempenham um papel fundamental na identificação e correção de falhas na rede elétrica, seja por meio da reparação de equipamentos danificados, substituição de componentes danificados ou restauração de conexões elétricas [Sinapsis, 2024]. Munidos de conhecimento técnico especializado e equipamentos adequados, os técnicos eletricistas enfrentam condições adversas e desafios logísticos para garantir que o fornecimento de energia seja restabelecido o mais rapidamente possível, minimizando assim o impacto sobre os consumidores e contribuindo para a resiliência do sistema elétrico [Marques, 2018].

Sistemas Multiagentes (SMA) - Os sistemas multiagentes têm se mostrado uma abordagem poderosa e versátil para modelar e simular sistemas complexos onde múltiplos agentes autônomos interagem entre si e com o ambiente, visando objetivos

comuns ou individuais. Esses sistemas são fundamentais para a coordenação de comportamentos inteligentes entre agentes [Weiss, 1999]. O termo sistema multiagente é utiizado para carcterizar um sistema que contém vários agentes comunicando entre si com objetivo de executar tarefas [Wooldridge, 2002]. Os SMA integram técnicas para obter um comportamento global coerente e eficiente [Russell, 2010]. Finalmente, a importância das interações dinâmicas entre agentes é reforçada pela necessidade de maximizar a eficiência e a confiabilidade de sistemas complexos [Shoham e Leyton-Brown, 2008]. No contexto da distribuição de energia elétrica, os SMA têm se destacado na otimização da operação e controle de redes elétricas inteligentes (smart grids), onde agentes como consumidores, geradores, dispositivos de armazenamento de energia e sistemas de controle interagem dinamicamente para maximizar a eficiência e a confiabilidade do sistema [Campos, 2018].

NetLogo - É uma plataforma de modelagem e simulação baseada em agentes amplamente utilizada para explorar fenômenos complexos em diferentes domínios. Com uma interface intuitiva e flexível, o NetLogo permite aos usuários criar modelos de agentes com facilidade, representando interações entre entidades autônomas em um ambiente virtual. Essa ferramenta é especialmente útil para investigar comportamentos emergentes e padrões de sistemas dinâmicos, incluindo fenômenos sociais, biológicos e ambientais. Além disso, o NetLogo oferece uma variedade de recursos para análise e visualização de dados, facilitando a interpretação e a comunicação dos resultados obtidos por meio da simulação [Souza, 2019].

3. Metodologia desenvolvida

O propósito é a criação de um modelo de simulação baseado em dados reais obtidos da COPEL e SIMEPAR criando multiagentes no software NetLogo. O NetLogo foi escolhido como framework devido a grande quantidade de exemplos e alinhamento com a linguagem de programação Java. Essa abordagem visa garantir a fidelidade e a representatividade dos cenários climáticos e operacionais simulados, permitindo uma análise precisa dos impactos de ECS na rede de distribuição de energia elétrica. Segue abaixo as etapas realizadas para criar e carregar o modelo com seus multiplos agentes:

3.1. Inicialização do modelo

- Bases de atendimento: Inicialmente, foram criadas as bases de atendimento georeferenciadas com latitude e longitude. Essas bases foram selecionadas levando em consideração critérios como cobertura geográfica, densidade populacional e infraestrutura disponível, visando garantir uma distribuição equitativa e eficiente dos recursos. Para a representação atual foram utilizadas as 41 agências de atendimento [COPEL, 2024].

- Conectividade entre as bases de atendimento: Em seguida, estabeleceu-se a conexão entre as bases de atendimento, que representa lógica de apoio entre as bases. Cada base tem ao menos uma outra base que pode emprestar ou solicitar apoio quando necessário. As bases foram conectadas seguindo o critério de proximidade geográfica, facilidade de acesso pelas rodovias e agrupamento e mesma Regional de atendimento (Leste, Oeste, Norte, Noroeste e Centro-Sul).

- Equipes técnicas de campo: Cada equipe é formada por um veículo, eletricistas, equipamentos de comunicação e manutenção. Todas as equipes pertencem a uma base de atendimento, realizam o deslocamento e atendimento das emergências e retornam no final do turno para base. Normalmente existem de 5 a 20 equipes para cada

base de atendimento.

- Impactos do ECS convertidos em emergências: A partir dos dados meteorológicos fornecidos pelo SIMEPAR, foram simulados os impactos e na sequência convertidos em situações de emergência, como desligamentos de equipamentos de proteção da rede e danos estruturais, permitindo uma análise detalhada dos cenários de crise [SIMEPAR, 2024].

- Execução das emergências: A função objetivo fundamental dos agentes, que são as equipes de campo, é executarem todas as emergencias pendentes da base de atendimento. Após executarem todas as emergências ou finalizar o turno da equipe elas devem retornar para base de atendimento. Dependendo do cenário de atendimento da simulação esse comportamento será alterado para que as equipes possam apoiar outras bases de atendimento antes do final de seu turno de trabalho.

A implementação do modelo e simulação dos cenários de atendiemnto foi realizada multiplos agentes no NetLogo:

Turtles - Agentes existentes no mundo representando equipes e emergências;

Patches – São agentes estacionários e representam um local no espaço: impactos e bases de atendimento;

Links – Conectividade entre bases de atendimento, equipes e emergências;

Observer - que contempla o ambiente formado pelos *turtles e patches*.

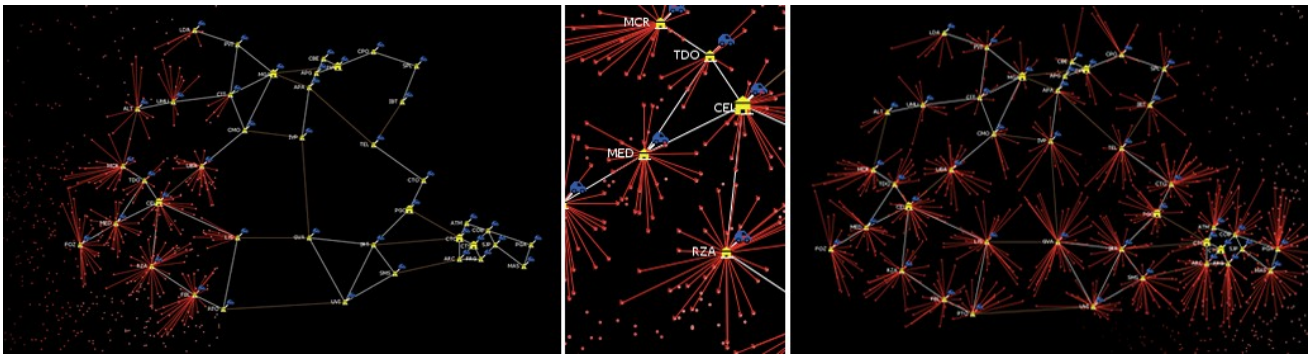


Figura 2. Bases de atendimento e início do ECS; Destaque do local; Final do ECS, impactos convertido em Emergências
Fonte: Autoria própria (2024)

A Figura 2, exibe a inicialização do modelo, com a localização das bases de atendimento e a conectividade entre elas, assim como ilustra, da esquerda para direita, a passagem de um ECS ocorrendo no estado do Paraná e na sequência os impactos são convertidos em emergências. Na imagem central é possível visualizar com mais detalhes as emergências e as linhas de conexão, em vermelho, até as bases de atendimento.

3.2. Desenvolvimento do modelo de simulação

Foram conduzidas simulações comparativas para avaliar diferentes estratégias de gestão de recursos durante ECS. A Figura 3 demonstra a tela do NetLogo, onde foi criado o modelo que representa as bases de atendimento e equipes de campo responsáveis pela restauração do fornecimento de energia elétrica.

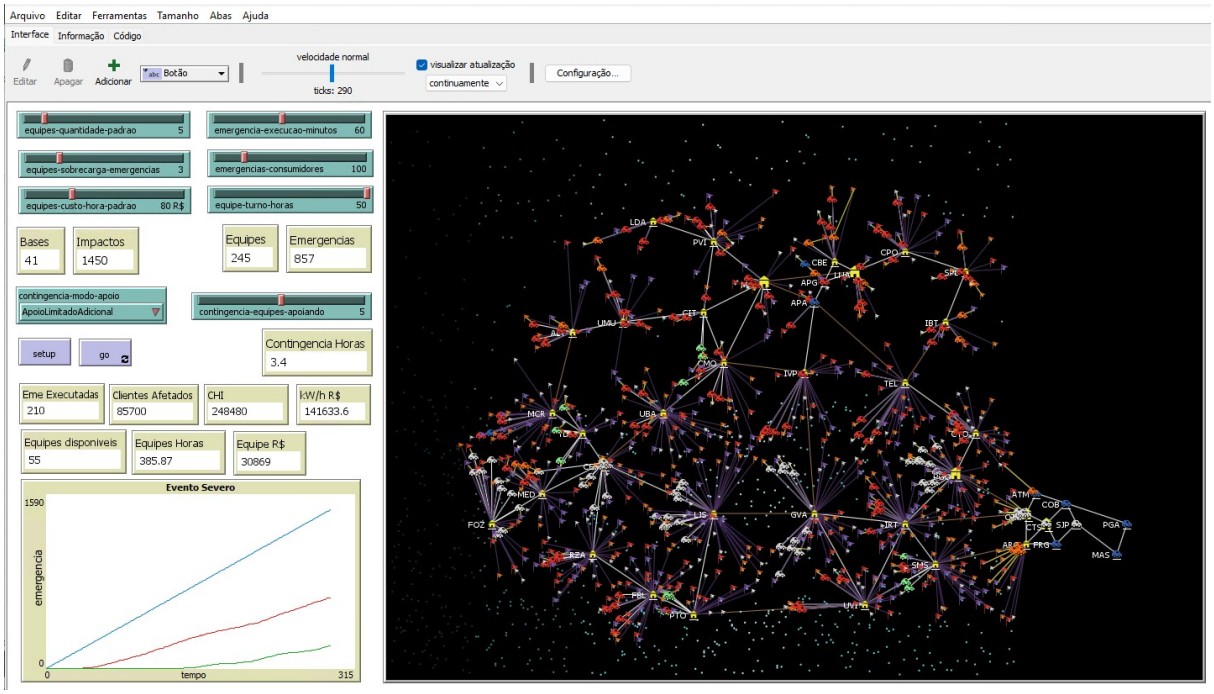


Figura 3. Tela do ambiente desenvolvido para realizar as simulações

Fonte: Autoria própria (2024)

3.3. Parâmetros fixos

Para que as simulações pudessem ser comparadas optou-se por manter fixos e constantes o valor de alguns parâmetros como:

- Bases de atendimento – 41

- Quantidade total de equipes – 205 (5 por base de atendimento)

- Total de emergências após impacto do ECS – 1.500

- Tempo médio individual de execução da emergência – 1 hora

- Quantidade de consumidores afetados por emergência – 100

- Custo da equipe por hora de trabalho – R$ 80,00

- Valor do kWh para consumidores desligados – R$ 0,57

Os valores acima foram estipulados através da média de situações ocorridas do período de Maio de 2023 a Maio de 2024. Todos os valores podem ser ajustados a cada novo ciclo de simulações comparativas.

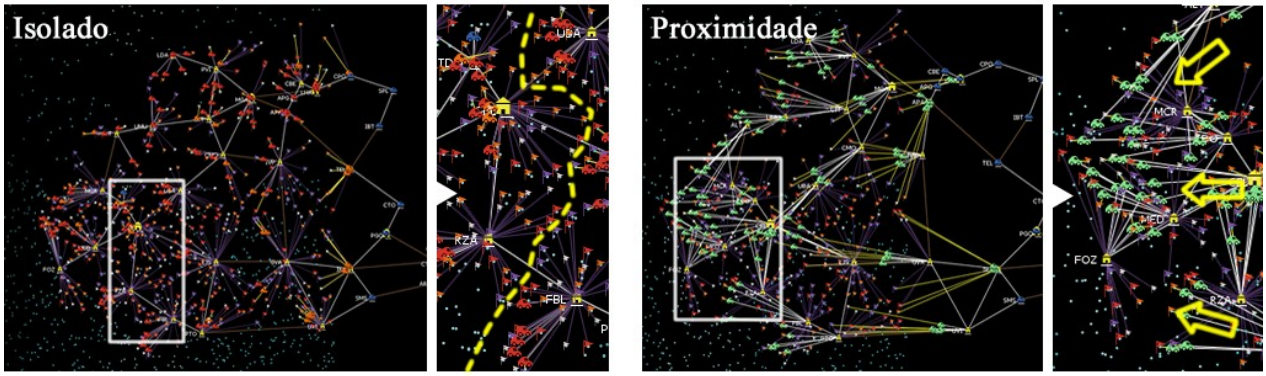
3.4. Cenários de atendimento

Foram configurados 4 cenários com formas de atendimento diferenciadas:

- **Isolado**: Atendimento ilhado sem solicitar ou fornecer apoio. As equipes pertencentes a cada base atendem apenas as emergências da própria base;

- **Proximidade:** Todas equipes se deslocar para atender demandas existentes. As equipes de qualquer base de atendimento se desloca assim que é identificada a necessidade de atendimento das emergências existentes;

Figura 4. Visualização e destaque da simulação Isolado e Proximidade



Fonte: Autoria própria (2024)

Na Figura 4, no detalhe lateral esquerdo, a linha amarela evidencia a clara separação do atendimento onde as equipes de cada base não cruzam sua área de abrangência. Na lateral direita da imagem, no cenário Proximidade as setas amarelas indicam o deslocamento das equipes com ícone verde que são as equipes emprestadas.

- **Apoio solicitado:** Até 50% das equipes da base podem ser emprestadas temporariamente desde que a base vizinha que precisa de apoio ultrapassar 3 emergências por equipe;

- **Apoio limitado + adicional:** Apoio de empréstimo de equipes limitado em 30% para bases vizinhas e um incremento de 50% de equipes a disposição antes do ECS.

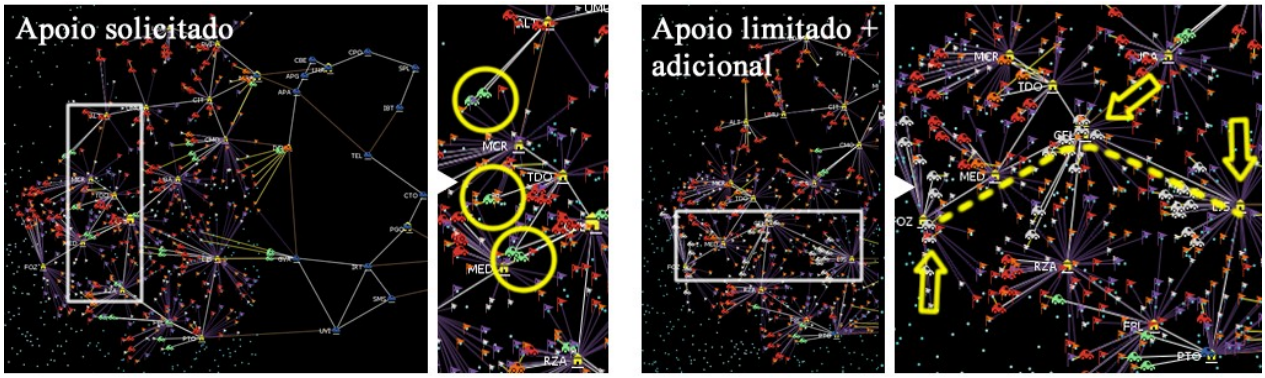


Figura 5. Visualização e destaque da simulação Apoio solicitado e Apoio limitado + adicional

Fonte: Autoria própria (2024)

Na Figura 5, no detalhe lateral esquerdo, os círculos amarelos evidenciam que apenas algumas equipes foram emprestadas para as bases vizinhas. Na lateral direita da imagem, as setas e a linha amarela indicam que as equipes adicionais, ícone branco, acompanharam a evolução do ECS e algumas equipes, ícone em verde, foram emprestadas paras as bases de atendimento vizinhas.

3.5. Indicadores de atendimento

Foram especificados indicadores para mensurar os possíveis ganhos obtidos em cada simulação de atendimento:

- Tempo em horas da duração do ECS (acima de 100 emergências existentes);

- Quantidade de consumidores hora desligados (CHI);

- Custo financeiro total. Horas de equipes de campo + kWh sem cobrança;

4. Resultados

A Tabela 1, demonstra os valores obtidos das simulações. Cada item na tabela representa parâmetros cruciais para a simulação no NetLogo, proporcionando informações valiosas sobre a melhor forma de colaboração entre os agentes em diferentes cenários.

Tabela 1. Comparativo entre as simulações

Simulação	Duração Total ECS	Consumidores Hora Desligados	Custo Equipes + kWh sem cobrança
Isolado	13h15	1.193.553	R$ 817.644,40
Proximidade	12h50	1.191.610	R$ 822.330,70
Apoio solicitado	13h00	1.207.193	R$ 837.320,20
Apoio limitado + adicional	11h30	1.048.893	R$ 749.997,20

Os experimentos realizados demonstraram uma redução de até 9% no custo financeiro total de atendimento durante ECS. Por meio da otimização na alocação de recursos e na coordenação entre as bases de atendimento, foi possível minimizar os gastos operacionais sem comprometer a eficácia das operações de resposta a emergências.

O resultado mais importante do experimento foi a redução em 15% do tempo de restabelecimento do sistema de distribuição de energia elétrica. A implementação de medidas preventivas e a adoção de uma abordagem adaptativa e dinâmica possibilitaram uma resposta mais ágil e eficaz às emergências, resultando em uma rápida normalização do serviço para os consumidores afetados.

Algumas hipóteses puderam ser validadas e outras constatações foram obtidas através do acompanhamento das simulações:

- Durante a fase de preparação, a antecipação na alocação de recursos é crucial para uma resposta eficaz durante ECS. Direcionar previamente os recursos permite que as equipes estejam prontas para agir imediatamente, minimizando o tempo de resposta e maximizando a eficiência das operações de atendimento.

- Quando identificada uma demanda excessiva, é essencial diminuir dinamicamente o tamanho da abrangência da base de atendimento. Essa estratégia permite concentrar os recursos disponíveis em áreas prioritárias, garantindo uma resposta mais rápida e eficiente às emergências.

- Determinar corretamente a conectividade entre as bases de atendimento é fundamental para otimizar a colaboração entre elas. Isso permite identificar quais bases têm relações de apoio mais fortes e mais fracas, portanto, estão mais bem posicionadas para colaborar e compartilhar equipes durante ECS.

- Da mesma forma, calcular previamente os pontos mais frágeis de atendimento das bases é essencial para uma gestão eficiente de recursos. Ao identificar os pontos vulneráveis, as equipes podem priorizar ações preventivas e estratégias de reforço, reduzindo o risco de interrupções no fornecimento de energia.

- O cálculo antecipado de atendimento de demanda possibilita o empréstimo de recursos, garantindo que as emergências sejam atendidas de acordo com sua gravidade e impacto potencial.

Por fim, as múltiplas simulações com variações de dados de entrada e forma de atendimento permitiu compreender melhor as múltiplas possibilidade de otimizar a distribuição de recursos para garantir uma resposta eficaz em diferentes cenários de emergência.

4.1. Constatações complementares

Seguem algumas observações identificadas durante a realização das simulações:

Sobrecarga – Durante a simulação, foi possível identificar e quantificar a sobrecarga enfrentada pelas bases de atendimento e pelas equipes de campo. Utilizando métricas específicas, como o número de emergências não atendidas dentro de um determinado período de tempo, foi possível realizar um somatório preciso da sobrecarga enfrentada por cada uma das equipes. Essa análise detalhada permitiu uma compreensão mais clara da distribuição de tarefas e recursos, facilitando a identificação de áreas de maior pressão e possibilitando a tomada de decisões mais eficazes para mitigar a sobrecarga.

Forma de atendimento da base - Durante o estudo, cada base de atendimento foi categorizada com base na forma como realizava suas operações durante ECS. Essa categorização permitiu uma análise detalhada das diferentes estratégias adotadas pelas bases de atendimento. Vale ressaltar que nem sempre deslocar todos os recursos disponíveis para apoio é a melhor opção, uma vez que isso pode prejudicar o atendimento da base de origem no dia seguinte, comprometendo a capacidade de resposta a emergências futuras.

Ponto de decisão - Um dos pontos críticos identificados durante a simulação foi o momento em que se tornava essencial realizar o empréstimo de equipes entre bases de atendimento. Esse ponto de decisão foi determinado com base em uma análise cuidadosa da distribuição de demandas de atendimento e da capacidade operacional de cada base. Quando uma base alcançava uma sobrecarga crítica e não conseguia atender adequadamente às demandas, tornava-se necessário acionar o empréstimo de equipes, redistribuindo recursos de outras bases para suprir a demanda excessiva. A identificação desse ponto de decisão foi fundamental para garantir uma resposta eficiente e rápida às emergências, minimizando os impactos causados pelos ECS.

5. Conclusão

A simulação utilizando multiagentes demonstrou-se adequada para representar a complexidade do problema abordado neste estudo. Foi possível modelar os comportamentos individuais e interações entre os diversos elementos do sistema de distribuição de energia elétrica, fornecendo uma representação mais realista e dinâmica do período completo do ECS. Além disso, a flexibilidade e adaptabilidade inerentes aos sistemas multiagentes permitiram a incorporação de cenários variados e a avaliação de diferentes estratégias de resposta, contribuindo para uma análise abrangente e abordagem integrada do problema.

De acordo com os resultados obtidos, foi possível evidenciar que a colaboração entre as bases de atendimento é uma estratégia eficaz e promissora para o gerenciamento de crises e resposta a emergências no sistema de distribuição de energia elétrica. Através

da modelagem e simulação das interações entre as bases de atendimento e equipes técnicas de campo, foi observado que a coordenação e troca de recursos entre os diferentes pontos de operação podem melhorar significativamente a eficiência e a eficácia das operações de atendimento sem causar sobrecarga dos recursos. Esses achados destacam a importância da colaboração e cooperação entre as entidades envolvidas na gestão de crises, promovendo uma resposta mais rápida, coordenada e efetiva diante de ECS.

Este trabalho buscou contribuir, no contexto de modelos multiagentes de simulação aplicados à gestão de crises em infraestruturas críticas. Ao combinar técnicas de modelagem e simulação com a análise de cenários complexos do mundo real, espera-se fornecer percepções valiosas e metodologia prática para o desenvolvimento de sistemas inteligentes e adaptativos para gestão de crises em ECS.

6. Trabalhos futuros

Explorar as possibilidades de otimização da utilização das equipes de campo mediante o emprego de técnicas de machine learning representa uma direção promissora para futuros estudos. Conectar o modelo de simulação com dados em tempo real dos possíveis impactos dos ECS e das emergências existentes em atendimento. Carregar os dados históricos de cada base de atendimento treinar o modelo para realizar simulações personalizadas.

Referências

ANEEL (2024) “Workshop promovido pela ANEEL destaca a importância da preparação do setor elétrico diante de fenômenos climáticos intensos”, Agência Nacional de Energia Elétrica (ANEEL), https://www.gov.br/aneel/pt-br/assuntos/noticias/2024/workshop-promovido-pela-aneel-destaca-a-importancia-da-preparacao-do-setor-eletrico-diante-de-fenomenos-climaticos-intensos, acessado em 10 de maio de 2024.

Campos, Í. R. da C. (2018). "Aplicação de sistemas multiagentes ao problema de autorrecuperação em sistemas elétricos de distribuição do tipo smart grid". Universidade Federal do Pará. Disponível em: https://bdm.ufpa.br/jspui/ handle/prefix/1340, acessado em 10 de maio de 2024.

CREA-PR (2024) “Copel detalha medidas preventivas”, Revista do Conselho Regional de Engenharia e Agronomia do Paraná (CREA-PR), https://revista.crea-pr.org.br/copel-detalha-medidas-preventivas/, acessado em 10 de maio de 2024.

COPEL (2024) Agências de atendimento, Copel Distribuição, https://www.copel.com/site/copel-distribuicao/agencias-de-atendimento/, acessado em 5 de maio de 2024.

Fernandes, T., Hacon, S. de S. e Novais, J.W.Z. (2021). MUDANÇAS CLIMÁTICAS, POLUIÇÃO DO AR E REPERCUSSÕES NA SAÚDE HUMANA: REVISÃO SISTEMÁTICA. Revista Brasileira de Climatologia. (abr. 2021), 138–164.

Gonen, T. (2014). "Electric Power Distribution System Engineerin." CRC Press.

IBRE (Instituto Brasileiro de Economia) - FGV (2024) “Respostas impulsivas e impensadas: eventos climáticos extremos”, Blog do IBRE - FGV, https://blogdoibre.fgv.br/posts/respostas-impulsivas-e-impensadas-eventos-climaticos-extremos, acessado em 10 de maio de 2024.

IEEE Power & Energy Society. (2018). "Understanding Electric Power Systems: An Overview of the Technology and the Marketplace." Wiley.

IPCC. Climate Change: The Physical Science Basis (2023). Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press.

Marques, Leandro Tolomeu (2024). Restabelecimento de energia em sistemas de distribuição considerando aspectos práticos [doi:10.11606/T.18.2018.tde-26072018-134924]. São Carlos: Escola de Engenharia de São Carlos, Universidade de São Paulo, (2018). Tese de Doutorado em Sistemas Elétricos de Potência, acessado em 10 de maio de 2024.

Oliveira Filho, S. F., Fernandes, A. C. G., Borges, I. M. S., Santos, A. F. L. dos, Campos, J. O., Silva, E. C. B. da, Martins, M. S., Silva, J. A. da, Paiva, C. R. B. de, Silva, J. A., Santos, M. J. R., & Maciel, J. K. V. S. (2022). Use of the GIS tool for monitoring crop development in the semi-arid region of Paraíba. Research, Society and Development, 11 (4), e53611427728. DOI: https://doi.org/10.33448/rsd-v11i4.27728.

Russel, S., Norvig (2010), P. Artificial Intelligence: A Modern Approach. 3a Edition. Prentice Hall. ISBN-13: 978-0-13-604259-4.

Sales, C. e Uhlig, A. (2024) “Eventos climáticos extremos e o suprimento de eletricidade”, Broadcast Energia, https://acendebrasil.com.br/artigo/eventos-climaticos-extremos-e-o-suprimento-de-eletricidade/, acessado em 10 de maio de 2024.

Schardong, Bianca Jupiara Fortes; Garcia, Vinícius Jacques; Kiefer, Gabriela Sanson; Pinto, Nelson Guilherme Machado. Otimização do Atendimento a Emergências no Setor Elétrico. Boletim de Conjuntura (BOCA), v. 18, n. 52, p. 82-115, 2024. (2024). Disponível em: https://doi.org/10.5281/zenodo.11003159, acesso em 10 de maio de 2024.

Shoham, Y. e Leyton-Brown, K. Multiagent Systems: Algorithmic, game-theoretic, and logical foundations. New York: Cambridge University Press, (2008).

SIMEPAR (2024) Dados das estações, Sistema de tecnologia e Monitoramento do Paraná, http://www.simepar.br/prognozweb/simepar/dados_estacoes/25264916, acessado em 6 de maio de 2024.

Simple Energy (2024) “Quais os efeitos dos eventos climáticos no fornecimento de energia?”, Simple Energy, https://simpleenergy.com.br/quais-os-efeitos-dos-eventos-climaticos-no-fornecimento-de-energia/, acessado em 10 de maio de 2024.

Sinapsis Energia (2024) “Como as mudanças climáticas impactam nossas redes de distribuição”, Sinapsis Energia, https://sinapsisenergia.com.br/como-as-mudancas-climaticas-impactam-nossas-redes-de-distribuicao/, acessado em 10 de maio de 2024.

Souza, Jusciel Kvan Gomes de. Modelagem baseada em agentes: possibilidades na Educação Matemática e pesquisa ambiental. (2019). 88 f. Monografia (Graduação) - Curso de Matemática, Universidade Federal do Tocantins, Araguaína, 2019.

Tierney, K. and Bruneau, M. (2007) Conceptualizing and Measuring Resilience: A Key to Disaster Loss Reduction, TR News, May-June, 250, 14-17.

Weiss, Gerhard (1999). Multiagent Systems: a modern approach to distributed artificial intelligence. MIT press, 1999.

Wooldridge, M. An Introduction to MultiAgent Systems. Department of

Computer Science, University of Liverpool, UK. John Wiley & Sons, LTD. (2002 [2009]).

Multi-agent System Architectural Aspects for Continuous Replanning

Carlos Joel Tavares[1], **Célia Ghedini Ralha**[1]

[1]Computer Science Department – Exact Sciences Institute – University of Brasília
Campus Darcy Ribeiro, 70.904-970 Brasília, Brazil

carlosjoel.tavares@gmail.com, ghedini@unb.br

Abstract. *Robots' coordination to achieve the system's goal is one of the challenges that complex Multi-Robot Systems (MRS) encounter. One could use automated planning (AP) to better face this challenge by diminishing problems and continually correcting the execution when failures occur. Some works in the literature try to fix this problem, but there are still few, and there's not much analysis between them. This work implements a Multi-Agent System (MAS) to simulate an MRS mission using a MAS architecture integrated with AP illustrated with space resource gathering robots. The results show the importance of the ability to plan recovery and research in complex space missions field.*

1. Introduction

Multi-robot Systems (MRS) are complex and need real-world environment execution [Klavins 2004], [Aziz et al. 2021]. Nevertheless, it is not always feasible to prepare for all environmental changes before the deployment of the system. Moreover, coordinating heterogeneous robots is a demanding task. One viable solution that creates the optimal plan recovers it when a failure occurs, and diminishes that demand is Automated Planning (AP). Therefore, the process of plan recovery becomes needed when systems run in dynamic environments [Schmitt et al. 2019], [Moreira and Ralha 2021b], [Moreira and Ralha 2022b], [da Silva and Ralha 2023].

The indispensable coordination of robots needed to achieve the system's goal is one of the many difficulties of MRS [Verma and Ranga 2021]. MRS' coordination, communication, and interaction between agents pose a problem similar to Multi-Agent Systems (MAS) [Wooldridge 2009], [Weiss 2016], [Salzman and Stern 2020]. While the formal definition of a planning problem, which includes a tuple composed of actions, propositions, initial state, and goal, helps to mitigate this problem, in classical planning, dynamic environments are not the focus, and such are the MRS's real-world environments. While planning solutions focus on changes that derive from internal actions, dynamic-focused solutions also focus on exogenous actions and action failures. That inserts the need for plan recovery since that plan can become unfeasible many times. This complex scenario relates to Multi-Agent Planning (MAP), which involves the coordination of resources and activities of many agents [Komenda et al. 2016], [Moreira and Ralha 2021a], [Moreira and Ralha 2022a].

There are studies focused on problems of coordination aligned with planning [Cashmore et al. 2015], [González et al. 2020], [Bischoff et al. 2021], [Martín et al. 2021], [Lesire et al. 2022]. The proposed solutions in the studies create a centralized analysis of the environment to coordinate the plan. The studies of MRS analyze many aspects like

goal decomposition, task allocation strategies, quality of attributes/plan adaptation using probabilistic and temporal planning, interactive coordination of heterogeneous robotic teams, relating architectures, frameworks, and robot operating systems. However, the recovery of plans in dynamic environments is seldom the focus. Even so, the literature discusses solutions to the adaptation of robotic missions [et al. 2021], [Carreno et al. 2022], design patterns of architectures to heterogeneous robots [Rodrigues et al. 2022] and planning agent architectures [Silva et al. 2020], [Magnaguagno et al. 2022].

Multi-robot planning in space robotics is crucial for enhancing efficiency, reliability, and productivity in space missions [Sun et al. 2023]. By enabling multiple robots to work collaboratively, tasks can be completed faster and more effectively, with specialized robots handling specific functions [Basmadji et al. 2020]. This approach provides redundancy, ensuring mission continuity even if one robot fails, and facilitates the execution of complex tasks that require precise coordination. This work's main contribution is the implementation of a MAS simulating a robotic space mission using the architecture defined in [da Silva 2024] with code available to promote open science (https://github.com/CJTS/city-planning).

The rest of the article includes: in Section 2, we show architectural aspects of MAS with AP; in Section 3, we display the experiments together with the used illustrative example; and lastly, in Section 4 the conclusions and future work.

2. Architectural Aspects

One aspect of MRS is the complex task-completing process which is viewed as decoupling complex tasks into simpler sub-tasks, forming a coalition of robots that will perform them, allocating to the coalitions, and using MAP to transform those tasks into sequential actions so the robots can perform [Rizk et al. 2019]. Figure 1 presents an adapted MRS workflow [Kiener and von Stryk 2010]. Note the workflow includes a human expert to decompose complex tasks into simpler sub-tasks based on the robots' capabilities available and the coalition formation of a set of agents. Then, the task (re-)allocation and robot planning and control steps are autonomously performed by the robot teams.

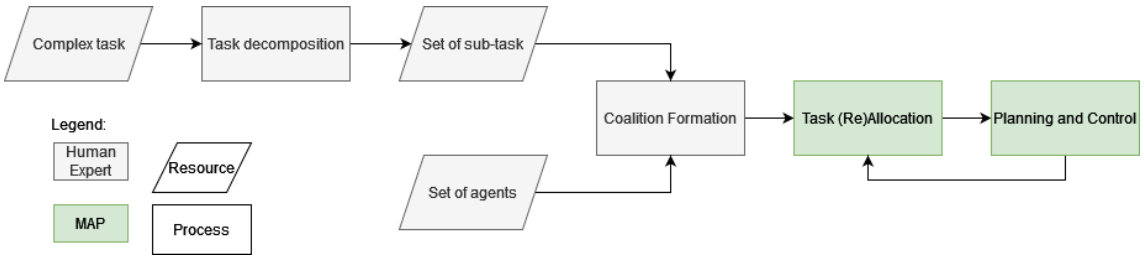


Figure 1. Adapted MRS workflow. Source: [Rizk et al. 2019].

The architecture used in this work is presented in [da Silva 2024], which integrates AP in MAS. The architecture's requirements involve the mission's decomposition into local plans to be executed by the agents. The *Coordinator* is responsible for receiving the mission request that contains the plans assigned to robot roles. Then, it coordinates the execution by assigning agents that fit the roles. It focuses on recovering the mission's plan when problems happen.

2.1. Design

As illustrated in Figure 2, the design phase necessitates a domain expert to outline the mission requirements. The *System Integrator* task is to develop the *Problem Domain* application component in planner syntax. This element is crucial to the architecture, as the effectiveness of the planning capabilities hinges on the formal definition of the problem and the domain. Goal reasoning functions may be incorporated later to address issues stemming from inadequate descriptions in this component.

The *Coordinator* and *Robot* runtime components exchange data on local plans and mission properties via messages. Establishing a communication protocol between these components is crucial.

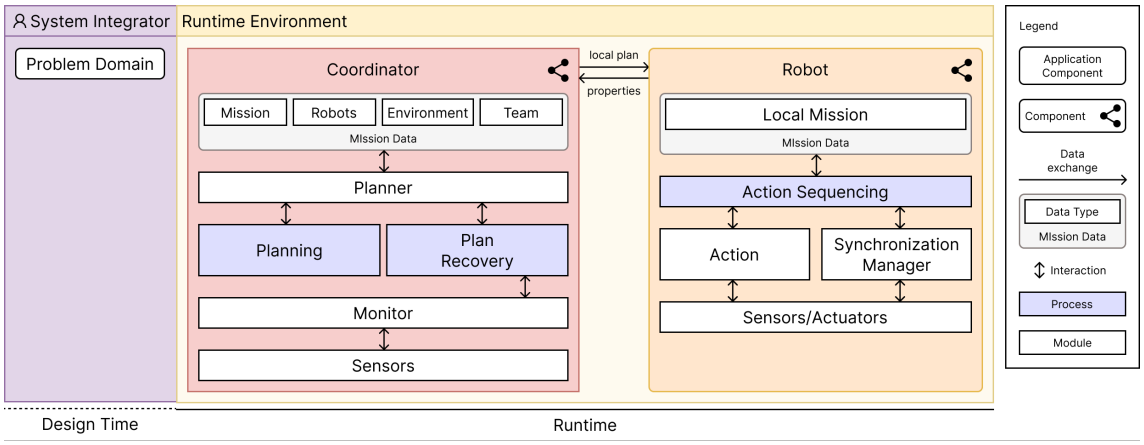


Figure 2. The high-level architecture. Source: [da Silva 2024].

The *Coordinator* oversees mission control, encompassing planning and plan recovery. This component stores the *Mission Data* required for mission execution, which includes the mission details, available robots, and the environment state. The *Coordinator Planner* module generates a mission plan through the *Planning* process using the automated planner, environment state, and problem domain. Subsequently, a planning execution cycle commences, where the *Coordinator* monitors the environment for unexpected changes via the *Monitor* and *Sensors* modules and receives feedback from the robots.

Once the *Local Mission* plan is established, the robots receive the plan from the *Coordinator* and initiate execution by carrying out actions sequentially through their *Action Sequencing* process. The *Robot Action* module executes each action using their *Sensors/Actuators* devices. Additionally, the *Synchronization Manager* module ensures coordination among actions performed by multiple robots. It's important to note that the plan is dynamically defined at runtime by the *Coordinator* component. In the event of a problem, the *Coordinator* reevaluates the original plan for redistribution to the robots, aiming to minimize disruptions in the MRS.

2.2. Execution Process

Figure 3 presents the execution workflow of the architecture. The initial step of the execution process is the initial trigger, determined by the system integrator and varies depending

on the domain. Upon receiving the initial trigger, the *Coordinator* adjusts the initial state, defines the mission, and initiates preparations for creating the plan to accomplish it. Consequently, the *Coordinator* must comprehend, based on the problem domain, the required robot types to assemble the team.

Utilizing the information within the *Mission Data*, the *Planner* module initiates the *Planning* process to generate the optimal plan. In this work, we employed the HyperTensioN planner for this purpose [Magnaguagno et al. 2022]. However, alternative planners can also be considered, with careful evaluation of their capabilities beforehand. For a comprehensive comparison between planners, see [Georgievski and Aiello 2015].

In the sequence, the *Coordinator* divides the plan among the robots in the team and dispatches their respective local missions. Each robot initiates its task sequencing process to execute the plan. Concurrently, the *Coordinator* monitors the environment for any changes that could jeopardize the plan's feasibility. If an issue occurs, the *Coordinator* initiates the plan recovery process, which involves rectifying the current state of the environment and subsequently generating a new plan (replan) for the mission. The plan is successfully executed by the team of robots if no unexpected events arise.

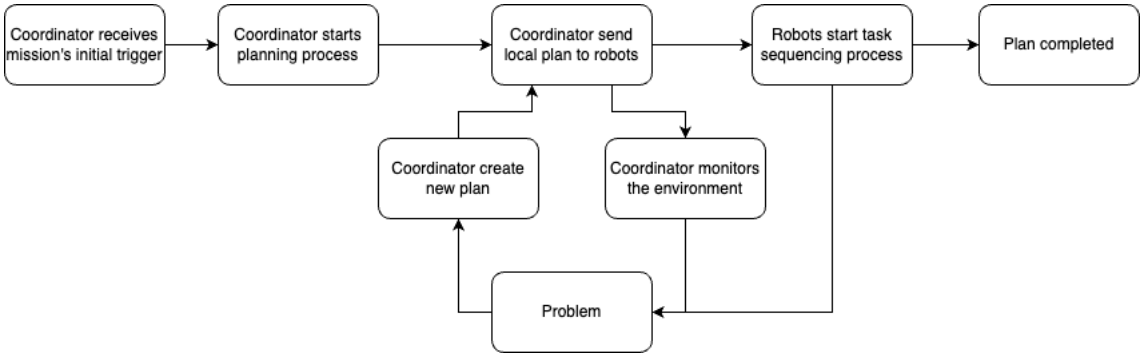


Figure 3. The solution's execution process. Source: [da Silva 2024]

Plan Recovery

One of the reasons the architecture might need to execute a plan recovery process is when all actions yield the desired outcome, but, an unexpected event occurs in a detectable manner. An example of such an event is the robot's battery depletion. Since most planners do not incorporate numeric values in their domain modeling, typically, a boolean flag indicates the battery level (e.g., low). Most actions necessitate this flag to indicate sufficient battery power. During execution, if the flag indicates a low battery level, the system must replan to include a task for recharging the robot. This scenario is not necessarily an error but rather an event that could occur at any given time.

Another scenario requiring plan recovery arises when there's an issue with the domain modeling. For instance, if a car is supposed to wait for a crane to finish loading all boxes before unloading, but the modeling precondition erroneously allows unloading if no boxes are present, the car may prematurely vacate the docking area. Consequently, the initial plan, which only considered one car navigation, must be replanned to ensure the car returns for the remaining boxes. This situation underscores the necessity of plan

recovery to rectify discrepancies in the domain modeling and ensure plan adherence.

The second scenario necessitating plan recovery arises from external events or agents interacting with the environment in a manner that unpredictably alters its state. For instance, consider a scenario where a door needs to remain open for a robot to pass through, but someone unexpectedly closes it. This unforeseen event disrupts the execution of the plan, prompting the need for plan recovery to address the changed environmental condition and revise the plan accordingly.

Regardless of the reason for replanning, the initial trigger is when the robot interacts with the environment. This interaction marks the commencement of the replan cycle, which can be visualized as a reactive replan process, as depicted in Figure 4 using a UML sequence diagram [Rumbaugh et al. 2004], [Object Management Group (OMG) 2015]. The reactive replan process transpires when the *Coordinator* receives the status of the environment after the robot attempts to act, and a problem emerges.

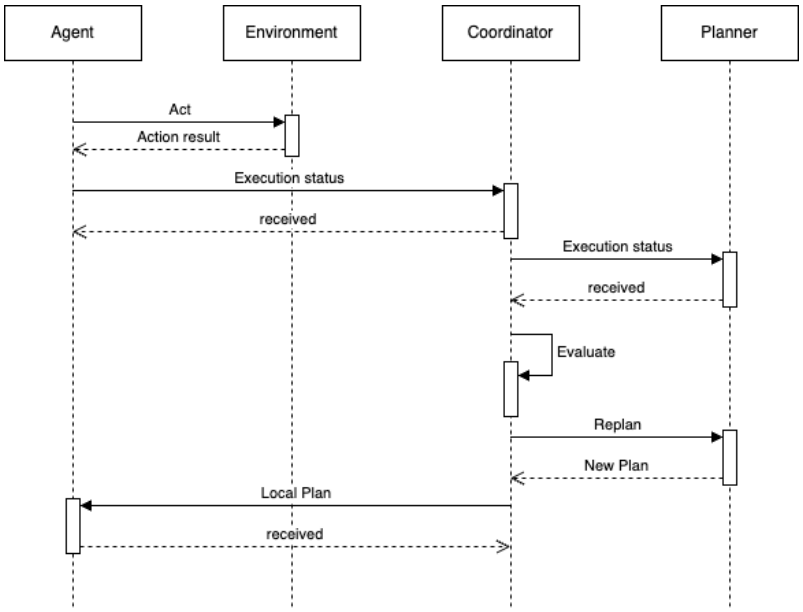


Figure 4. The architecture reactive replan sequence diagram.

When the environment status confirms the action is successful, the plan's execution can proceed uninterrupted. However, if the robot encounters difficulty completing the action, the *Coordinator* must ascertain the reason for the failure. The robot communicates the problem through an execution status message. Subsequently, the *Coordinator* updates the environment state model and requests a new plan from the planner to address the encountered issue. This iterative process enables the architecture to respond dynamically to environment changes and ensure plan adherence.

Dealing with false action results can indeed pose significant challenges. These issues arise when sensors erroneously indicate that an action was successful when it was not, or conversely, signal a problem when there isn't one. Resolving false action results typically involves two approaches, each with its complexities.

The first approach entails the *Coordinator* validating all actions to ensure their

accuracy. However, this method can incur additional hardware costs, such as employing sensors to verify all action effects. Alternatively, the second approach needs a sophisticated heuristic, such as a backtracking algorithm that stores execution traces. This method enables the *Coordinator* to backtrack to the point where the problem occurred when the action's effects are needed.

At the current stage, the proposed architecture assumes that agents cooperate with veracity behavior when exchanging messages. Thus, addressing false action results was left as an avenue for future exploration. Such exploration underscores the need for further research to develop robust mechanisms for identifying and mitigating false action results within the architecture.

Figure 5 depicts a proactive replan process, where the *Coordinator* actively monitors the environment, preemptively identifying potential problems. In this process, the *Coordinator* maps the environment states, evaluates them to detect issues, updates the planner model as necessary, requests a new plan, and distributes the updated local plans to the agents for execution.

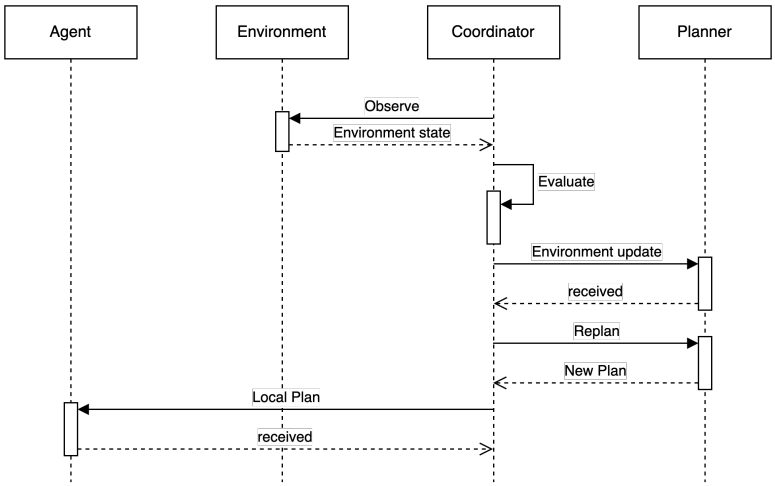


Figure 5. The architecture proactive replan sequence diagram.

This work extends beyond traditional plan recovery by addressing planning in situations where unexpected events occur with reactive (Figure 4) or proactive (Figure 5) replan, leading to changes in the environmental state. By encompassing both aspects, this architecture offers a comprehensive approach to handling dynamic and unpredictable scenarios within MRS.

3. Experiments

The main goal of the experiments is to describe and validate the strategy of the architecture's plan recovery process [da Silva 2024]. Experiments using MRS and a healthcare case already were made [da Silva and Ralha 2023]. However, the architecture focuses on working with multiple domains. Thus, the necessity of testing with other technologies and domains is vital. This work uses the Space Resource Gathering (SRG) on planet exploration illustration example.

3.1. Space Resource Gathering

Motion and path planning is already a common area of study in space robotics, as seen in works of [Basmadji et al. 2020], [Sun et al. 2023]. However, planet exploration is also a possible focus, and with the advancement of technology, we demand more research. On that note, to expand the research of planet exploration, we describe this illustrative example of *SRG* where we idealize that the cost of sending robots to the planets is lower and the possibility of sending a heterogeneous group of robots to perform various tasks is a possibility.

The *SRG* includes three robot types. One robot can map the environment looking for resources (*Scout*), another can collect the found resources (*Gatherer*), and the last has the skill of removing obstacles (*Remover*) that can appear during the plan execution. The mission has two types of main plans, one of mapping the environment and the other of collecting the resources. In the last part, should any obstacles arise, the plan will fail, and the *Coordinator* needs to replan accordingly.

Figure 6 shows a simple example of a map to illustrate how the simulation would work. The map has four parts: Base, Location 1, Location 2, and Location 3. In the figure, while the *Gatherer* is collecting resources in Location 1, the *Remover* is removing obstacles in Location 2, and the *Scout* searches for resources in Location 3.

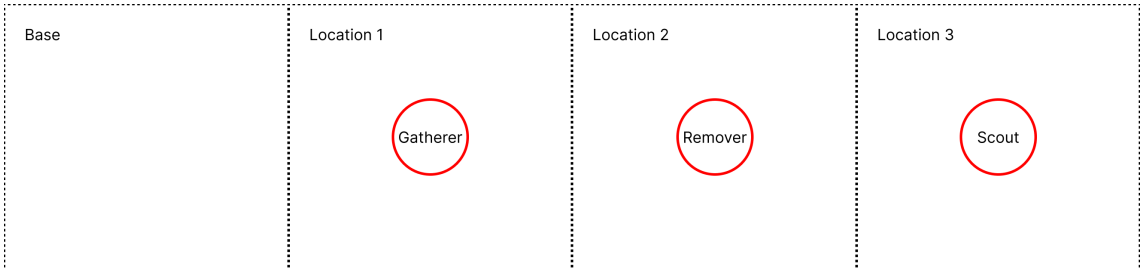


Figure 6. Example of a simulation map.

Listing 1 shows an example of a plan generated by the HyperTensioN where no replan is needed. The *Scout* searches for a resource, and then the *Gatherer* collects it. In Listing 2, a new part of the plan is needed after the *Gatherer* tries to collect the resource, but an external event happens, and an obstacle is in the way, so the *Remover* need to remove it before the *Gatherer* can finish its local plan.

Listing 1. Example of a plan without replan.

```
move_scout(scout base location)
map(scout resource enemy location)
move_scout(scout location base)
move(gatherer base location)
pickup(gatherer resource)
move(gatherer location base)
drop(gatherer resource)
```

Listing 2. Example of a plan with replan.

```
move_scout(scout base location)
map(scout resource enemy location)
move_scout(scout location base)
move_remover(remover base location)
```

remove_obstacle(remover obstacle location)
move_remover(remover location base)
move(gatherer base location)
pickup(gatherer resource)
move(gatherer location base)
drop(gatherer resource)

3.2. Experimental Setup

The experiments use Ruby (v2.6.10) [Matsumoto 2022] as the principal language. A process creates a thread for each agent (*Coordinator* with planner and robots). The communication between the agents uses a web socket with the TCP protocol. The messages exchange uses a JSON format with two attributes, "action" with a string representing the message purpose and a "value" with the necessary data for the agent act. Figure 7 shows the execution process of the experiment.

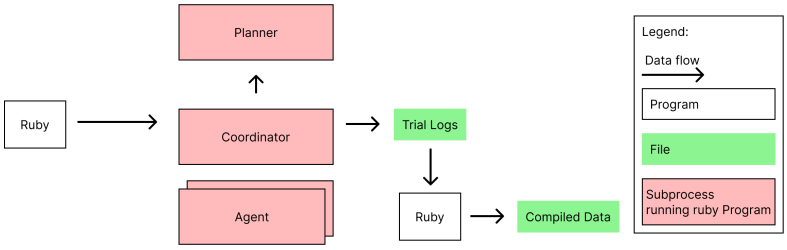


Figure 7. Experimental process.

The experiment execution took into account the following aspects:

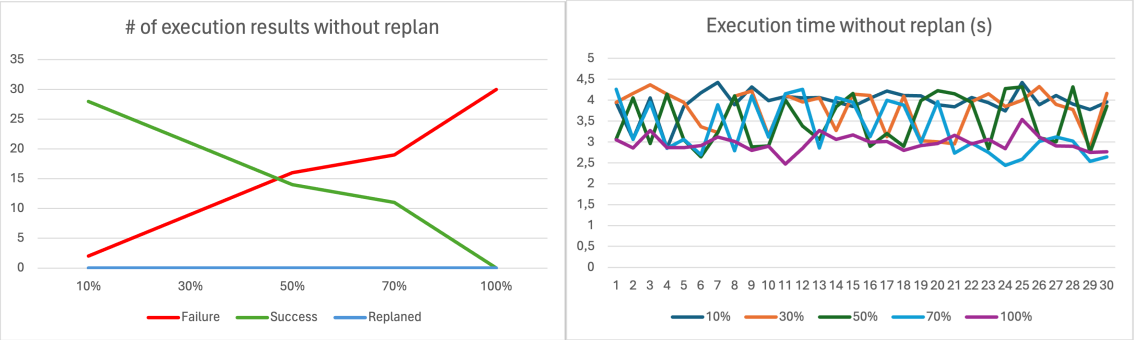
- objective - to evaluate how effectively the architecture mitigates issues when operating in a simulated dynamic environment.
- problem types - Objects preventing resource collection.
- case study - the SRG using five experimental scenarios related to the evaluated problem.
- validation strategy - we conducted five scenarios, each repeated 30 times for statistical significance, to assess the *Coordinators*' ability to complete the plan. Plan results were analyzed at the end of each execution, focusing on the percentage of uncompleted missions related to the evaluated problem. The scenarios featured a door-closed event with probabilities of 10%, 30%, 50%, 70%, and 100%.
- results evaluation metric - plan completion and time required to complete the plan. We compared the experimental results with a baseline case where the *Coordinator* cannot replan.

3.3. Results

In this section, we analyze the experimental results to assess the effectiveness of the proposed architecture in dynamic environments, specifically focusing on the plan recovery process illustrated with space resource-gathering robots.

3.3.1. Baseline Analysis

The baseline scenario serves as a control to compare the effectiveness of the plan recovery mechanism. In the baseline, the architecture cannot replan in response to unexpected events. The results are depicted in Figures 8a and 8b.



(a) # of execution results without replan. **(b) Execution time without replan (s).**

Figure 8. Charts of mission's results without replan.

Figure 8a shows the number of mission completions without replan capability across various probabilities of obstacles (10%, 30%, 50%, 70%, and 100%). As expected, the number of failures increases proportionally with the obstacle occurrence probability. This result demonstrates the limitation of static planning in dynamic environments, where an increased likelihood of external disruptions significantly impacts mission success rates.

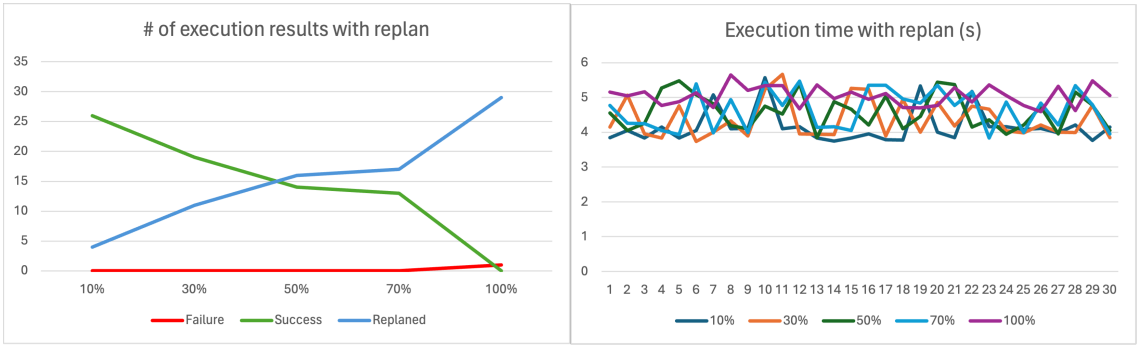
Figure 8b presents the execution time for missions without replanning. Despite variations in the probability of obstacles, the execution times remain relatively consistent. This result relates to the simplified plan nature used in the baseline scenario, where missions either succeed without disruption or fail early due to obstacles.

3.3.2. Replan Capability Analysis

The replan scenarios incorporate the architecture's ability to recover plans dynamically in response to obstacles. Figure 9a and Figure 9b present the results.

Figure 9a depicts the number of mission completions with replan capability across varying probabilities of obstacles considering success when no problem occurs. Unlike the baseline, the architecture's ability to replan significantly improves mission success rates, even as the probability of obstacles increases. This result demonstrates the effectiveness of the plan recovery mechanism in mitigating the impact of dynamic environmental changes.

Figure 9b shows the execution time for missions with replanning. Similar to the baseline, the execution times are consistent across different probabilities of obstacles. However, note that the architecture's replan capability adds a slight overhead to the execution time due to the replanning process. Despite this, the improvement in mission success rates outweighs the marginal increase in execution time.



(a) # of execution results with replan. **(b) Execution time with replan (s).**

Figure 9. Charts of mission's results with replan.

3.3.3. Comparative Analysis

The advantages of the replan capability are evident in Figures 8 and 9. The architecture's ability to dynamically recover plans allows it to maintain high mission success rates, even with frequent obstacles. The slight increase in execution time is a reasonable trade-off for the enhanced robustness and reliability of the replan mechanism.

Overall, the results validate the effectiveness of the proposed architecture in handling dynamic environments through its replan capability. The ability to dynamically adjust plans in response to unexpected events significantly enhances mission success rates, demonstrating the potential of this approach for real-world applications in multi-robot systems operating in unpredictable conditions.

4. Conclusion

The experiments validate the proposed MAS architecture in [da Silva 2024] integrating AP for multi-agent coordination in dynamic environments. The *SRG* example illustrates the architecture's capabilities in a simulated space robotics scenario, managing task decomposition, coalition formation, and task allocation adapted to dynamic changes, ensuring mission continuity. The central *Coordinator* oversees planning and execution, presenting a reliable strategy for managing heterogeneous robotic teams in the studied scenario. Comparing the scenarios, it is evident that this functionality enhances the system's ability to complete missions successfully, even when the probability of encountering obstacles increases.

Future work focuses on enhancing the architecture's capabilities with false action results using sophisticated heuristic algorithms. Further, performing experiments in other domains to validate the architecture's versatility, experiments with increasing numbers of robots, integrating with more advanced planning algorithms to enhance communication protocols, and finding a suitable benchmark for multi-agent planning and comparing with some existing work will be salient to future research.

References

Aziz, H., Chan, H., Cseh, A., Li, B., Ramezani, F., and Wang, C. (2021). Multi-robot task allocation-complexity and approximation. In *Proc. of 20th Int. Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, page 133–141.

Basmadji, F., Seweryn, K., and Sasiadek, J. (2020). Space robot motion planning in the presence of nonconserved linear & angular momenta. *Multibody System Dynamics*, 50.

Bischoff, E., Teufel, J., Inga, J., and Hohmann, S. (2021). Towards interactive coordination of heterogeneous robotic teams – introduction of a reoptimization framework. In *Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC)*, pages 1380–1386.

Carreno, Y., Ng, J. H. A., Petillot, Y., and Petrick, R. (2022). Planning, execution, and adaptation for multi-robot systems using probabilistic and temporal planning. In *Proc. of 21st Int. Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, page 217–225.

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carreraa, A., Palomeras, N., Hurtós, N., and Carrerasa, M. (2015). ROSPlan: Planning in the robot operating system. In *Proc. of 35th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, page 333–341.

da Silva, C. J. T. (2024). A multi-robot system architecture with multi-agent planning. Computer Science Department, University of Brasilia, Campus Darcy Ribeiro - Asa Norte, Brasília - DF, 70910-900, Brazil.

da Silva, C. J. T. and Ralha, C. G. (2023). Multi-robot system architecture focusing on plan recovery for dynamic environments. In *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1668–1673.

et al., M. (2021). RoboMAX: Robotic mission adaptation exemplars. In *Proc. of Int. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 245–251.

Georgievski, I. and Aiello, M. (2015). Htn planning: Overview, comparison, and beyond. *Artificial Intelligence*, 222:124–156.

González, J. C., García-Olaya, A., and Fernández, F. (2020). Multi-layered multi-robot control architecture for the robocup logistics league. In *Proc. of IEEE Int. Conf. on Autonomous Robot Systems and Competitions*, pages 120–125.

Kiener, J. and von Stryk, O. (2010). Towards cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. *Robotics and Autonomous Systems*, 58(7):921–929. Advances in Autonomous Robots for Service and Entertainment.

Klavins, E. (2004). *Communication Complexity of Multi-robot Systems*, pages 275–291. Springer, Berlin, Heidelberg.

Komenda, A., Stolba, M., and Kovacs, D. L. (2016). The international competition of distributed and multiagent planners (CoDMAP). *AI Magazine*, 37(3):109–115.

Lesire, C., Bailon-Ruiz, R., Barbier, M., and Grand, C. (2022). A hierarchical deliberative architecture framework based on goal decomposition. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 9865–9870.

Magnaguagno, M. C., Meneguzzi, F., and De Silva, L. (2022). HyperTensioN: A three-stage compiler for planning. In *Proc. of 30th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 1–4.

Martín, F., Clavero, J. G., Matellán, V., and Rodríguez, F. J. (2021). PlanSys2: A planning system framework for ROS2. In *Proc. of IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, page 9742–9749.

Matsumoto, Y. 2022). Ruby. `https://www.ruby-lang.org/en/`. Accessed: 2024-06-03.

Moreira, L. H. and Ralha, C. G. (2021a). Evaluation of decision-making strategies for robots in intralogistics problems using multi-agent planning. In *Proc. of IEEE Congress on Evolutionary Computation*, pages 1272–1279.

Moreira, L. H. and Ralha, C. G. (2021b). Plan recovery process in multi-agent dynamic environments. In Gusikhin, O., Nijmeijer, H., and Madani, K., editors, *Proc. of 18th Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO)*, pages 187–194.

Moreira, L. H. and Ralha, C. G. (2022a). An efficient lightweight coordination model to multi-agent planning. *Knowledge and Information Systems*, 64:415–439.

Moreira, L. H. and Ralha, C. G. (2022b). Method for evaluating plan recovery strategies in dynamic multi-agent environments. *Journal of Experimental & Theoretical Artificial Intelligence*, pages 1–25.

Object Management Group (OMG) (2015). Meta-Object Facility (MOF) Specification, version 2.5. OMG Document Number formal/2015-03-01.

Rizk, Y., Awad, M., and Tunstel, E. W. (2019). Cooperative heterogeneous multi-robot systems: A survey. *ACM Comput. Surv.*, 52(2).

Rodrigues, G., Caldas, R., Araujo, G., de Moraes, V., Rodrigues, G., and Pelliccione, P. (2022). An architecture for mission coordination of heterogeneous robots. *Journal of Systems and Software*, 191(111363).

Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education.

Salzman, O. and Stern, R. (2020). Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proc. of 19th Int. Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, page 1711–1715.

Schmitt, P. S., Wirnshofer, F., Wurm, K. M., Wichert, G. v., and Burgard, W. (2019). Modeling and planning manipulation in dynamic environments. In *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, pages 176–182.

Silva, L. d., Meneguzzi, F., and Logan, B. (2020). BDI Agent Architectures: A Survey. In *Proc. of 29th Int. Joint Conf. on Artificial Intelligence, (IJCAI)*, pages 4914–4921.

Sun, Y., Wu, J., and Liu, T. (2023). Joint task allocation and path planning for space robot. *IEEE Access*, 11:42314–42323.

Verma, J. K. and Ranga, V. (2021). Multi-robot coordination analysis, taxonomy, challenges and future scope. *Journal of Intelligent & Robotic Systems*, 102(1).

Weiss, G. (2016). *Multiagent Systems*. The MIT Press, 2nd edition.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Refatoração da Extensão NetLogo de Aprendizagem por Reforço para Integração com a Biblioteca BURLAP

Eloísa Bazzanella[1], **Matheus M. Barros**[1], **Fernando Santos**[1]

[1]Departamento de Engenharia de Software
Universidade do Estado de Santa Catarina (UDESC)
Ibirama – SC – Brasil

{elobazzanella,matheusmbarros01}@gmail.com, fernando.santos@udesc.br

Abstract. *Agent-based modeling and simulation is a simulation paradigm that allows focusing on individuals, their interactions, and the resulting complex behavior. Agent-based simulations are typically developed in simulation environments that provide agent-related features. One such environment is NetLogo, for which there is an extension that enables the use of reinforcement learning by agents, specifically the Q-Learning algorithm. This paper describes the refactoring of this extension to integrate with BURLAP, a mature reinforcement learning Java library. Evaluations have shown that the refactoring has maintained the consistency and functionality of the NetLogo extension. With the refactoring, the NetLogo extension now enables the use of reinforcement learning algorithms SARSA(λ) and Actor-Critic, in addition to Q-Learning.*

Resumo. *A modelagem e simulação baseada em agentes é um paradigma de simulação que permite focar nos indivíduos, em suas interações e no comportamento complexo resultante. Simulações baseadas em agentes são geralmente desenvolvidas em ambientes de simulação que fornecem recursos relacionados a agentes. Um desses ambientes é o NetLogo, que possui uma extensão para que os agentes possam utilizar aprendizagem por reforço, especificamente o algoritmo Q-Learning. Este artigo descreve a refatoração dessa extensão para integração com a BURLAP, uma biblioteca Java consolidada de aprendizagem por reforço. Avaliações evidenciaram que a refatoração manteve a extensão NetLogo consistente e funcional. A partir da refatoração, a extensão NetLogo passa a permitir o uso dos algoritmos de aprendizagem por reforço SARSA(λ) e Actor-Critic, além do Q-Learning.*

1. Introdução

As simulações baseadas em agentes (SBAs) são uma ferramenta poderosa para modelar e entender fenômenos complexos em diversas áreas do conhecimento. De acordo com [Epstein e Axtell 1996], essas simulações consistem na criação de modelos realistas e dinâmicos que capturam a interação e o comportamento dos agentes em um sistema. Portanto, as SBAs constituem uma ferramenta precisa para compor e testar ocorrências de diversos fenômenos e analisar suas implicações.

Agentes são sistemas inteligentes, capazes de perceber o ambiente, autônomos em sua tomada de decisão e aptos a interação com outros agentes. Para a tomada de decisão,

um agente pode se basear em deduções lógicas, assim como o raciocínio humano, ou por meio da dedução combinada a algum mecanismo [Wooldridge 2009]. Sendo assim, os agentes e suas interações são os principais alvos de estudo quando se fala em SBAs.

SBAs podem ser implementadas em linguagens de propósito geral, como Python. No entanto, já existem ambientes de simulação que fornecem recursos específicos e simplificam o desenvolvimento. Um destes é o NetLogo, um ambiente de programação projetado para modelagem de sistemas complexos. Segundo [Wilensky 1999], o NetLogo permite a criação de SBAs com facilidade e rapidez, bem como a análise e visualização dos resultados das simulações. Recentemente foi disponibilizada uma extensão (biblioteca) para o NetLogo que permite utilizar aprendizagem por reforço em SBAs [Kons 2019]. Uma limitação dessa extensão é que suporta apenas o algoritmo *Q-Learning*.

Este artigo descreve a refatoração da extensão NetLogo existente. O objetivo é integrar a extensão com a BURLAP [MacGlashan et al. 2018], uma biblioteca de aprendizagem por reforço Java consolidada, e disponibilizar outros algoritmos de aprendizagem por reforço. A partir da refatoração, a extensão agora permite utilizar, além do *Q-Learning*, também os algoritmos *SARSA(λ)* e *Actor-Critic*.

Uma avaliação foi realizada para evidenciar que a refatoração manteve a estabilidade da extensão NetLogo, sem introduzir inconsistências nos comandos existentes e nos resultados do algoritmo *Q-Learning* já disponível. Por fim, o *SARSA(λ)* e *Actor-Critic* foram utilizados em duas SBA para evidenciar o funcionamento da extensão refatorada.

2. Fundamentação Teórica

2.1. Simulações Baseadas em Agentes

Simulações baseadas em agentes (SBAs) têm sido utilizadas para simular sistemas complexos e dinâmicos, nos quais um conjunto de agentes autônomos interagem entre si e com o ambiente [Macal e North 2010]. As SBAs têm sido amplamente utilizado em diversas áreas para compreender e prever o comportamento de sistemas complexos, tais como em economia, ciências sociais, biologia, e engenharia [Bonabeau 2002]. SBAs permitem criar modelos mais realistas e precisos, uma vez que levam em consideração a heterogeneidade, a adaptação e a interação dos agentes com o ambiente [Macy e Willer 2002].

Para a criação de uma SBA é necessário definir as características dos agentes, como seu comportamento, objetivos e regras de interação. Uma vez definidos esses aspectos, é possível utilizar algoritmos de aprendizagem de máquina, para permitir que os agentes aprendam e se adaptem ao ambiente em que estão inseridos.

Além disso, as SBAs permitem a realização de experimentos virtuais que seriam inviáveis ou impraticáveis na vida real [Epstein e Axtell 1996]. Dessa forma, os modelos baseados em agentes podem ser utilizados para testar diferentes cenários, avaliar a eficácia de estratégias de tomada de decisão e propor soluções para problemas complexos.

2.2. Aprendizagem por Reforço

A aprendizagem por reforço (*reinforcement learning — RL*) é uma área de estudo da inteligência artificial que busca desenvolver algoritmos capazes de aprender a tomar decisões em situações complexas a partir do *feedback* de um ambiente externo [Sutton e Barto 2018]. A RL é uma abordagem indicada para situações em que

se deseja encontrar uma política de comportamento ótima. Uma política especifica como o agente deve agir em qualquer situação (estado) que possa se deparar. A política ótima, por sua vez, é a política que resulta no melhor desempenho do agente. A política ótima pode ser aprendida a partir das interações com o ambiente, mediante o conhecimento do estado do agente, das ações efetuadas e das mudanças nos estados [Sutton e Barto 2018].

Entre as características da RL, é possível mencionar a aprendizagem baseada na ação do agente no ambiente, que resulta num valor de recompensa, usado para tomar decisões posteriores. Outra característica importante é o *delayed reward*, que diz respeito à qualidade das ações, isto é, um valor de recompensa alto em determinado momento não significa necessariamente que a ação tomada é a recomendada, pois o intuito do agente é alcançar objetivos globais e não locais. Ademais, a RL também caracteriza-se por ser orientada à objetivo, ou seja, o agente não precisa conhecer detalhes da modelagem do ambiente; ele simplesmente age no ambiente tentando alcançar um objetivo [Sutton e Barto 2018]. Dentre os algoritmos de RL disponíveis estão o *Q-Learning*, o *SARSA(λ)*, e o *Actor-Critic*. Estes algoritmos buscam maximizar a recompensa esperada do agente no ambiente. Sobre estes algoritmos, a seguir serão apresentados aspectos relevantes ao presente trabalho. Recomenda-se que o leitor interessado em se aprofundar nos detalhes destes algoritmos consulte [Russel e Norvig 2004] e [Sutton e Barto 2018].

No *Q-Learning* [Watkins e Dayan 1992] e no *SARSA(λ)* [Rummery e Niranjan 1994], o agente aprende uma função de ação-valor. Tal função, denominada $Q(s, a)$, estima a qualidade de cada ação a em cada estado s do ambiente. Estes algoritmos se baseiam na ideia de que a melhor ação a ser tomada em um determinado estado é aquela que maximiza o valor de Q. Os valores de $Q(s, a)$ são determinados iterativamente, a medida que o agente atua no ambiente. A cada ação executada, o ambiente fornece ao agente um sinal de recompensa r. Desta forma, após experienciar o ambiente, tanto o *Q-Learning* quanto o *SARSA(λ)* estimam os valores de $Q(s, a)$ a partir das recompensas obtidas ao longo do tempo. A diferença entre o *Q-Learning* e o *SARSA(λ)* é que o último é um algoritmo *on-policy*, enquanto o primeiro é *off-policy*. Em um algoritmo *on-policy* o agente segue uma política de comportamento definida, e a aprendizagem atua para refinar essa política. Já em um algoritmo *off-policy* o agente não segue uma política de comportamento definida, sendo livre para experienciar ações arbitrariamente durante a aprendizagem [Sutton e Barto 2018].

Por fim, o *Actor-Critic* [Konda e Tsitsiklis 2000] combina as estratégias de funcionamento do *Q-Learning* e do *SARSA(λ)*, resultando em uma estratégia de RL mais eficiente. O *Actor-Critic* possui duas etapas. A etapa de avaliação de política diz respeito ao uso eficiente da experiência já aprendida pelo agente. Enquanto a etapa de melhoria da política serve para melhorar a política em todas as etapas até a convergência [Peters e Schaal 2008]. Em outras palavras, no *Actor-Critic* a etapa de avaliação pode ser interpretada como um elemento ator, que realiza a aproximação da função de utilidade dos estados (estratégia do *SARSA(λ)*). Já a etapa de melhoria pode ser interpretada como um elemento crítico, que realiza a aproximação da função de utilidade das ações (estratégia do *Q-Learning*) [Wang et al. 2007].

2.3. NetLogo e a Extensão Q-Learning

O NetLogo é um ambiente programável para modelagem e simulação de fenômenos naturais e sociais [Wilensky 1999]. Ele oferece uma linguagem de programação própria, com

comandos de alto nível para implementar funcionalidades de SBAs. Com isso, é possível dar instruções para centenas de agentes independentes que agem simultaneamente, com o intuito de explorar as conexões entre os comportamentos de nível micro dos indivíduos e os padrões de nível macro que resultam de suas interações [Sklar 2007]. Estatísticas indicam que em 2020, 33.6% das simulações disponíveis no repositório CoMSES eram feitas em NetLogo [CoMSES 2020].

A partir da versão 2.0.1, o NetLogo passou a fornecer uma API para a criação de extensões. Uma extensão é um módulo NetLogo que estende sua linguagem de programação para prover comandos e recursos adicionais. A API disponibiliza diversos elementos para o desenvolvedor utilizar durante a implementação, facilitando a integração com a linguagem NetLogo [Tisue e Wilensky 2004].

Recentemente, [Kons 2019] desenvolveu e disponibilizou uma extensão para uso do algoritmo *Q-Learning* no NetLogo. O objetivo desta extensão é auxiliar no processo de criação de agentes inteligentes, possibilitando que o desenvolvedor de simulações não precise implementar o *Q-Learning* manualmente. Para isso, a extensão disponibiliza novos comandos que, quando utilizados, definem os elementos requeridos pelo *Q-Learning*. A Figura 1 apresenta os comandos disponíveis na extensão, que podem ser classificados em comandos de *configuração* e de *execução* da aprendizagem.

```
; comandos de configuracao da aprendizagem
qlearningextension:state-def ; especificar os estados
qlearningextension:actions ; especificar as acoes
qlearningextension:reward ; especificar a recompensa
qlearningextension:learning-rate ; especificar a taxa de aprendizagem
qlearningextension:discount-factor ; especificar o fator de desconto
qlearningextension:action-selection ; especificar estrategia de selecao da acao

; comandos de execucao da aprendizagem
qlearningextension:learning ; executar acao e aprender
qlearningextension:act ; apenas executar uma acao
qlearningextension:learn ; apenas aprender a partir da acao executada e recompensa obtida
```

Figura 1. Comandos da Extensão NetLogo. Fonte: [Kons 2019]

Os comandos de *configuração* permitem que o desenvolvedor especifique o problema de aprendizagem, formado pelos estados, ações, recompensas, e também os parâmetros do algoritmo. Para usar o *Q-Learning*, basta ao desenvolvedor utilizar estes comandos no *setup* da simulação. Já os comandos de *execução* permitem ativar a aprendizagem na implementação do *step* da simulação, fazendo com que o agente execute o algoritmo *Q-Learning* considerando o estado, ação, recompensa, e os parâmetros do algoritmo. A extensão e a documentação de seus comandos estão disponíveis online.[1]

[Bazzanella e Santos 2021] avaliaram o benefício que a extensão *Q-Learning* fornece para o desenvolvimento de SBAs. Por meio de uma avaliação quantitativa, evidenciaram que o tamanho do código fonte das simulações é significativamente reduzido quando se utiliza a extensão para implementar o *Q-Learning*. Apesar dessa vantagem, atualmente a extensão se limita a disponibilizar apenas o algoritmo *Q-Learning* para RL.

[1] https://github.com/agentbasedsimulations/qlearning-netlogo-extension

2.4. Biblioteca BURLAP

A *Brown-UMBC Reinforcement Learning and Planning* (BURLAP) é uma biblioteca Java para desenvolvimento de agentes inteligentes que usam aprendizagem por reforço e planejamento [MacGlashan et al. 2018]. A BURLAP estabelece uma estrutura geral para que seja possível definir um domínio de problema, e fornece uma ampla variedade de algoritmos RL integrados, domínios pré-fabricados e ferramentas de visualização. Dentre os algoritmos disponibilizados estão o *Q-Learning*, o *SARSA(λ)*, e o *Actor-Critic*. Além disso, ela fornece uma interface de programação simples para desenvolver novos algoritmos de aprendizagem e planejamento. Com o uso da BURLAP, torna-se mais fácil implementar e testar algoritmos de RL em diversos tipos de problemas [MacGlashan et al. 2018].

A BURLAP tem sido amplamente utilizada em pesquisas acadêmicas em aprendizagem por reforço e planejamento, como em estudos sobre controle de robôs e jogos eletrônicos. Além disso, a biblioteca também tem sido usada para desenvolver aplicações práticas em áreas como sistemas autônomos e robótica. A BURLAP está disponível online[2]. Para utilizá-la, basta realizar as interfaces Java disponibilizadas para implementar o problema de aprendizagem em questão. Há interfaces para implementar os estados, as ações, o ambiente, e as transições que ocorrem no ambiente quando as ações são executadas. Uma vez instanciadas e integradas, pode-se aplicar os algoritmos disponibilizados pela BURLAP para realizar a aprendizagem do agente no problema em questão.

3. Refatoração da Extensão

O objetivo da refatoração da extensão NetLogo de aprendizagem por reforço é utilizar a estrutura e algoritmos de RL fornecidos pela biblioteca BURLAP. Esta seção descreve as alterações realizadas na extensão para utilizar a BURLAP e disponibilizar, além do *Q-Learning*, outros dois algoritmos de RL, o *SARSA(λ)* e o *Actor-Critic*.

Inicialmente, foi necessário realizar a adaptação da extensão proposta por [Kons 2019], que estava originalmente implementada em Scala, para a linguagem Java, visando a sua integração com a biblioteca BURLAP. Essa adaptação foi essencial devido à natureza da biblioteca BURLAP, escrita em Java, com o objetivo de assegurar a compatibilidade e viabilizar uma integração eficiente entre as duas partes.

Para incorporar a BURLAP na extensão, foi especificada uma arquitetura de integração entre a BURLAP e a simulação no NetLogo. Essa integração se utiliza dos recursos disponibilizados pela API do NetLogo[3] para acessar elementos da simulação. A partir dos comandos NetLogo disponibilizados pela extensão, pode-se instanciar os objetos da BURLAP necessários para implementar a aprendizagem nos agentes. Estes objetos, por estarem vinculados aos elementos da simulação, permitem que a BURLAP acesse as informações necessárias à aprendizagem, tais como estados, ações e recompensas.

A arquitetura projetada para fazer a integração entre a biblioteca BURLAP e a simulação NetLogo é apresentada na Figura 2. Os elementos em azul são disponibilizados pela BURLAP, no caso, as interfaces a serem implementadas. Já os elementos em vermelho são as classes disponibilizadas pela API do NetLogo. As classes da extensão, na cor branca, são responsáveis por criar comandos para serem utilizados no NetLogo,

[2] http://burlap.cs.brown.edu

[3] https://github.com/NetLogo/NetLogo/wiki/Extensions-API

Figura 2. Arquitetura da Extensão NetLogo refatorada com a BURLAP

afim de receber e armazenar as informações da simulação e implementar o problema de aprendizagem utilizando a BURLAP. A seguir são explicadas as classes desta arquitetura.

A `ExtensionCommands` é responsável por determinar quais comandos serão disponibilizados no NetLogo. Ela herda as funcionalidades da `DefaultClassManager`, exigência da API de integração para permitir ao ambiente NetLogo reconhecer os comandos disponibilizados e seus argumentos. Um exemplo de comando mostrado no diagrama da Figura 2 é a `LearningRateCommand`, que especifica a sintaxe e a execução do comando que define a taxa de aprendizagem. Todas as outras classes de comando seguem a mesma estrutura, realizando a interface `Command`. Se um comando retorna algum valor, é implementado como um `Reporter`.

A `Session` segue o padrão *singleton* para armazenar a instância da extensão e os agentes envolvidos, que são objetos do tipo `AgentLearning`. A `AgentLearning` é responsável por todas as informações necessárias para a aprendizagem, tais como fator de desconto, taxa de aprendizagem, função de recompensa e informações do objeto `Agent` fornecido pela API do NetLogo.

A `StateDefinition` define os estados, armazenando as informações que serão usadas na sua especificação. Já a `ActionSelection` seleciona a próxima ação, armazenando as informações sobre a estratégia de seleção de ação e seus parâmetros. Ambas são agregadas à `AgentLearning`. A `Reward` implementa a interface `RewardFunction` e é responsável por gerenciar as recompensas do agente de acordo com seu estado atual. A `IsEndEpisode` implementa a interface `TerminalFunction` fornecida pela BURLAP e determina se um estado é terminal.

A `AgentState` implementa a interface `MutableState` da BURLAP e converte as informações sobre os estados do objeto `StateDefinition` para uso na BURLAP. Já a `AgentStateModel` implementa a interface `FullStateModel` da BURLAP e faz a transição de estados após a execução da ação selecionada.

Por fim, as classes `QLearningAlgorithm`, `SarsaAlgorithm` e `ActorCriticAlgorithm` têm a responsabilidade de iniciar a criação da aprendizagem e executá-la usando os recursos fornecidos pela BURLAP. Cada uma delas implementa a interface `DomainGenerator` e tem uma única instância, com dois métodos principais: para configurar a aprendizagem; e para executar o processo de aprendizagem.

Tendo em vista que a extensão foi ampliada para incorporar outros algoritmos de RL além do *Q-Learning*, foi necessário renomeá-la, passando a se chamar `learningextension`. Além disso, para a incorporação desses novos algoritmos, foi necessário criar novos comandos NetLogo para serem executados no *setup* da simulação. A Figura 3 apresenta os novos comandos incorporados.

```
learningextension:define-algorithm ; "qlearning", "sarsa-lambda" or "actor-critic"
learningextension:lambda
learningextension:setup
```

Figura 3. Novos comandos disponibilizados na extensão.

O comando `learningextension:define-algorithm` permite especificar qual algoritmo será utilizado na aprendizagem (*Q-Learning*, *SARSA(λ)*, ou *Actor-Critic*).

O comando `learningextension:lambda` define a taxa de exigibilidade, parâmetro requerido pelos algoritmos *SARSA(λ)* e *Actor-Critic*. Por fim, o comando `learningextension:setup` deve ser utilizado para instanciar os objetos da BURLAP internamente na extensão e deixá-la pronta para executar o processo de aprendizagem. A documentação de todos os comandos da extensão refatorada está disponível online.[4].

O diagrama de atividades da Figura 4 descreve o fluxo de integração da extensão refatorada (raia central) com a simulação NetLogo e a biblioteca BURLAP. Essa integração ocorre através da arquitetura apresentada anteriormente na Figura 2. Conforme mencionado na seção 2.3, os comandos da extensão podem ser classificados em comandos de *configuração* e de *execução* da aprendizagem.

O ponto de início `A` representa a chamada de algum comando de *configuração* (por exemplo `learningextension:state-def`). A classe `StateDefinitionCommand` é responsável por validar a sintaxe e processar o comando utilizando-se da API do NetLogo. Nessa classe são definidas as variáveis que serão consideradas para especificar o estado do problema de aprendizagem. Depois de validada a sintaxe, é iniciado o processamento do comando. Neste processamento é instanciada a classe `StateDefinition` com as informações recebidas e este objeto é vinculado ao objeto `AgentLearning`, que também foi instanciado. A classe `AgentLearning` centraliza todas as informações relevantes para o aprendizado do agente. Todos os demais comandos de *configuração* seguem a mesma lógica de receber a informação por meio de uma classe que implementa a API do NetLogo e salva essas informações nos objetos da extensão refatorada.

Para finalizar a configuração da aprendizagem, o desenvolvedor utiliza o comando `learningextension:setup`, representado pelo ponto de início `B`. Ao ser executado, chama a classe `LearningCommand` que verifica a sintaxe e usa todas as informações que estão salvas no objeto `AgentLearning` para chamar a classe de *learning*. Essa classe varia conforme o algoritmo que foi selecionado, podendo ser o `QLearningAlgorithm`, `SarsaAlgorithm` ou `ActorCriticAlgorithm`. Nessa classe que é feita a inicialização do aprendizado na BURLAP, ou seja, a modelagem do problema.

Por fim, o ponto de início `C` representa a chamada do comando `learningextension:learning`, que é um comando de *execução* da aprendizagem. Este comando deve ser chamado dentro do procedimento que implementa o *step* da simulação. Ao utilizá-lo, é feita a chamada para a classe `LearningCommand`, que valida a sintaxe e reconhece o algoritmo que está sendo utilizado. Ao saber o algoritmo que está sendo utilizado, chama a classe de *learning* e executa os passos do aprendizado.

4. Avaliação da Extensão Refatorada

Com o intuito de avaliar o funcionamento da extensão, foram implementadas duas SBAs para verificar a aprendizagem de cada algoritmo. A hipótese é que, se o agente aprender uma política consistente com o que cada algoritmo se propõe, então a refatoração da extensão com integração do NetLogo e BURLAP está consistente e funcional.

A primeira SBA é o cenário *Cliff Walking* de [Sutton e Barto 2018], apresentado na Figura 5a. Neste cenário o ambiente é um mundo bidimensional dividido em células. O agente sempre ocupa uma única célula e pode se locomover por entre as células com

[4]https://github.com/agentbasedsimulations/qlearning-netlogo-extension/tree/burlap-migration

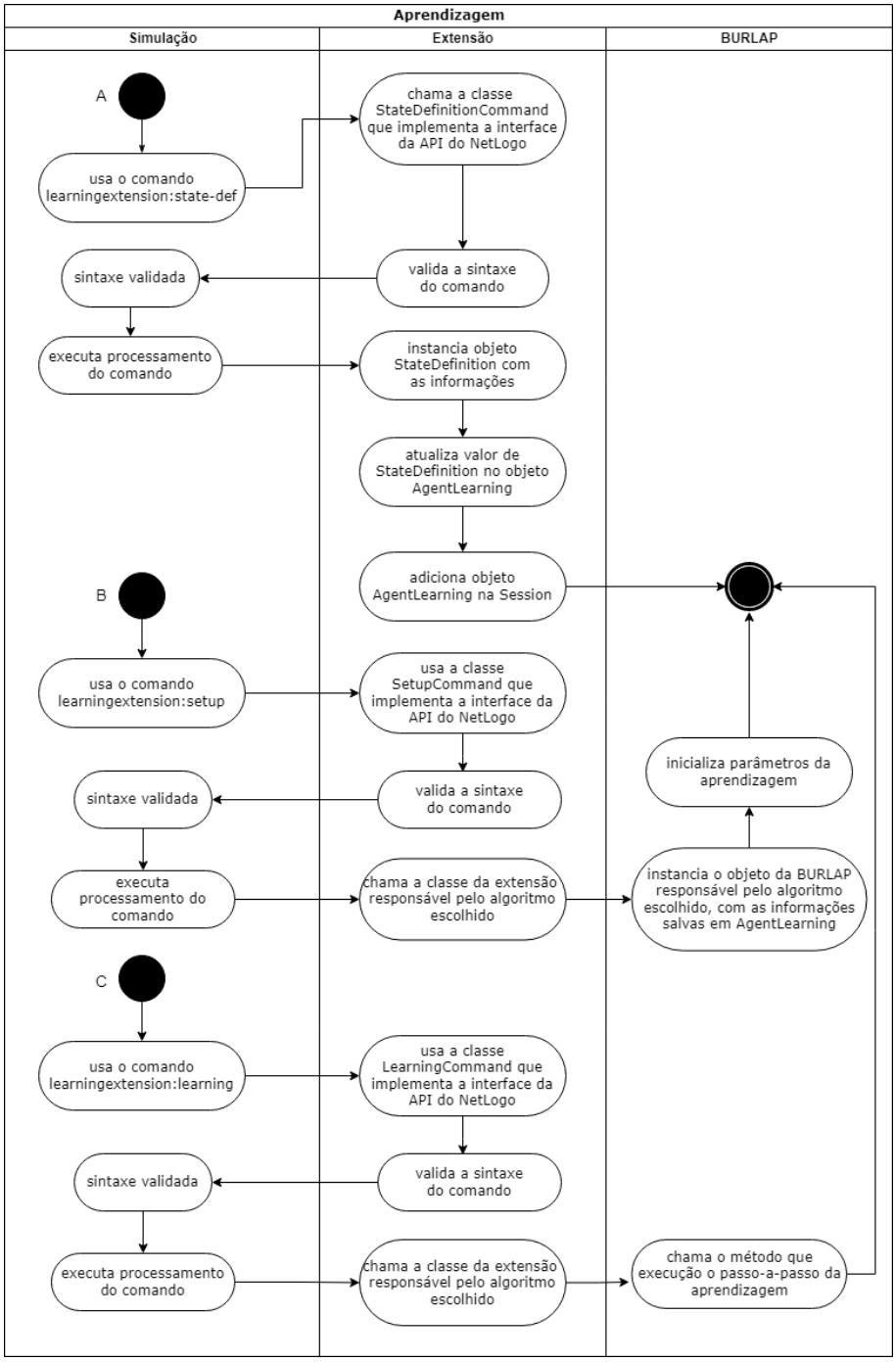
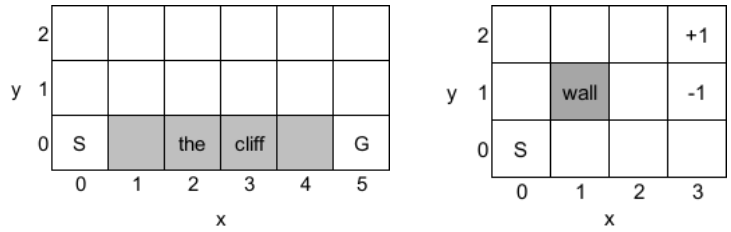


Figura 4. Diagrama de Integração NetLogo e BURLAP

as ações de ir para: cima, baixo, esquerda e direita. O estado do agente, portanto, é representado pela sua posição na grade de células. Neste cenário o agente deve aprender a se locomover, para sair do ponto S e chegar ao ponto G, traçando o menor percurso possível ou o mais seguro, sem passar pelo penhasco, representado pelas células em cinza. Quando o agente se desloca para qualquer célula que não seja o penhasco, recebe uma recompensa no valor -1. Se cair no penhasco recebe recompensa de -100.

A segunda SBA é o cenário *Mundo 4x3* de [Russel e Norvig 2004]. Neste cenário

o ambiente também é um mundo bidimensional dividido em células, apresentado na Figura 5b. As ações do agente, bem como a representação dos estados, são as mesmas do cenário *Cliff Walking*. Há dois estados terminais, sendo que um oferece recompensa +1 e outro fornece recompensa -1. Qualquer outra célula fornece ao agente uma recompensa no valor de -0.04 (exceto a célula inacessível ao agente indicada em cinza). O agente deve aprender a se locomover do ponto S até o estado terminal com máxima recompensa.



(a) Cliff Walking **(b)** Mundo 4x3

Figura 5. Simulações utilizadas para avaliação da extensão refatorada

Para avaliar a aprendizagem proporcionada pelos algoritmos *Q-Learning* e *SARSA(λ)*, foram coletados os valores estimados da função $Q(s, a)$ para cada par estado/ação (s, a) do problema. Já no caso do algoritmo *Actor-Critic*, foram coletados os valores de utilidade de cada estado s do problema. A Tabela 1 apresenta a média destes valores na SBA *Cliff Walking*, considerando 5 execuções por algoritmo. Os parâmetros adotados para cada algoritmo foram: *Q-Learning*: $\alpha = 0,1$, $\gamma = 0,3$, 500 episódios de aprendizagem; *SARSA(λ)*: $\alpha = 0,5$,$\gamma = 0,5$,$\lambda = 0,9$, 700 episódios de aprendizagem; *Actor-Critic*: $\alpha = 0,5$,$\gamma = 1$, $lambda = 0,3$, 100 episódios de aprendizagem.

A Tabela 2 apresenta a média destes valores na SBA *Mundo 4x3*, também considerando 5 execuções por algoritmo, sendo que os parâmetros adotados em cada algoritmo foram: *Q-Learning*: $\alpha = 1$, $\gamma = 1$; *SARSA(λ)*: $\alpha = 1$, $\gamma = 1$,$\lambda = 0,9$; *Actor-Critic*: $\alpha = 1$,$\gamma = 1$, $\lambda = 1$; 500 episódios de aprendizagem em todos os algoritmos.

Nas duas SBAs, os parâmetros de aprendizagem e quantidade de repetições foram selecionados experimentalmente de modo que o agente aprendesse a política ótima. Cabe destacar que a avaliação realizada não tem o intuito de comparar o desempenho dos algoritmos ou encontrar seus parâmetros ótimos, mas sim verificar se a extensão está funcionando adequadamente.

No caso dos algoritmos *Q-Learning* e *SARSA(λ)*, a política aprendida em cada estado corresponde à ação que tem valor máximo de $Q(s, a)$, indicados em negrito nas tabelas. Pode-se verificar em ambas SBAs que o *Q-Learning* faz com que o agente aprenda a política do menor caminho até a célula de destino. Já o *SARSA(λ)* faz com que o agente aprenda uma política diferente do *Q-Learning* na SBA *Cliff Walking*, pois considera a segurança do agente, adotando um caminho que não passa próximo dos estados com recompensa mínima e que representam o penhasco (para evitar risco de queda no penhasco). Essas políticas aprendidas pelo *Q-Learning* e pelo *SARSA(λ)* estão consistentes com o que apontam [Sutton e Barto 2018] para a *Cliff Walking* e [Russel e Norvig 2004] para a *Mundo 4x3*. No *Actor-Critic*, a política aprendida pelo agente em cada estado corresponde à ação que o leva ao estado vizinho de maior utilidade. A partir das tabelas verifica-se que com o *Actor-Critic* o agente também aprende uma política que privilegia a segurança

Estado	*Q-Learning*				*SARSA(λ)*				*Actor-Critic*
(x,y)	Cima	Baixo	Esquerda	Direita	Cima	Baixo	Esquerda	Direita	Utilidade
(0,0)	**-1,42826**	-1,42837	-1,42833	-99,88934	**-1,99636**	-12,05645	-5,23845	-100,00000	-34,44597
(0,1)	-1,42772	-1,42802	-1,42790	**-1,42753**	**-1,99270**	-2,51677	-2,24678	-15,01358	-8,00000
(0,2)	-1,42694	-1,42694	-1,42690	**-1,42684**	-2,30813	-2,32880	-2,01280	**-1,98531**	-7,00000
(1,1)	-1,42611	-98,51834	-1,42668	**-1,42510**	**-1,76315**	-100,0000	-7,02749	-2,57476	-36,99150
(1,2)	-1,42530	-1,42520	-1,42526	**-1,42512**	-3,20844	-8,05625	-2,33142	**-1,98297**	-6,00000
(2,1)	-1,42172	-96,09352	-1,42283	**-1,41700**	**-1,98662**	-99,99982	-12,74462	-19,45179	-47,34018
(2,2)	-1,42094	-1,42080	-1,42122	**-1,42067**	-2,77747	-3,91706	-2,12292	**-1,94073**	-5,00000
(3,1)	-1,40693	-91,22813	-1,40938	**-1,39000**	**-1,97260**	-98,12012	-9,42074	-22,32461	-37,33405
(3,2)	-1,40961	**-1,40914**	-1,40976	**-1,40914**	-2,18766	-11,85778	-2,09680	**-1,88126**	-4,00000
(4,1)	-1,36837	-88,26094	-1,37723	**-1,30000**	-1,90400	-99,32617	-10,46501	**-1,50151**	-34,25671
(4,2)	-1,38097	-1,37830	-1,38035	**-1,37827**	-1,93889	-9,60968	-1,96275	**-1,76251**	-3,00000
(5,1)	-1,26880	**-1,00000**	-1,28172	-1,19679	-1,78433	**-1,00000**	-7,38230	-1,50117	-1,00000
(5,2)	-1,30143	**-1,29590**	-1,30105	-1,30987	-1,96631	**-1,50002**	-1,90512	-1,80811	-2,00000

Tabela 1. Resultados na SBA *Cliff Walking*

Estado	*Q-Learning*				*SARSA(λ)*				*Actor-Critic*
(x,y)	Cima	Baixo	Esquerda	Direita	Cima	Baixo	Esquerda	Direita	Utilidade
(0,0)	**0.840**	0.800	0.800	**0.840**	**0.790**	0.631	0.556	0.359	0,840
(0,1)	**0.880**	0.800	0.840	0.840	**0.841**	0.526	0.683	0.644	0.488
(0,2)	0.880	0.840	0.880	**0.920**	0.541	0.589	0.670	**0.910**	0.952
(1,0)	0.840	0.840	0.800	**0.880**	-0.078	-0.434	**0.691**	-0.001	0,488
(1,2)	0.920	0.920	0.880	**0.960**	0.794	0.777	0.662	**0.952**	0.976
(2,0)	**0.920**	0.880	0.840	0.840	-0.812	-0.252	**0.652**	-0.486	-0,256
(2,1)	**0.960**	0.880	0.920	-1.000	**0.925**	0.297	0.184	-0.800	0.208
(2,2)	0.960	0.920	0.920	**1.000**	0.960	0.865	0.821	**1.000**	1,000
(3,0)	-1.000	0.840	**0.880**	0.840	-1.000	-0.424	**0.298**	-0.305	-0,216

Tabela 2. Resultados na SBA *Mundo 4x3*

de não passar próximo dos estados com recompensa mínima. De forma geral, verifica-se que todas as políticas aprendidas são consistentes com o que cada algoritmo se propõe, evidenciando a consistência da refatoração da extensão.

De forma geral, os resultados obtidos evidenciam que a refatoração da extensão e integração com a biblioteca BURLAP foi bem sucedida. Deste modo, a extensão NetLogo agora disponibiliza três algoritmos de RL para uso em SBAs.

5. Conclusão

Este artigo apresentou a refatoração da extensão NetLogo para aprendizagem por reforço. A extensão foi alterada para utilizar a biblioteca BURLAP já amplamente testada. Também foram incoporados outros dois algoritmos, fazendo com que a extensão suporte *Q-Learning*, *SARSA(λ)* e *Actor-Critic*. Para validar a refatoração, foram implementados duas SBAs: *Cliff Walking* e *Mundo 4x3*. Verificou-se que nessas configurações, os algoritmos *Q-Learning*, *SARSA(λ)* e *Actor-Critic* convergiram às políticas esperadas para esses algoritmos. Esses resultados demonstraram que a refatoração realizada para utilizar a BURLAP não introduziu inconsistências na extensão e permitiu a incorporação bem-sucedida dos algoritmos *SARSA(λ)* e *Actor-Critic*.

Como sugestão para trabalhos futuros, recomenda-se realizar uma avaliação da extensão em diferentes SBAs com o objetivo de demonstrar sua aplicabilidade em diversos cenários de aprendizagem por reforço. Além disso, seria interessante incorporar outros algoritmos de aprendizagem disponíveis na BURLAP, a fim de expandir as opções

e possibilidades da extensão. Essas melhorias promoveriam a versatilidade e o potencial de aplicação da extensão em um contexto mais amplo.

Os autores agradecem a FAPESC pelo apoio financeiro recebido (TO2021TR882 e TO2023TR246).

Referências

Bazzanella, E. e Santos, F. (2021). Does a q-learning netlogo extension simplify the development of agent-based simulations? In *Anais do XV Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações*, pages 1–12, Rio de Janeiro.

Bonabeau, E. (2002). Agent-based modeling: methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(3):7280–7287.

CoMSES (2020). CoMSES Catalog. `https://catalog.comses.net/visualization/`, Acesso em: Jul/2020.

Epstein, J. e Axtell, R. (1996). Growing artificial societies: social science from the bottom up. *Brookings Institution Press*.

Konda, V. e Tsitsiklis, J. (2000). Actor-critic algorithms. *Advances in neural information processing systems*, pages 1008–1014.

Kons, K. (2019). *Biblioteca Q-Learning para desenvolvimento de simulações com agentes na plataforma NetLogo*. Trabalho de conclusão de curso, Universidade do Estado de Santa Catarina (UDESC).

Macal, C. M. e North, M. J. (2010). *Introduction to agent-based modeling and simulation*. Springer Science & Business Media.

MacGlashan, J., Loftin, R., Littman, M. L., e Roberts, D. L. (2018). BURLAP: Brown-UMBC Reinforcement Learning and Planning. `burlap.cs.brown.edu/`.

Macy, M. W. e Willer, R. (2002). The factors affecting cooperation in a social dilemma: A multiagent simulation. *Computational & Mathematical Organization Theory*, 8(3):187–207.

Peters, J. e Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190.

Rummery, G. e Niranjan, M. (1994). On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*.

Russel, S. e Norvig, P. (2004). *Inteligência Artificial*. Rio de Janeiro: Campus, 2 edition.

Sklar, E. (2007). Software review: Netlogo, a multiagent simulation environment. *Journal of Artificial Life*.

Sutton, R. S. e Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tisue, S. e Wilensky, U. (2004). Netlogo: Design and implementation of a multi-agent modeling environment. In *Proceedings of agent*, volume 2004, pages 7–9. Springer.

Wang, X.-S., Cheng, Y.-H., e Yi, J.-Q. (2007). A fuzzy actor–critic reinforcement learning network. *Information Sciences*, 177(18):3764–3781.

Watkins, C. J. C. H. e Dayan, P. (1992). Q-learning. *Machine learning*, 33(3–4):279–292.

Wilensky, U. (1999). NetLogo. `http://ccl.northwestern.edu/netlogo/`.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Utilizando Modelos de Decisão Ética em Simulação de Agentes*

Gabriel Galvan Neres[1], André Pinz Borges[1], Gleifer Vaz Alves[1]

[1]Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná(UTFPR)
Ponta Grossa – PR – Brazil

neres@alunos.utfpr.edu.br, apborges@utfpr.edu.br, gleifer@utfpr.edu.br

Abstract. *This work explores simulations involving multi-agent systems. Our major goal is to represent a didactic story: The Ant and the Grasshopper by using three different ethical approaches. The modelling and simulation of this problem were done with the NetLogo tool, where some parameters were used to explore some possible extensions of this problem. Our work includes the analysis of results obtained from two different scenarios, where it is possible to draw comparisons between the three ethical models: consequentialism, deontology and virtue ethics.*

Resumo. *Este trabalho tem como objetivo explorar simulações envolvendo sistemas multi-agentes. O objetivo principal é representar a fábula da formiga e da cigarra por meio de três tipos de decisões éticas envolvendo os agentes do sistema. A modelagem e simulação desse problema foi realizada no NetLogo, com a determinação de alguns parâmetros para explorar possíveis extensões para esse problema. O trabalho apresenta a análise de resultados obtida de dois diferentes cenários, que permite delinear comparações entre os três modelos éticos implementados: consequencialista, deontológico e ética das virtudes.*

1. Introdução

Sistemas Multi-Agentes (SMAs) referem-se a uma área de estudo envolvendo agentes autônomos inseridos em um determinado contexto [Wooldridge 2009]. É possível analisar e entender diferentes sociedades compostas por diferentes indivíduos, com diferentes mentalidades. Isso é possível fazendo simulações entre agentes. Simulações de agentes é uma importante ferramenta para facilitar a visualização e compreensão de SMAs. Essas simulações podem ser utilizadas nas mais diversas áreas de aplicação como mudanças em diferentes volumes de água podem afetar o solo [da Silva Leitzke and Adamatti 2023] e simulação da evolução da cadeia alimentar [Armstrong and Norling 2023].

Ainda dentre as diferentes áreas de aplicação de simulação de agentes, encontra-se a simulação de problemas éticos. A modelagem ética abrange diversas áreas envolvendo temas e tópicos discutidos na sociedade. Em [Hills 2006] é demonstrado um exemplo de simulação envolvendo aspectos éticos, porém sem o uso específico de uma teoria ética, consistindo em simular agentes em um problema envolvendo cooperação e altruísmo.

Por outro lado, [Bench-Capon 2020] apresenta a utilização de três diferentes modelos éticos (baseados em teorias éticas): Consequencialista, Deontológico e Ética das

Virtudes. Bench-Capon faz uso desses modelos usando o cenário da fábula da cigarra e da formiga.

Em [Markovicz and Alves 2023] são apresentados estes mesmos três modelos éticos utilizando uma modelagem formal por meio de autômatos temporais.

O trabalho aqui apresentado tem como objetivo principal estender os modelos formais definidos em [Markovicz and Alves 2023] de forma a criar modelos de simulação de agentes na ferramenta NetLogo [Wilensky 1999] para realizar experimentos e comparar os três modelos éticos. De forma geral são modelados e simulados os três modelos éticos citados anteriormente, cujo problema básico é a tomada de decisão da formiga em dar (ou não) comida para a cigarra.

Com a realização dos testes foi possível observar que o modelo Deontológico obteve os resultados com maior número de mortes (das cigarras). Por outro lado, o modelo Consequencialista foi o que apresentou o maior número de sobrevivência das cigarras.

O artigo apresenta uma explicação a respeito dos modelos éticos na seção 2. A seção 3 contém a descrição da modelagem do problema, além de uma explicação dos agentes e dos parâmetros contidos na simulação. Na sequência, na seção 4, tem-se uma análise dos resultados obtidos. Por fim, a seção 5 apresenta as considerações finais.

2. Modelos Éticos

Essa seção descreve os 3 modelos éticos (Consequencialista, Ética das Virtudes e Deontológico) em um cenário de simulação usando como inspiração a fábula da formiga e da cigarra [Bench-Capon 2020] e também usada como exemplo na verificação formal demonstrada em [Markovicz and Alves 2023]. Usando o cenário da fábula e os modelos éticos, é possível observar diferentes sociedades compostas por indivíduos que são determinados para uma decisão de compartilhamento de alimento, tendo como objetivo observar no longo prazo o desenvolvimento daquela sociedade, e fazer um comparativo de qual delas em determinada situação, possuiu uma maior taxa de sobrevivência.

A seguir os modelos éticos utilizados na simulação são descritos:

- Consequencialista: é filosoficamente baseado na tese de que a principal propriedade em uma decisão é a consequência que determinado ato vai causar na sociedade. Quando nos referimos a esse pensamento, as ações de um determinado agente é considerada uma decisão moralmente certa quando é examinado o resultado que causou essa ação. Nesse artigo, o modelo consequencialista foi implementado de maneira que o agente avalie uma tomada de decisão visando a melhor opção para o desenvolvimento da sociedade em conjunto de formigas e cigarras.
- Deontológico: é baseado em uma seleção de regras pré-estabelecidas, a definição de uma decisão moralmente certa é julgada justamente no cumprimento dessas normas. Neste artigo, o modelo deontológico foi implementado de tal forma que o agente é julgado com base em seguir fielmente as regras estabelecidas, porém com uma certa avaliação do contexto de sociedade que ele está inserido.
- Ética das virtudes: O modelo da ética das virtudes, mais antigo dentre os citados, é baseado em Virtudes e Vícios que permite perceber quando um determinado indivíduo faz um ato. Se for um ato bom é tratado como uma virtude e caso tenha sido um ato ruim é visto como um vício, a categorização do ato é definida

pelas consequências que o ato ira causar na sociedade, trabalhar é considerado um ato bom por ser um ato coletivista provendo alimento para o indivíduo, enquanto brincar por ser um ato individualista é considerado um ato ruim. Nesse artigo, o modelo da ética das virtudes foi tratado visando um julgamento na sociedade para identificação de agentes com Vícios e com Virtudes.

3. Modelagem do problema

A Fig. 1 apresenta o modelo de simulação desenvolvido no Netlogo, o qual permite ao usuário definir a quantidade de formigas, a probabilidade de trabalho, a quantidade de cigarras, a probabilidade de comida e também visualizar o comportamento das cigarras e formigas.

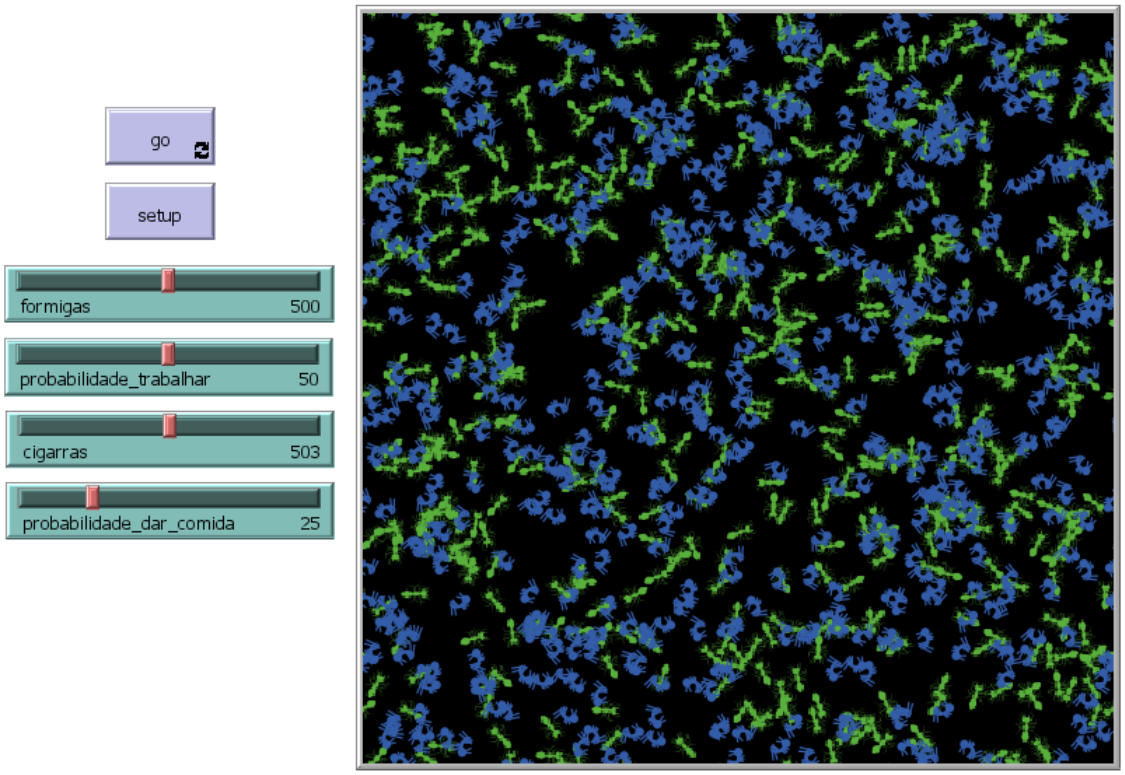


Figura 1. Primeira parte da composição da simulação

A Fig. 2 apresenta nove painéis que mostram a quantidade total de agentes, a quantidade de cigarras em cada uma de suas fases que são detalhadas na Sec. 3.1 e a quantidade de cigarras mortas. São mostrados também os resultados dos experimentos graficamente: proporção de cigarras vivas e mortas ao longo do tempo no gráfico “a”; a proporção de quantidade entre os estados das cigarras nos gráficos “b” e “c”; e o *balance* (que é um termo definido a seguir na seção 3.2), parâmetro médio das cigarras que ainda estão vivas na simulação no gráfico “d”.

3.1. Agentes

Neste trabalho são usados dois tipos de agentes (formiga e cigarra) seguindo o conto da cigarra e da formiga, onde uma formiga trabalhava e guardava comida enquanto a cigarra

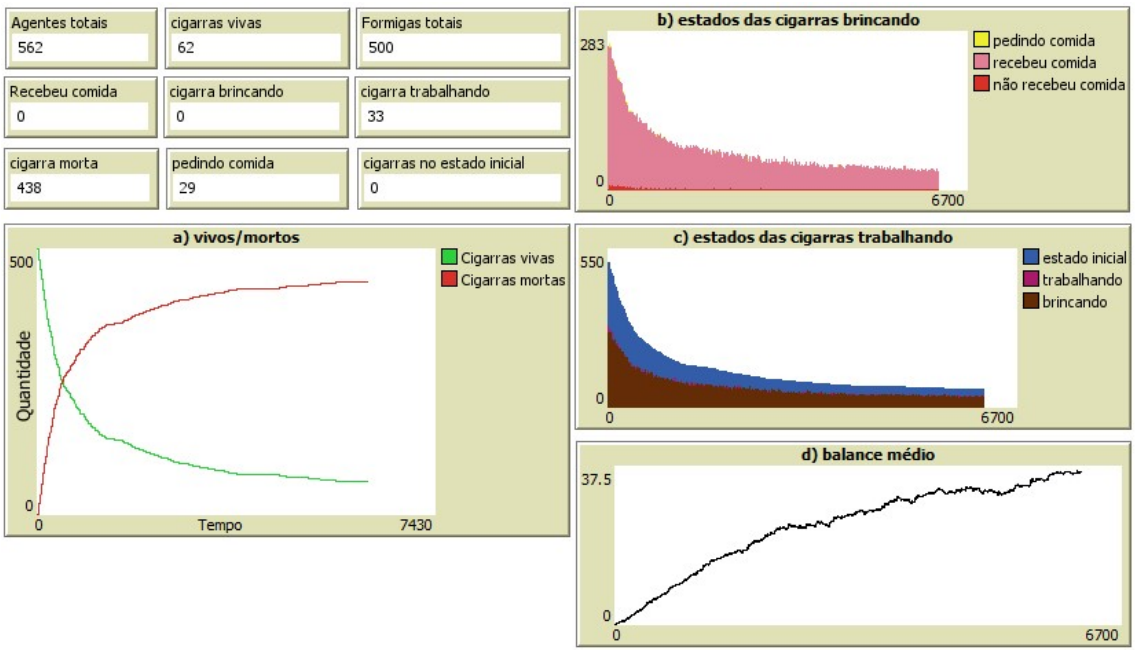


Figura 2. Segunda parte da composição da simulação

brincava, e então dependia da formiga compartilhar alimento para conseguir sobreviver. Neste modelo, formigas e cigarras possuem características e personalidades distintas em suas definições.

Um dos agentes presente nas simulações é a formiga (ver Fig. 3), entidades que sempre vão trabalhar e produzir alimento. As formigas são as que tomam a decisão se vão ou não compartilhar sua comida com as cigarras. Essa tomada de decisão é baseada no modelo ético simulado. Conforme o modelo usado, a formiga muda sua tomada de decisão. Outro fator que influencia a tomada de decisão é um parâmetro probabilístico determinado pelo usuário, para que uma formiga possua uma chance de dar comida às cigarras. Elas não possuem mais de um estado pois em nenhum momento tomam a decisão de trabalhar, por isso elas sempre se mantêm no estado inicial que não possui alterações de cor, ficando sempre na cor verde.

Figura 3. Representação gráfica de uma formiga

O segundo agente na simulação é a cigarra (ver Fig. 4). Cigarras são agentes que possuem a escolha entre trabalhar ou brincar, essa tomada de decisão possui um parâmetro probabilístico definido pelo usuário. Quando ela brinca e não consegue alimento, fica dependente das formigas para conseguir alimento e sobreviver. Diferente das formigas que possuem um único estado, as cigarras possuem diferentes estados representados pelas seguintes cores:

- **Azul**: o estado inicial. Nenhuma decisão foi tomada ainda neste estado e a cigarra pode decidir entre trabalhar ou brincar.

- **Violeta**: estado em que a decisão de trabalhar foi tomada. Como a cigarra trabalhou, ela não depende de comida das formigas, então ela está em situação de autonomia. A cigarra fica nele por 3 *ticks* (que determinam a contagem de interações da simulação), logo após isso, a cigarra retorna ao estado azul.
- **Marrom**: estado em que a decisão de brincar foi tomada. Como a cigarra brincou, ela não conseguiu comida, e fica dependente de uma formiga lhe fornecer alimento para conseguir sobreviver, com isso ela tem que fazer um pedido à formiga passando para o estado amarelo.
- **Amarelo**: estado em que a cigarra está pedindo comida à formiga. É nesse estado então que as cigarras que estavam no estado marrom e tinham uma formiga próxima para pedir comida vão ser avaliadas, em cada um dos critérios do modelo ético, para ver se vão ganhar comida ou não.
- **Rosa**: estado em que a cigarra recebeu comida da formiga, independente de qual motivo ou modelo ético. Neste estado a cigarra conseguiu sobreviver com a ajuda de uma formiga. Logo após o estado rosa ela retorna ao estado azul no qual o ciclo se reinicia.
- **Vermelho**: estado que a cigarra não está próxima de nenhuma formiga para pedir comida, ou quando as formigas próximas da cigarra acham que ela não é merecedora de receber comida. O estado vermelho simboliza o estado de morte do agente, então quando a formiga chega nesse estado é seu estado final, ela se encontra morta, após 1 *tick* nesse estado essa cigarra é removida da simulação.

Figura 4. Representação gráfica de uma cigarra no estado inicial

3.2. Parâmetros

Cada um dos agentes possui características de tomada de decisão, estabelecidas no início de cada experimento, que permitem modificar o comportamento dos agentes, sendo:

- ***Balance***: é a diferença entre as vezes que a cigarra brincou e as vezes que trabalhou. Quando uma cigarra trabalha, o seu *balance* incrementa em uma unidade, e quando ela brinca e deixa de trabalhar ela decrementa uma unidade. O *balance* é uma das variáveis que vai definir se a cigarra vai ganhar comida ou não.
- **Probabilidade de trabalhar**: é pré-definida pelo usuário antes de iniciar uma simulação. Esta propriedade se refere às cigarras, e define qual a porcentagem de chance de uma cigarra trabalhar ou brincar, por exemplo, caso haja uma probabilidade de 98% de trabalhar, a quantidade de cigarras que vai brincar e então depender de alimento vai ser de 2 a cada 100. Caso a probabilidade de trabalhar tenha sido definida em 10% uma minoria das cigarras estará trabalhando, sendo assim a maioria necessitando de alimento de formigas.
- **Probabilidade de dar comida**: probabilidade da formiga fornecer comida para a cigarra. Mesmo quanto a formiga avalia que uma cigarra não é socialmente válida para receber comida com base em seus conceitos éticos ela ainda possui uma probabilidade de fornecer comida para a cigarra.

- **Proximidade**: é uma das condições que existe, referente tanto a formigas quanto a cigarras. Considerando a situação em que uma cigarra precisa pedir comida para uma formiga, ela apenas vai conseguir fazer isso se a cigarra estiver próxima da formiga. Cada um dos agentes da simulação possui um raio próprio de tamanho definido em duas cigarras de distância, e somente quando uma cigarra está dentro do alcance do raio de uma formiga consegue solicitar comida.

3.3. Implementação

A implementação dos três modelos tem alguns aspectos similares entre si. O código está estruturado de maneira que analisa a tomada de decisões baseadas nos estados em que as cigarras se encontram na simulação, na condição se existe uma formiga próxima da cigarra ou não e a tomada de decisão ética é o que distingue cada um dos modelos.

O diferencial entre cada um dos modelos é implementado quando a cigarra está no estado amarelo. Neste ponto da simulação é feita uma análise dos parâmetros da cigarra, sendo avaliados para cada modelo:

- Ética das Virtudes: o *balance* e a quantidade de vezes que a cigarra brincou. Para ela receber o recurso (comida) a cigarra tem que ter brincado menos de 2 vezes ou o *balance* deve ser maior que 0. Caso um dos parâmetros seja verdadeiro a formiga fornece o recurso.
- Consequencialista: o *balance* deste modelo é mais permissível em relação à ética das virtudes, pois para uma formiga decidir fornecer comida para cigarra é necessário que ela tenha um *balance* maior que -1.
- Deontológico: a quantidade de vezes que a cigarra brincou. Esse modelo é rigoroso em relação aos outros. Uma formiga só vai dar comida para a cigarra, caso ela tenha brincado menos que duas vezes.

4. Experimentos

Ao todo foram realizados 12 experimentos que são divididos em dois blocos como mostra a Tabela 1. Além do *Modelo* simulado foi determinada também a quantidade de agentes (*Formigas* e *Cigarras*) de cada simulação. Outros parâmetros registrados são as questões probabilísticas: probabilidade da cigarra trabalhar (*Pr_Tra*) e a probabilidade da formiga dar comida (*Pr_Com*). Para registro dos experimentos realizados foi usada uma contagem de iterações da simulação (*ticks*) como controle de tempo, assim tendo uma percepção de quanto tempo foi registrado em cada comunidade. O resultado de cada experimento, indicado pela quantidade de cigarras que permaneceram vivas é apresentado na coluna *Res*.

No primeiro bloco (experimentos de 1 até 6) tem-se 2 testes para cada modelo ético. O mesmo serve para os experimentos de 7 ao 12 que compõem o segundo bloco de testes. Durante os 6 primeiros experimentos são analisadas as diferenças entre 2 probabilidades da simulação (50% e 25%), e nos 6 últimos experimentos é testado quando há uma diferença entre os tipos de agentes (formigas e cigarras) presentes na simulação.

4.1. Cenário A: Comparação de dois tipos de sociedades

Estes experimentos são baseados nas probabilidades da cigarra trabalhar e da formiga dar comida. Os parâmetros definidos para essa simulação de população são 500 formigas e 500 cigarras, para assim ter um equilíbrio entre os dois tipos de agentes.

Tabela 1. Parâmetros e resultados dos 12 experimentos realizados

Exp	Modelo	Formigas	Cigarras	*Pr_Tra* (%)	*Pr_Com* (%)	*ticks*	Res.
1	E.V.	500	500	50	25	50000	18
2	E.V.	500	500	25	50	24189	0
3	Deon.	500	500	50	25	1437	0
4	Deon.	500	500	25	50	23177	0
5	Con.	500	500	50	25	50000	29
6	Con.	500	500	25	50	24333	0
7	E.V.	200	800	50	50	3852	0
8	E.V	800	200	50	50	50000	177
9	Deon.	200	800	50	50	701	0
10	Deon.	800	200	50	50	50000	172
11	Con.	200	800	50	50	3484	0
12	Con.	800	200	50	50	50000	184

Foram realizados 2 experimentos, para cada um dos modelos éticos, para distinção entre os 2 tipos de probabilidades. A probabilidade de trabalhar é definida pela chance de uma cigarra conseguir o seu próprio alimento, assim trabalhando e não estando dependente de uma formiga para conseguir seu alimento (recurso). A outra probabilidade é a de que a formiga fornece comida mesmo que um determinado indivíduo não seja considerado merecedor de alimento, assim ficando em uma situação de dependência das formigas para conseguir o seu próprio alimento.

É possível fazer o comparativo entre 2 tipos de sociedade, uma sociedade em que os indivíduos são auto-suficientes e apenas uma parcela fica dependente de formigas para sobreviver. Então, é determinada uma probabilidade de 50% das cigarras conseguirem seu próprio alimento, e uma probabilidade de 25% de as formigas darem comida. E a outra simulação tem os valores invertidos, sendo 25% para as cigarras trabalharem e 50% para as formigas fornecerem alimento.

Comparando os experimentos da Tabela 1, é possível observar os resultados obtidos no Cenário A. No modelo Ética das Virtudes, que é referente aos experimentos 1 e 2, nota-se uma diferença em relação à sobrevivência das cigarras ao final da simulação, é perceptível uma maior taxa de sobrevivência dos indivíduos no experimento 1 em relação ao 2. As simulações 3 e 4, referentes ao modelo Deontológico, obtiveram em ambos testes a morte totalitária de sua população. Porém, observa-se que no experimento 4 a sociedade viveu por um tempo maior que no experimento 3. Os resultados obtidos nos experimentos 5 e 6 representam o modelo Consequencialista. Nesses resultados, observa-se que o experimento 5 apresenta quantidade de cigarras vivas ao fim da simulação, enquanto no experimento 6 toda a população de cigarra foi morta.

Durante os experimentos da Ética das Virtudes e Consequencialista é possível observar que os que tiveram maior taxa de sobrevivência são os que possuem indivíduos com mais independência. Porém, quando se analisam os resultados obtidos no modelo Deontológico, percebe-se um resultado oposto. Nos experimentos 3 e 4 tem-se a morte de toda população de cigarras, Mas, no experimento 3 a população foi morta em apenas *1.437 ticks*, ao passo que no experimento 4 foram *23.177 ticks*.

Os experimentos 3 e 5 apresentam resultados bem distintos. O experimento 3 apresenta o pior resultado enquanto o experimento 5 apresenta o melhor resultado (em quantidade de cigarras vivas). Observa-se na Fig. 5 a relação entre cigarras vivas e mortas ao longo do tempo da simulação. Ao fim da população chegou em um intervalo de tempo menor comparado ao tempo da Fig 6 (exp. 5). A diferença de valores ocorre devido ao modelo Deontológico ter um aspecto mais punitivo em comparação ao modelo Consequencialista.

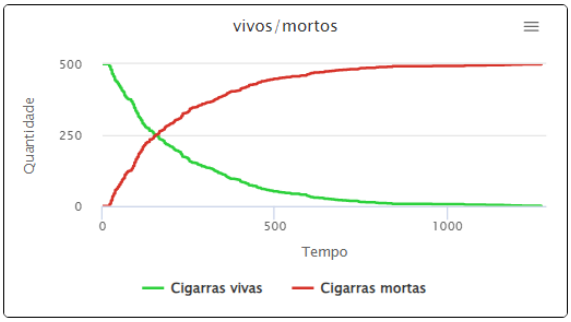


Figura 5. Gráfico de cigarras mortas do experimento 3

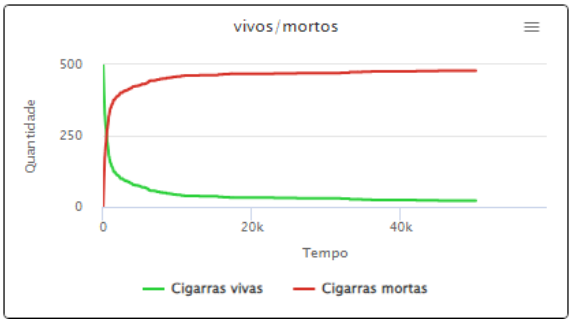


Figura 6. Gráfico de cigarras mortas do experimento 5

Observando os experimentos da Tabela 1 é possível fazer um comparativo entre os experimentos 2, 4 e 6 que são as simulações com 25% de trabalhar e 50% de dar comida. Logo, as sociedade com indivíduos dependentes de formigas. Com os resultados obtidos, nota-se que os três experimentos geraram resultados semelhantes. Todos tiveram a morte total de sua população em um valor aproximado de *24.000 ticks*. Isso demonstra a dificuldade de desenvolvimento da população em um ambiente de dependência e não de auto-sustentação.

4.2. Cenário B: Comparação entre 2 tipos de populações

Aqui o objetivo é abordar a quantidade de agentes na sociedade. Nesta simulação as questões probabilísticas foram fixadas em 50% para as cigarras trabalharem e 50% para as formigas fornecerem alimento, assim mantendo uma questão de igualdade entre os esses dois aspectos da simulação.

O Cenário B aborda também situações de superpopulação e de subpopulação, com uma determinada espécie com alta densidade populacional e outra com baixa densidade

populacional (800 formigas e 200 cigarras). Com isso, é possível analisar uma sociedade composta pela proporção de 4 indivíduos sustentando 1. E o segundo tem uma inversão dos valores populacionais. Contendo 800 cigarras para 200 formigas, fazendo então uma sociedade que uma pequena parcela sempre produz seu alimento e a maior parte da população então dependa de poucos indivíduos para a própria sustentação.

Conforme resultados da Tabela 1, todos os modelos tiveram resultados parecidos. Observando os experimentos 7, 9 e 11 referentes aos testes contendo 200 formigas e 800 cigarras, tem-se a morte totalitária da população. Quando se analisam os resultados dos experimentos 8, 10 e 12, onde as populações são compostas por 800 formigas e 200 cigarras, tem-se a sobrevivência da população com apenas algumas mortes no intervalo de tempo de *50000 ticks*.

Os experimentos 7 e 11 nos modelos Ética das Virtudes e Deontológico não tiveram diferença em comparação ao resultado da simulação 9, apesar desta possuir a mesma conclusão dos outros dois. Chegou ao fim da civilização em um tempo de *701 ticks*, sendo esse o menor intervalo de tempo dentre todos os 12 experimentos, algo que se é possível visualizar no gráfico da Figura 7 .

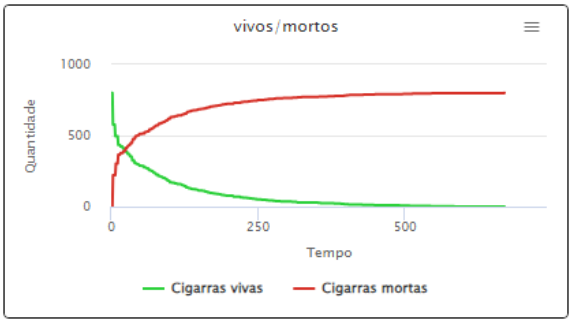


Figura 7. Gráfico de cigarras mortas do experimento 9

4.3. Discussões

Quando se observam os experimentos do cenário A, é perceptível que os melhores resultados de desempenho em termos de sobrevivência da população foram os experimentos do modelo Consequencialista. Por outro lado, os piores resultados foram os experimentos do modelo Deontológico devido à rigorosidade do modelo ético em relação às regras pré-estabelecidas visto que, caso o indivíduo faça algo contra os princípios ele não poderá receber os recursos solicitados.

Os modelos Consequencialistas de Ética das virtudes tiveram resultados parecidos, com uma taxa mais elevada de sobrevivência para o modelo Consequencialista. É possível perceber isso no experimentos 1 e 5, com uma diferença de apenas 11 cigarras devido ao fato de que uma cigarra é taxada como não merecedora de alimento quando está com *balance* menor que -1, enquanto no Ética das virtudes é com *balance* de 0.

É possível perceber analisando os experimentos do cenário B, que independente do modelo ético abordado, uma sociedade com superpopulação de cigarras não obtém um longo tempo de vida em comparativo à sociedade com subpopulação de cigarras, que obtiveram um resultado de longo prazo com apenas algumas mortes.

5. Conclusão

Nesse artigo foi implementada uma simulação baseada na fábula da formiga e da cigarra, envolvendo diferentes modelos éticos. O trabalho analisou a tomada de decisão de agentes com base em experimentos para realizar uma análise comparativa entre os modelos éticos e o impacto de suas decisões para a alocação de recursos entre os indivíduos da população (formigas e cigarras).

Os resultados apresentados possibilitam recomendar a utilização de um dado modelo ético, conforme o problema e cenário de aplicação apresentados. Em um cenário com uma sociedade composta por uma maioria de cigarras independentes (de formigas para obtenção de recursos) o modelo ético adequado para a sobrevivência das cigarras é o consequencialista, enquanto o deontológico é aquele que apresenta o pior resultado. Em outro cenário composta com a maioria de cigarras dependentes (das formigas) o resultado obtido foi a morte de toda população de cigarras em um semelhante espaço de tempo, independente do modelo ético utilizado. Assim, naturalmente, uma sociedade de cigarras com menor dependência de recursos é duradoura em comparação a uma que necessite de outros indivíduos (formigas) para sua sobrevivência.

Como trabalhos futuros, pretende-se estender e adaptar a modelagem para um cenário realista, onde tem-se veículos autônomos em cenários de tomada de decisões éticas em ambientes urbanos, envolvendo cruzamentos, pedestres, sinalizações e outros veículos.

Referências

Armstrong, S. and Norling, E. (2023). Reconsidering an agent-based model of food web evolution. In Lorig, F. and Norling, E., editors, *Multi-Agent-Based Simulation XXIII*, pages 70–81, Cham. Springer International Publishing.

Bench-Capon, T. (2020). Ethical approaches and autonomous systems. *Artif. Intell.*, 281(C).

da Silva Leitzke, B. and Adamatti, D. F. (2023). Land use management using multi-agent based simulation in a watershed in south of the brazil. In Lorig, F. and Norling, E., editors, *Multi-Agent-Based Simulation XXIII*, pages 1–15, Cham. Springer International Publishing.

Hills, T. T. (2006). Building"ethical agent"based simulations: A case study of a pathological problem in altruistic punishment. In *ALife Ethics Workshop Artificial Life X, Bloomington, IN, USA*.

Markovicz, J. V. and Alves, G. V. (2023). Modelagem formal de abordagens éticas para comportamento de agentes. *In Anais do Workshop-Escola de Informática Teórica (WEIT)*, pages 134–138.

Wilensky, U. (1999). NetLogo. `http://ccl.northwestern.edu/netlogo/`.

Wooldridge, M. (2009). *An Introduction to Multiagent Systems*. Wiley, Chichester, UK, 2 edition.

Simulação de Rotatitividade de Pessoal de TI através de um Sistema Multiagente: Desenvolvimento e Aplicações

Eduardo Ferreira[1,2], **Oscar Rete**[2]

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
Rio de Janeiro – RJ – Brasil

[2]Universidad de Ciencias Empresariales y Sociales (UCES)
Ciudad Autónoma de Buenos Aires – Argentina

Abstract. *Efficient staff turnover management is crucial for organizational competitiveness and innovation. In this context, this article presents a personnel turnover simulation using multi-agent systems, utilizing the GAMA platform to enable dynamic analysis of domain-relevant variables. For this purpose, we developed a model composed of agents representing IT professionals in a work environment. This model aims to identify the factors influencing employees' decisions to stay or leave the company. Through this simulation, the model represents complex interactions involving job satisfaction, external opportunities, organizational commitment, and Human Resources (HR) policies. The simulation reveals the effect of HR management strategies on turnover. The results provide an understanding of the problem and the opportunity to develop more effective retention strategies. This study contributes both to the academic knowledge on turnover and as a practical tool for HR managers.*

Resumo. *A gestão eficiente da rotatividade de pessoal é crucial para a competitividade e a inovação organizacional. Nesse contexto, este artigo apresenta uma simulação de rotatividade de pessoal por sistemas multiagentes, utilizando a plataforma GAMA para permitir uma análise dinâmica das variáveis relevantes ao domínio. Para isso, desenvolvemos um modelo composto por agentes que representam os profissionais de TI em um ambiente de trabalho. Esse modelo visa identificar os fatores que influenciam a decisão dos empregados de permanecer ou deixar a empresa. Através desta simulação o modelo representa as interações complexas que envolvem satisfação no trabalho, oportunidades externas, comprometimento organizacional e políticas de Recursos Humanos (RH). A simulação revela o efeito de estratégias de gestão de RH na rotatividade. Os resultados proporcionam compreensão do problema e oportunidade de desenvolvimento de estratégias de retenção mais efetivas. Este estudo contribui tanto para o conhecimento acadêmico sobre rotatividade quanto como ferramenta prática para gestores de RH.*

1. Introdução

No cenário corporativo, caracterizado pela rápida evolução tecnológica e pela constante mudança nas estruturas de mercado, a rotatividade de pessoal emerge como um desafio crítico para a estabilidade organizacional e a continuidade operacional. As organizações enfrentam o impacto significativo das altas taxas de rotatividade (*turnover*) de profissionais, que incluem custos elevados de recrutamento e seleção, perda de capital intelectual, e a deterioração do moral entre os empregados remanescentes [Chiavenato 2020].

Há condições do mercado de trabalho dos profissionais, principalmente no âmbito de tecnologia, como o trabalho remoto, o trabalho globalizado e a escassez de profissionais qualificados de tecnologia em diversos paises, inclusive no Brasil. Nas empresas de tecnologia da informação, a rotatividade de pessoal é comum devido à constante demanda por habilidades técnicas avançadas e à competição entre as organizações. Os profissionais de tecnologia da informação (TI) possuem habilidades altamente especializadas e podem encontrar emprego facilmente em outras organizações [Orozco et al. 2020] [Frufrek and Pansanato 2017]. O problema da rotatividade de pessoal já é um problema global [Orozco et al. 2020]. O macrossetor de Tecnologia da Informação emprega 2,02 milhões de profissionais correspondendo a 4% dos empregos do Brasil, com aumento de 117 mil novos postos de trabalho no ano de 2022 [BRASSCOM 2023].

A rotatividade não é uma causa em si, mas efeito de algumas variáveis externas e internas. Dentre as variáveis externas estão a situação de oferta e procura do mercado de talentos, a conjuntura econômica, as oportunidades de empregos no mercado de trabalho e etc [Chiavenato 2020]. Nesse contexto, os sistemas multiagentes (MAS - *Multi-Agent System*) [Wooldridge 2009] apresentam-se como um mecanismo promissora para a simulação de processos sociais complexos, incluindo o rotatividade de pessoal. Através desta pesquisa, propomos desenvolver e implementar um modelo computacional baseado em MAS na plataforma GAMA para explorar como as interações entre agentes e variáveis ambientais podem influenciar as decisões de permanência ou saída dos funcionários de determinada organização.

O artigo está estruturado inicialmente com uma revisão da literatura sobre as teorias de rotatividade de pessoal e trabalhos relacionados com sistemas multiagentes, estabelecendo assim a base teórica fundamental para nosso estudo. Após consolidar essa fundamentação teórica, detalhamos a metodologia empregada e a construção do modelo de simulação na plataforma GAMA, explicando como esses elementos metodológicos estão alinhados aos objetivos da pesquisa. Em seguida, os resultados da simulação são apresentados e discutidos. Finalmente, concluímos com uma reflexão crítica sobre as limitações do estudo e oferecemos sugestões para futuras pesquisas no campo da rotatividade de pessoal e sistemas multiagentes.

2. Fundamentação Teórica

Neste estudo, realizamos uma revisão da literatura sobre teoria de rotatividade de pessoal com o objetivo de estabelecer uma sólida base teórica e compreeensão do fenômeno. A compreensão das teorias relacionadas à rotatividade de pessoal e os trabalhos relacionados de sistemas multiagentes são fundamentais para o desenvolvimento da modelagem da solução, pois ao analisar estudos de rotatividade e o tema de sistemas multiagentes, identificamos técnicas de modelagem e simulação, aprimorando a precisão e a relevância da construção do nosso modelo. Essa abordagem assegura que as variáveis e processos modelados sejam pertinentes e baseados em evidências, viabilizando simulações que podem oferecer resultados de valor para a àrea de gestão de recursos humanos (RH).

2.1. Rotatividade de Pessoal

A rotatividade de pessoal, um fenômeno amplamente observado nas organizações modernas, refere-se ao processo pelo qual os empregados deixam uma organização e são

substituídos por outros. Compreender as causas e as consequências da rotatividade é crucial para a gestão eficaz de RH. Nesta subseção, estão relacionadas algumas das teorias predominantes na literatura que buscam explicar por que os empregados decidem deixar seus empregos. Em [Porter et al. 1974] estudaram as mudanças ao longo do tempo nas medidas de compromisso organizacional e satisfação no trabalho, cada uma relacionada à subsequente rotatividade entre 60 aprendizes de técnico psiquiátrico recentemente contratados. Esta teoria sugere que os empregados mantêm um equilíbrio entre o que contribuem para a organização e o que recebem em troca. Se perceberem que este equilíbrio está desproporcional, é mais provável que busquem oportunidades em outro lugar. O compromisso, a satisfação no trabalho e a percepção de equidade são fatores cruciais na teoria do Equilíbrio Organizacional. Assim, a teoria do Equilíbrio Organizacional sustenta que, na maioria das vezes, as pessoas que decidem sair de um emprego passam um tempo avaliando sua posição atual em comparação com suas futuras possibilidades. Este processo inclui desenvolver intenções sobre seus próximos passos e empreender ações para buscar novas oportunidades de trabalho. Esta teoria aborda desde a insatisfação no trabalho até o planejamento antecipado do caminho a seguir [Aranibar Gutiérrez et al. 2017].

Há trabalhos que investigam diretamente a rotatividade de pessoal em empresas do Brasil [Neto et al. 2016] [Soares et al. 2015] [Cardoso and Ferrando 2021] [Frufrek and Pansanato 2017]. Os autores [Neto et al. 2016] fizeram um estudo que investiga as opiniões de altos diretores de RH sobre a ação sindical em empresas brasileiras. Outros autores [Soares et al. 2015] desenvolveram um estudo de caso que examina a rotatividade dos servidores públicos na área de TI desde a perspectiva da gestão de pessoas. Já [Cardoso and Ferrando 2021] investiga a rotatividade de pessoal em empresas brasileiras de desenvolvimento de software através de uma pesquisa realizada a profissionais do setor. Outro estudo [Frufrek and Pansanato 2017] realizado que explora a relação entre as oportunidades percebidas de aprendizado, as intenções comportamentais e a retenção de empregados em organizações tecnológicas.

Para [Dornbusch et al. 2013], é caro para as empresas trocarem os trabalhadores devido aos custos de demissão, custos de contratação e custos de treinamento. Assim, as empresas de tecnologia do desenvolvimento de software costumam sofrer com a rotatividade de profissionais de tecnologia da informação, principalmente de desenvolvimento de software. Outro ponto, há também nas organizações, principalmente nas empresas brasileiras, que sofrem com a formalidade e a informalidade laboral, que para gestores de empresas e colaboradores [Beccaria and Maurizio 2018]. Em cada país há uma categorização distinta para considerar um trabalhador como formal ou informal. A rotatividade de profissionais de TI continua sendo um dos desafios mais persistentes enfrentados pelas empresas do setor, que estão ligadas à inovação contínua, como é o caso da indústria de desenvolvimento de software, onde o custo de substituição de uma vaga operacional é muito alto [Llamas et al. 2017].

Em meio à pandemia de COVID-19 em 2019 [ECLAC 2021] , temos algumas políticas de proteção das relações laborais e contratação de subsídios. As características de inovação e flexibilidade das empresas de tecnologia e seus profissionais muitas vezes habilitam as pessoas a trabalhar de formas não tradicionais, como um vínculo profissional mais livre. A legislação de trabalho não é a mesma em todos os países. As mudanças de modalidades de trabalho, formalidade, informalidade e flexibilidade são fatores de

atenção para conhecer os impactos e causa da rotatividade. Para compreender o contexto atual das empresas, principalmente no campo da tecnologia, é fundamental explorar os conceitos e as mudanças da transformação digital. Hoje nas empresas já vivem ou, provavelmente, vão começar no mundo da transformação digital [Albrieu et al. 2019]. Na medida em que as empresas, especialmente aquelas no setor de tecnologia, avançam nessa jornada da transformação digital, a complexidade dos desafios operacionais e organizacionais aumenta significativamente, exigindo soluções inovadoras que possam lidar com a complexidade e dinamismo desses novos ambientes digitais.

2.2. Abordagem em Sistemas Multi Agentes

Os sistemas multiagentes (MAS - *Multi-Agent System*) são amplamente utilizados no campo da inteligência artificial, incluindo controle de robôs, alocação de recursos, negociação e tomada de decisões [Arshad and Yao 2024]. No contexto de crescente complexidade de transformação digital a pesquisa em MAS se apresenta um caminho interessante para simular dinâmicas de simulações sociais complexas. Assim, a pesquisa em MAS tem como objetivo entender como um conjunto de entidades autônomas chamadas agentes pode se organizar para resolver problemas, alcançar tarefas e produzir fenômenos globais que os agentes não conseguem realizar sozinhos [Naciri and Tkiouat 2015]. O campo dos sistemas multiagente é altamente interdisciplinar: ele busca inspiração em áreas diversas como economia, filosofia, lógica, ecologia e as ciências sociais [Wooldridge 2009]. Através da simulação, os indivíduos podem receber feedback e recompensas, otimizando assim o seu próprio comportamento [Arshad and Yao 2024]. Com isso, MAS são definidos como plataformas que simulam ações e interações entre múltiplos agentes autônomos, cada um capaz de perceber e reagir ao ambiente de acordo com suas próprias regras de comportamento. A capacidade dos MAS de replicar a autonomia individual e a interdependência complexa entre os agentes torna-os particularmente úteis para modelar fenômenos sociais como a rotatividade.

Para esse trabalho, foi adotada a plataforma de modelagem e simulação GAMA (*GIS & Agent-based Modeling Architectures*) [Drogoul et al. 2013]. Além da GAMA, há outras plataformas do campo da modelagem e simulação baseada em agentes para auxiliar pesquisadores em MAS, como: *NetLogo*, *AnyLogic* e *Repast Symphony*. A *NetLogo* se trata de uma plataforma de uso mais acadêmico e modelagem de cenários menos complexos. A *AnyLogic* é conhecida por combinar simulação baseada em agentes e modelagem de sistemas dinâmicos, porém com uma interface de uso mais complexa. Por fim, outra plataforma que se assemelha bastante com a GAMA é a *Repast Symphony* amplamente utilizada em pesquisa acadêmica, desenvolvimento de políticas e modelagem de sistemas complexos. Considerando as alternativas de tecnologias de solução, o motivo para seleção da plataforma GAMA foi a independência de outras tecnologias ou bibliotecas, facilidade de implementação na linguagem GAML e a interface gráfica intuitiva para visualização das simulações. Essa plataforma de código aberto permite a definição de modelos baseados em agentes com representações complexas de ambientes e capacidades multi-nível genéricas. A modelagem baseada em agentes em múltiplos níveis requer a manipulação de agentes em diferentes níveis de representação com relação ao tempo, espaço e comportamento [Drogoul et al. 2013].

Na prática, a modelagem de comportamento em MAS utiliza regras e algoritmos baseados em estados internos dos agentes, como satisfação no trabalho, e fatores

externos, como condições do mercado. Isso permite análises detalhadas e simulações de intervenções antes de sua implementação real, ajustando políticas de RH para avaliar efeitos na satisfação e decisões dos funcionários. Além disso, o MAS ajuda a explorar cenários futuros sob diferentes condições, fornecendo uma ferramenta robusta para adaptar estratégias de RH, desenvolver políticas de retenção mais eficazes e minimizar riscos e custos associados à rotatividade.

3. Metodologia

Este estudo adota utilizando a plataforma GAMA [Drogoul et al. 2013] na modelagem de sistemas multiagentes para simular o domínio da rotatividade de pessoal de profissionais de TI em um contexto organizacional fictício de políticas de RH. Na Figura 1 abaixo apresenta o passso a passo do processo utilizado para construção do modelo de simulação na plataforma GAMA. A metodologia quantitativa se apresentou adequada por suportar o processamento e representação dos dados processados e analisados a partir das interações complexas e dinâmicas entre os agentes e ambiente, permitindo uma interpretação dos padrões de comportamento.

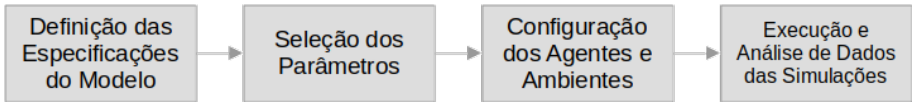


Figura 1. Processo de construção do modelo de simulação na plataforma GAMA

O processo começa com a construção de modelos de agentes, onde cada agente é representado por um funcionário e é atribuído características específicas como satisfação no trabalho, comprometimento organizacional, e percepção das políticas de RH. Esses atributos são inicialmente modelados usando distribuições probabilísticas, refletindo a variabilidade natural entre indivíduos em uma população de trabalho real.

As interações entre os agentes são definidas por regras que imitam comportamentos reais no local de trabalho, incluindo troca de informações sobre condições de trabalho, influência de colegas na percepção das políticas de RH e a dinâmica de suporte social. Este aspecto é crucial, pois as interações complexas e as redes de relacionamentos são essenciais para entender como os fatores sociais e organizacionais influenciam as decisões de rotatividade.

O ambiente simulado representa a organização e é caracterizado por variáveis como cultura organizacional, estrutura de liderança, e condições do mercado externo. Ajustes nesses parâmetros permitem explorar como diferentes contextos organizacionais influenciam o comportamento dos agentes, tornando possível simular uma variedade de cenários organizacionais para observação.

O processo de simulação é realizado em ciclos, cada um representando um período de tempo específico, como um mês. Durante cada ciclo, os agentes avaliam sua situação e tomam decisões baseadas em uma combinação de seus atributos internos, interações sociais, e influências ambientais. Este processo iterativo se repete até alcançar um estado de equilíbrio ou por um número predefinido de ciclos, permitindo a análise longitudinal dos comportamentos de rotatividade.

A abordagem quantitativa deste estudo, reforçada pela modelagem detalhada e análise de dados. Além de avançar no entendimento teórico sobre as causas da rotatividade, o modelo serve como uma ferramenta prática para os gestores de RH desenvolverem estratégias de intervenção baseadas em evidências para mitigar a rotatividade.

4. Modelo de Rotatitividade de Pessoal de TI

Nesta seção, será apresentado o modelo de rotatividade de pessoal de TI proposta como uma abordagem quantitativa para compreensão e previsão da probabilidade de um colaborador deixar a empresa com base em vários fatores influentes. Nas subseções abaixo, serão apresentadas as fórmulas que compõem o modelo e respectiva fórmula de cálculo para alcançar a taxa de rotatividade global.

4.1. Variáveis do Modelo

- s_i: Satisfação no trabalho do funcionário i.
- e_i: Nível de engajamento do funcionário i com a empresa.
- a_i: Afinidade do funcionário i com a equipe ou empresa.
- p_i: Percepção das estratégias de RH pelo funcionário i.
- w_i: Salário do funcionário i.
- b_i: Qualidade da relação do funcionário i com a chefia imediata.

4.2. Fórmula

A probabilidade de um funcionário deixar a empresa é modelada pela função logística:

$$P(\text{leave}_i) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 s_i + \beta_2 e_i + \beta_3 a_i + \beta_4 p_i + \beta_5 w_i + \beta_6 b_i)}}$$

onde $\beta_0, \beta_1, ..., \beta_6$ são os coeficientes do modelo que refletem o impacto de cada variável na decisão do funcionário de deixar a empresa.

4.3. Modelo de Taxa de Rotatividade Global

A taxa de rotatividade global (r) para a empresa é calculada como a média das probabilidades de saída ajustadas pelo tempo médio de permanência (t):

$$r = \frac{1}{N} \sum_{i=1}^{N} \frac{P(\text{leave}_i)}{t}$$

onde N é o número total de funcionários na empresa.

Este modelo fornece uma estrutura de análise e previsão da rotatividade de pessoal, permitindo uma abordagem sistemática para a gestão de RH na analise de taxa rotatividade. A sua aplicação pode ajudar as organizações a identificar e abordar os motivos e causas da rotatividade, contribuindo para priorização de estratégias de retenção de talentos.

5. Modelagem

A implementação deste sistema multiagente foi realizada utilizando uma plataforma GAMA, com simulação baseada em agentes, que permite a modelagem detalhada de interações individuais e coletivas dentro de um ambiente organizacional fictício. Esta seção descreve as etapas fundamentais seguidas para a configuração do sistema, incluindo a definição de agentes, a configuração do ambiente, as regras de interação, e os procedimentos de simulação.

Na Figura 2 é apresentado o diagrama de Modelo de Usuário relacionado com o meta modelo GAMA, com os relacionamento de Agente (*Agent*) , Modelo (*Model*) e Experimento (*Experiment*).

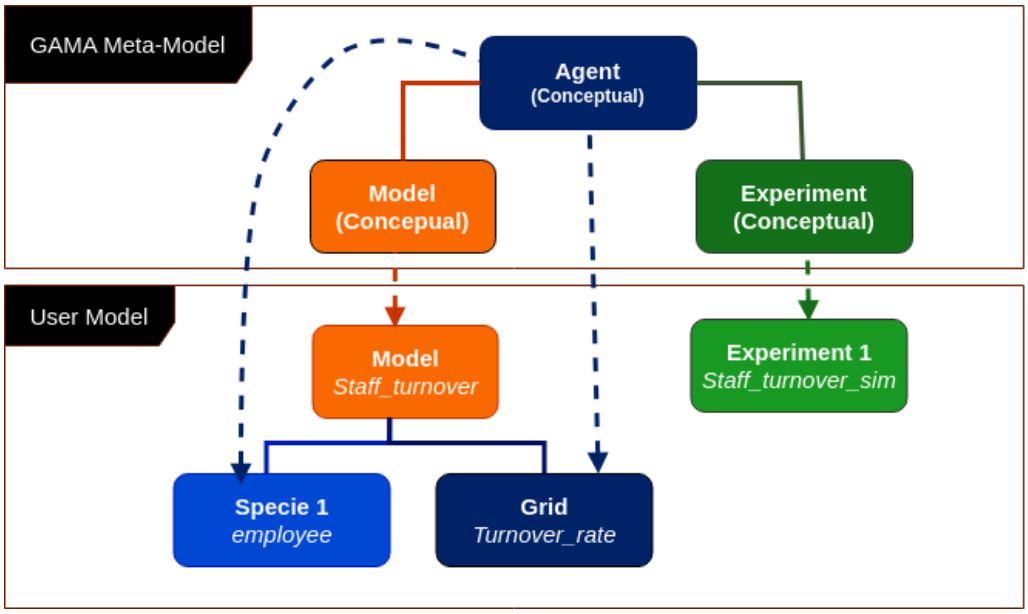


Figura 2. Gama Meta Modelo e Modelo do Usuário de Rotatividade de Pessoal

Nas subseções a seguir serão detalhados o que cada componente da modelagem representa.

5.1. Agente - employee

Cada agente no sistema representa um funcionário da organização (*Employee*). Os atributos de cada agente incluem: Satisfação no Trabalho: Influencia diretamente a probabilidade de um agente escolher deixar a organização. Nível de Estresse: Afeta a saúde mental e física do agente, impactando sua decisão de rotatividade. Comprometimento Organizacional: Determina o nível de lealdade do agente à organização e sua resistência a sair. Percepção das Políticas de RH: Modula como o agente vê a empresa em termos de suporte e justiça. Esses atributos são inicialmente atribuídos com base em distribuições probabilísticas que refletem a variabilidade encontrada nas características dos funcionários reais. Qualidade da relação com a chefia imediata: Modula como o agente vê considera boa a relação com a sua chefia imediata da organização. Esses atributos são inicialmente atribuídos com base em distribuições probabilísticas que refletem a variabilidade encontrada nas características dos funcionários inspirado na realidade.

5.2. Model Staff_turnover

O ambiente modelo *Staff_turnover* simula a empresa e é composto por diversos fatores que afetam todos os agentes: Cultura Organizacional: Define o contexto geral no qual os agentes operam e pode alterar os níveis de satisfação e estresse dos agentes. Estrutura de Liderança: Influencia o comprometimento organizacional e a percepção das políticas de RH dos agentes. Condições do Mercado Externo: Simula o mercado de trabalho fora da organização, afetando a decisão dos agentes de deixar a empresa por oportunidades externas.

5.3. Grid Turnover_rate

As interações entre agentes são definidas para simular a comunicação e influência entre colegas, que podem alterar os atributos dos agentes envolvidos. As principais regras incluem: Comunicação sobre Condições de Trabalho: Agentes podem compartilhar suas percepções, o que pode alterar o nível de satisfação ou estresse dos colegas. Influência Social: Agentes com alto comprometimento organizacional podem influenciar seus pares a perceber a empresa mais positivamente. Suporte Social: Interações que reduzem o estresse de agentes através do suporte emocional ou prático. Os resultados das variáveis e decisões dos agentes são representados pelo cálculo na *grid* de *Turnover_rate*.

5.4. Experiment Staff_turnover_sim

A simulação do experimento (*Staff_turnover_sim)* é executada em ciclos, cada um representando um intervalo de tempo (e.g., um mês). Em cada ciclo, os seguintes passos são realizados:

Avaliação de Estado: Cada agente avalia seu estado atual com base em seus atributos e no ambiente. Decisão de rotatividade: Cada agente decide se continuará na organização ou se sairá, baseado em sua satisfação, estresse, comprometimento e percepção das políticas de RH, junto com as condições do mercado externo. Atualização de Atributos: Os atributos dos agentes são atualizados com base nas interações do ciclo atual e nos efeitos do ambiente. Coleta de Dados: Dados são coletados para análise posterior, incluindo taxas de rotatividade, causas, e mudanças nos atributos dos agentes.

Após a execução da simulação por um número predefinido de ciclos, os dados coletados são analisados para identificar padrões de rotatividade, avaliar a eficácia de diferentes configurações de ambiente, e testar o impacto de potenciais políticas de RH.

6. Implementação na Plataforma GAMA

Este capítulo detalha a implementação do modelo de rotatividade de pessoal proposto utilizando a plataforma GAMA, um ambiente de simulação que facilita o desenvolvimento de modelos baseados em agentes. A implementação é explicada passo a passo para permitir a replicação e adaptação do modelo em outros contextos organizacionais.

6.1. Definição de Agentes

Cada agente no modelo representa um funcionário e é caracterizado pelas seguintes propriedades: *Satisfação no trabalho* (s_i); *Nível de engajamento* (e_i); *Afinidade com a equipe* (a_i); *Percepção das estratégias de RH* (p_i); *Concordância com o Salário recebido* (w_i); *Qualidade da relação com a chefia imediata* (b_i)

Os valores iniciais destas propriedades podem ser derivados de dados históricos da empresa ou de uma distribuição estatística assumida.

6.2. Modelagem do Comportamento dos Agentes

O comportamento dos agentes é modelado para refletir sua decisão de deixar a empresa. A decisão é calculada usando a função logística apresentada anteriormente no modelo. A simulação atualiza periodicamente as propriedades dos agentes com base em eventos ou interações no ambiente de trabalho.

```
P(leave_i) = 1 / (1 + exp(-(beta_0 + beta_1 * s_i +
             beta_2 * e_i +  beta_3 * a_i + beta_4 * p_i +
             beta_5 * w_i + beta_6 * b_i)));
```

6.3. Implementação do Modelo na Plataforma GAMA

Esta subseção descreve a implementação do modelo de rotatividade de pessoal na plataforma GAMA. Foi Utilizada uma abordagem baseada em agentes, onde cada agente representa um funcionário com atributos específicos que influenciam sua decisão de permanecer ou deixar a empresa.

6.3.1. Código GAMA

O trecho de código[1] abaixo exemplifica a definição global do modelo.

Listing 1. Código GAMA para Simulação de Rotatividade de Pessoal

```
model staff_turnover_sim
global{
    float worldDimension <- 5#m;
    geometry shape <- square(worldDimension);
    int nEmployeeLeft <- 0;
    int nNEmployeeInitial <- 100;
    float turnoverRateInitial <- nEmployeeLeft/
        ↪ nNumberBaloonInitial;
    reflex buildEmployee when: (flip(0.1)) {
        create employee number: nNEmployeeInitial;
    }
    reflex verifyTurnover when: ((nEmployeeLeft/
        ↪ nNumberBaloonInitial) > turnoverRateInitial) {
    ask turnover_rate {
        do update_color;
                }
        }
    reflex endSimulation when: nEmployeeLeft>1 {
        do pause;
    }
}
```

[1]Link do repositório: `https://anonymous.4open.science/r/staff_turnover-F611`

7. Resultados da Simulação

A simulação é executada para um período definido, e os resultados são coletados e analisados para avaliar a eficácia do modelo. As métricas de interesse incluem: Taxa de rotatividade global da empresa; Distribuição das probabilidades de saída de funcionários; e Impacto de estratégias de RH na redução da rotatividade. O GAMA oferece ferramentas de visualização para acompanhar a simulação em tempo real e analisar os resultados após a conclusão das simulações. Gráficos e mapas podem ser usados para apresentar insights significativos da análise. O parâmetro *Initial Number of Employees*: permite ajustar o número inicial de empregados. Elementos visuais como *display* e *monitor* são configurados para mostrar os agentes e monitorar a taxa de rotatividade ao longo do tempo. Vide exemplos de telas de simulação na Figura 3:

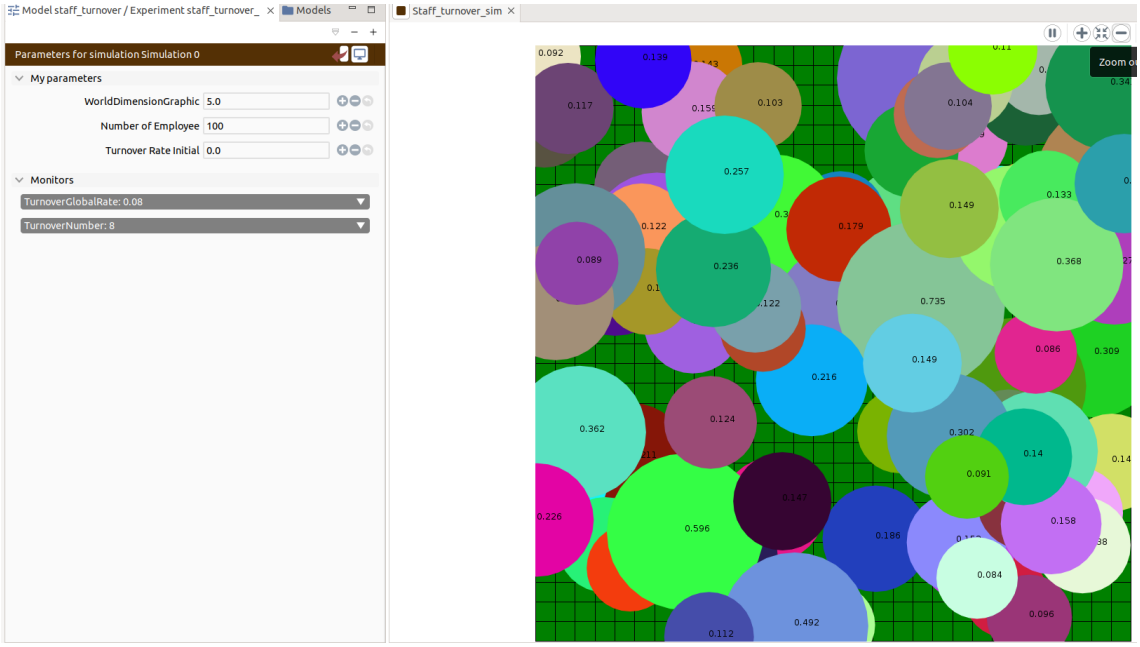


Figura 3. *Display* da Simulaçao de Rotatividade de Pessoal na Plataforma GAMA

Este modelo oferece uma plataforma para estudar como diferentes fatores podem influenciar a rotatividade dos funcionários em uma organização, permitindo simular e analisar potenciais estratégias para reduzir a rotatividade e melhorar a retenção de empregado.

A análise dos ciclos de simulação fornece insumos para interpretar o comportamento dos agentes, entender as dinâmicas de rotatividade, e avaliar a eficácia das políticas de RH implementadas.

Os fatores obsevados na análise possibilitam inferir que há influencia na rotatividade e destaca a complexidade das dinâmicas de retenção de funcionários. Podemos considerar que políticas de RH bem fundamentadas e estrategicamente implementadas podem significativamente mitigar a rotatividade. Este estudo não apenas fornece *insights* para gestores de RH sobre como estruturar suas políticas, mas também demonstra o potencial dos MAS como uma ferramenta poderosa para simulação e análise em ambientes organizacionais.

8. Considerações Finais

Este estudo demonstrou como a implementação de um MAS pode ser utilizada para simular e analisar o fenômeno do rotatividade de pessoal em ambientes organizacionais. Através da modelagem detalhada de agentes e do ambiente de trabalho, foi possível explorar as interações complexas que influenciam as decisões de permanência ou saída dos funcionários. A análise dos dados coletados durante as simulações forneceu insights sobre os principais fatores que contribuem para a rotatividade e a relação com políticas de RH do problema de rotação de pessoal.

Com a simulação realizada reforça a importância de abordagens proativas na gestão de RH, especialmente em contextos dinâmicos e competitivos. Políticas que melhoram a satisfação no trabalho, cuidem da relaçãon do colaborador com a equipe e o chefe, e fortalecem o comprometimento organizacional mostraram-se eficazes. Além disso, a capacidade de simular diferentes cenários permite aos gestores testar o impacto de potenciais estratégias organizacionais antes de implementá-las, oferecendo uma base para tomada de decisão.

Para avançar nas contribuições deste estudo, há alguns trabalhos futuros sugeridos a serem explorados: Integração de Variáveis Externas Complexas: Além das condições do mercado externo, futuras pesquisas podem incorporar variáveis econômicas e geopolíticas adicionais que influenciam o rotatividade, como mudanças legislativas e crises econômicas; Comparação de diferentes cenários de estratégias de RH; Exploração de Estratégias de RH Inovadoras: Estratégias emergentes, como trabalho remoto e programas de saúde mental, podem ser analisadas para entender seu impacto no comprometimento dos funcionários; Modelagem de Interdependências Mais Complexas: Aprofundar a modelagem das redes de relacionamento na organização, considerando alianças estratégicas e dinâmicas interpessoais; Uso de Dados Reais na Calibração de Modelos: Integrar conjuntos de dados reais na calibração dos modelos de simulação aumentaria a precisão das análises; e Interfaces Avançadas: Interfaces interativas permitiriam aos gestores de RH compreender melhor o impacto de mudanças nas políticas.

Essas sugestões de trabalhos futuros de pesquisa têm o potencial de enriquecer a simulação de sistemas multiagentes na gestão de RH, contribuindo para avanços teóricos e práticos na compreensão da rotatividade de pessoal e no aprimoramento das estratégias organizacionais.

Referências

Albrieu, R., Basco, A., Brest López, C., De Azevedo, B., Peirano, F., Rapetti, F., and Vienni, G. (2019). *Travesía 4.0: Hacia la transformación industrial Argentina*. CIPPEC.

Aranibar Gutiérrez, M. F., Melendres Carlos, V. D., Ramírez Barón, M. C., and García Rivera, B. R. (2017). Los factores de la rotación de personal en las maquiladoras de exportación de ensenada, bc (factors of staff turnover in textile factories of ensenada city, bc). *Revista Global de Negocios*, 6(2):25–40.

Arshad, M. and Yao, W. (2024). Human resource management decision support system based on multi agent. pages 1–5.

Beccaria, L. A. and Maurizio, R. D. L. (2018). Un análisis dinámico de los flujos de entrada a la formalidad en américa latina.

BRASSCOM (2023). Relatório setorial 2022 macrossetor de tic. Acessado em: 20 de março de 2024.

Cardoso, R. C. and Ferrando, A. (2021). A review of agent-based programming for multi-agent systems. In *Computers*, volume 10,4, page 16.

Chiavenato, I. (2020). *Gestão de Pessoas - O Novo Papel da Gestão do Talento Humano*. Grupo GEN, 5 edition.

Dornbusch, R., Fischer, S., and Startz, R. (2013). *Macroeconomia*, volume 1. AMGH, 13 edition.

Drogoul, A., Amouroux, E., Bommel, P., Gaudou, B., Grignard, A., Marilleau, N., Taillandier, P., Vavasseur, M., Vo, D. A., and Zucker, J.-D. (2013). Gama: A simulation platform that integrates geographical information data, agent-based modeling and multi-scale control. In *Principles and Practice of Multi-Agent Systems*, pages 242–258. Springer.

ECLAC (2021). Policies to protect labour relations and hiring subsidies amid the covid-19 pandemic. Employment Situation in Latin America and the Caribbean. LC/TS.2021/163.

Frufrek, G. L. and Pansanato, L. (2017). Rotatividade de pessoal: Pesquisa com profissionais de empresas brasileiras de desenvolvimento de software. *iSys - Brazilian Journal of Information Systems*, 13:05–29.

Llamas, S., Lopez Torres, V., and Moreno Moreno, L. (2017). Impacto del clima organizacional en la rotación del personal: Evidencia en sector desarrollo de software. *Revista Internacional Administración & Finanzas*, 10:49–61.

Naciri, N. and Tkiouat, M. (2015). Multi-agent systems: theory and applications survey. *Int. J. Intelligent Systems Technologies and Applications*, 14(2):145–167.

Neto, A. C., de Amorim, W. A. C., and Fischer, A. L. (2016). Top human resources managers' views on trade union action in brazilian corporations. *BAR - Brazilian Administration Review*, 13(4):e160066.

Orozco, D. G. A., Fernández, M. D., Arredondo, M., Manjarrez, N. I. R., and Ruiz, V. (2020). *ROTACIÓN DE PERSONAL ¿Qué es y cómo combatirla?*

Porter, L. W., Steers, R. M., Mowday, R. T., and Boulian, P. V. (1974). Organizational commitment, job satisfaction, and turnover among psychiatric technicians. *Journal of Applied Psychology*, 59(5):603.

Soares, M. L., Capistrano, A. G. A., and Barbosa, M. B. A. (2015). A rotatividade de servidores pÚblicos na Área de ti: Um estudo de caso sobre a Ótica da gestÃo de pessoas. V COLÓQUIO INTERNACIONAL DE GESTÃO UNIVERSITÁRIA –CIGU.

Wooldridge, M. J. (2009). *An Introduction to MultiAgent Systems*. Wiley, 2 edition.

Análise Comparativa de um Protótipo de IDE para Desenvolvimento de SMA Embarcados

Elaine Maria Pereira Siqueira, Gabriel Ramos, Thácito Raboni, Carlos Eduardo Pantoja, Nilson Mori Lazarin

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
Rio de Janeiro – RJ – Brazil

{elaine.siqueira, gabriel.ramos, thacito.medeiros}@aluno.cefet-rj.br

{nilson.lazarin, carlos.pantoja}@cefet-rj.br

Abstract. *Nowadays, using Integrated Development Environments (IDEs) in creating software processes is widely adopted and has become a necessity to increase the time optimization in the development process. However, when developing embedded multi-agent systems, the tools for automating the development process still require improvement. So, this paper evaluates a specific IDE for the development of cognitive hardware supported by robotic agents, the chonIDE. It analyzed its graphical characteristics, features, and benefits as a tool to facilitate Embedded MAS programming. Furthermore, this paper proposes a few changes in the chonIDE layout to improve the embedded MAS designer experience.*

Resumo. *Usar um Ambiente de Desenvolvimento Integrado (IDE) como ferramentas auxiliar na criação de softwares é uma necessidade, principalmente num contexto em que a otimização de tempo e trabalho são características valiosas no progresso do mercado tecnológico. Entretanto, as ferramentas de desenvolvimento de sistemas multiagentes (SMA) embarcados atravessam etapas que ainda carecem de atualização neste sentido, dessa forma, este trabalho apresenta uma avaliação da chonIDE. Foram analisadas suas características gráficas, funcionalidades e benefícios como ferramenta para facilitar a programação de SMA Embarcados. Além disso, são apresentadas sugestões de alteração no layout da ferramenta para melhorar a experiência do desenvolvedor.*

1. Introdução

O Ambiente de Desenvolvimento Integrado (IDE) fornece muitas ferramentas que suportam o processo de desenvolvimento de software, incluindo editores para escrever e editar programas e depuradores para localizar erros de lógica em programas [Deitel and Deitel 2009]. Antes da existência das IDEs, os códigos fonte eram escritos em editores de texto comuns, como o bloco de notas do celular, por exemplo. Em seguida, o código era conduzido a um compilador, responsável por traduzi-lo de linguagem de alto nível (usada pelos programadores) para linguagem simbólica, isto é, de baixo

nível (usada pelas máquinas), para ser lido pelos processadores do computador, e as respostas aos erros, caso houvesse algum, eram anotadas pelo programador, para que este pudesse corrigi-los, usando novamente o editor de texto. Caso não houvesse erros, o código era levado para execução, e, se ele apresentasse um comportamento diferente do esperado pelo programador, era direcionado a um depurador. Este, por sua vez, auxilia na identificação dos problemas ocorridos, uma vez que é utilizado para realizar testes no software, a fim de encontrar quaisquer problemas ou bugs no código. Por fim, o código volta ao editor de texto para ser aperfeiçoado [Hou and Wang 2009].

Sendo assim, é possível verificar que a criação de uma aplicação antes do surgimento das IDEs era complicado, haja vista que todo o procedimento previamente detalhado se repetia sempre que surgisse algum erro, o que, por consequência, diminuía a eficiência no processo de desenvolvimento de um software, enquanto aumentava as demandas de tempo e trabalho para integrar e gerenciar todas as funcionalidades citadas. O aparecimento das IDEs, tal qual Eclipse, VSCode, IntelliJ, entre outras, trouxe atualizações, como debug, instrumental integrado em single-screen, integração com versionadores de código, etc, que contribuíram para resolver alguns dos problemas já abordados. Esta inovação, entretanto, carece de equipagem específica para incorporar soluções de Inteligência Artificial (IA) em SMA, dada a natureza distribuída destes. O ambiente de desenvolvimento explorado neste trabalho, a chonIDE [Souza de Jesus et al. 2022, Souza de Jesus et al. 2023], por outro lado, resolve essa questão, mas sofre da ausência de ferramental que auxilie o projetista no desenrolar da aplicação.

O objetivo desse artigo é realizar um comparativo que permita analisar as principais funcionalidades da chonIDE, e outras IDEs atuais, para propôr avanços no contexto de programação de SMA Embarcados. Com esse intuito, serão retomadas definições que contribuem para o entendimento do cenário em que esse novo software atua e destacada como ele pode colaborar para melhorar tais circunstâncias. Além disso, foram identificados duas funcionalidades para serem desenvolvidas: o versionamento de código e o debug integrado.

Este trabalho está estruturado da seguinte forma: na Seção 2 são apresentados conceitos que enriquecem a compreensão do cenário abordado; na Seção 3 são expostas características que dizem respeito a identidade da chonIDE; na Seção 4 exibe tabelas que contribuem visualmente para a análise das limitações e diferenciais da chonIDE em contraste com as outras três IDEs mencionadas; a Seção 5 há uma explicação do motivo de considerar especificamente as IDEs Eclipse, VSCode e IntelliJ relevantes para a seção anterior; a Seção 6 evidencia um novo protótipo do software que é o foco deste documento e, finalmente, na Seção 7 são apresentadas as considerações finais e a indicação de trabalhos futuros.

2. Referencial Teórico

A essência dos sistemas computacionais autônomos é a autogestão, cuja intenção é libertar os administradores de sistema dos detalhes de funcionamento e manutenção desse sistema e fornecer aos usuários uma máquina que funciona com desempenho máximo 24 horas por dia, 7 dias por semana [Kephart and Chess 2003]. Um agente é um sistema reativo que exibe algum grau de autonomia, no sentido de que delegamos alguma tarefa a ele

e o próprio sistema determina a melhor forma de realizar essa tarefa [Bordini et al. 2007]. Esses agentes são denominados cognitivos, visto que, nos agentes, há uma representação explícita do ambiente e dos membros da sociedade. Eles podem raciocinar sobre as ações tomadas no passado e planejar as ações a serem tomadas no futuro [Sichman et al. 1992]. Para tanto, um agente pode: perceber seu ambiente ou comunicar-se com outros agentes.

Devido a habilidade social, que caracteriza os agentes, isto é, capacidade de realizar interações com outros agentes [Bordini et al. 2007], o mais comum no projeto de um software é que se trabalhe com mais de um agente em um mesmo ambiente. Os SMA são sistemas compostos por múltiplos agentes autônomos e proativos, com capacidade de tomada de decisão e comunicação com outros. Sendo assim, esses agentes podem cooperar entre si para alcançar um objetivo comum ao SMA [Wooldridge 2000], isto é, realizar tarefas complexas [Wooldridge 2009]. Apesar de autônomos, os agentes de um SMA eventualmente seguem normas e assumem papéis no grupo que estão contidos, permitindo alinhar seus objetivos individuais, os quais os delegamos, com os objetivos do sistema e, portanto, convergindo para o cumprimento de atividades complexas [Manoel et al. 2022].

Diferentemente dos computadores pessoais, de propósito geral e programados pelo usuário final, um sistema é classificado como embarcado quando este é dedicado a uma única tarefa e interage continuamente com o ambiente a sua volta por meio de sensores e atuadores [Ball 2002] que, por sua vez, compõem o dispositivo físico no qual o sistema está necessariamente encapsulado. Em outras palavras, um sistema embarcado é um sistema computacional embutido/dedicado ao hardware de um dispositivo e responsável pelo controle e monitoramento deste hardware, realização de processos e comunicações com outros dispositivos [Marwedel 2021]. Baseado em um microprocessador, projetado para controlar uma função ou uma gama de funções [Heath 2002], ele faz parte de um sistema ainda maior, para implementar alguns dos requerimentos deste sistema [IEEE 1990], por exemplo: em semáforos, aparelhos de ar-condicionado (controle da temperatura), impressoras, etc. As principais características de classificação de um sistema embarcado são a sua capacidade computacional e a sua independência de operação [Ball 2002], já que funciona de maneira exclusiva [Heath 2002], o que o torna destaque nestes sentidos.

O produto da união de um SMA, devido as suas qualidades de autonomia, proatividade e capacidade deliberativa, ao hardware de um *high-end IoT device* [Ojo et al. 2018] (um computador de placa única como a Raspberry Pi, por exemplo), por conta de suas capacidade computacional e independência de operação, denomina-se SMA Embarcado. Já existe, na literatura, uma arquitetura consolidada sobre esse tipo de sistema, que se secciona em: raciocínio, interfaceamento, firmware e hardware [Pantoja et al. 2016].

A camada de raciocínio, hospedada em um dispositivo dedicado [Pantoja et al. 2016, Souza de Castro et al. 2022], é a camada onde está situado o SMA e é responsável pela cognição, isto é, pela deliberação e execução de objetivos e intenções [Souza de Jesus et al. 2022]. A camada de firmware, hospedada em um ou mais microcontroladores [Lazarin and Pantoja 2015, Pantoja et al. 2016, Souza de Castro et al. 2022], é responsável por receber as mensagens do SMA para atuar no ambiente e envia as informações do ambiente para o SMA [Souza de Jesus et al. 2022]. Dessa forma, em um SMA embarcado, existem duas camadas de processamento: aquela responsável pela cognição dos agentes (onde o SMA está localizado) e aquela responsável pela camada de sensoriamento e atuação

(firmware).

É cabível ressaltar, ainda, que a construção de um SMA embarcado requer conhecimento em diferentes campos da programação e conceitos de eletrônica. Além disso, o desenvolvimento de software para equipamentos embarcados com SMA, mesmo seguindo métodos e especificações, nem sempre é uma tarefa fácil. Muitas vezes, depende-se de diferentes ferramentas para construir, compilar e analisar o código escrito em busca de erros de implementação, falhas de sincronia, problemas de alocação de memória, erros de cálculo, entre outros problemas possíveis [Schimuneck 2014]. Logo, para reduzir essas dificuldades que os designers de SMA atualmente enfrentam, principalmente para integrar e gerenciar diferentes IDEs na tentativa de incorporar e acompanhar o sistema, pode-se concluir que a utilização de uma IDE especializada é uma solução apropriada.

Portanto, tendo em consideração todo progresso que as IDEs trouxeram para a programação de software, analogamente, a chonIDE visa atender às necessidades das camadas de raciocínio e de firmware do SMA Embarcado, através da centralização das IDEs voltadas para essas camadas em um único ambiente de desenvolvimento, facilitando para os projetistas, desse modo, a embarcação do SMA. Considerando as demandas dos designers de SMA Embarcados, a chonIDE foi criada para ser um ambiente que reúne as ferramentas intrínsecas à construção de um SMA Embarcado em uma única IDE com todas as configurações indispensáveis já executadas, permitindo, dessa maneira, que o projetista não se distraia com a preparação do ambiente e concentre seu tempo e trabalho no projeto em si. Outrossim, a embarcação e o monitoramento do SMA, com a implementação da chonIDE, podem ser efetuados remotamente, dispensando, assim, ligações por fios que tornavam esse processo limitado e inviável dependendo da aplicação.

Independentemente da função executada por um SMA Embarcado, sua estrutura é dividida em dois conjuntos de componentes fracamente acoplados: um conjunto de componentes de hardware que inclui uma unidade central de processamento, normalmente na forma de um microcontrolador; e uma série de programas de software, normalmente incluídos como firmware que conferem funcionalidade ao hardware. Na tela principal da chonIDE (Figura 1) é possível gerenciar e desenvolver a cognição do SMA no qual os agentes se relacionam, e a lógica do firmware, que transforma os sinais analógicos ou digitais em informação [Souza de Jesus et al. 2022] para comandar os sensores e atuadores através do microcontrolador, possibilitando, assim, a interação com o ambiente no qual o dispositivo está inserido, conforme as deliberações dos agentes.

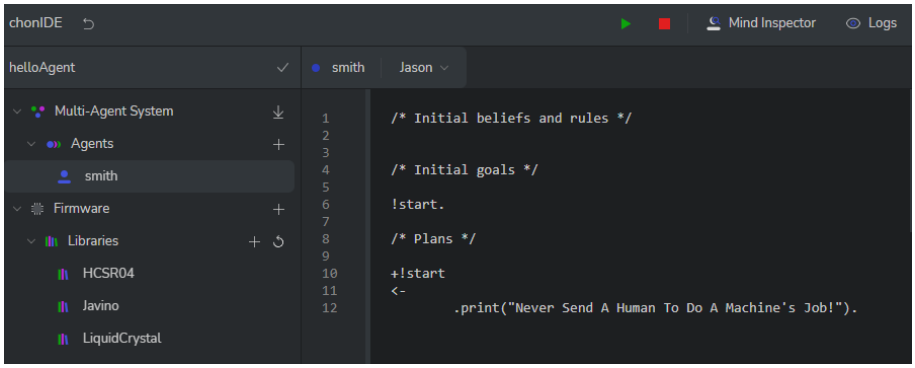


Figura 1. A tela principal da chonIDE.

A chonIDE, em sua versão atual, oferece a maneira tradicional de operação de uma IDE, na qual o usuário pode instalá-la em seu computador pessoal para desenvolver, enviar, iniciar ou interromper a execução do raciocínio, ao transmiti-lo para rodar em um protótipo robótico, e desenvolver, compilar ou implantar o firmware. Uma forma alternativa de operá-la, é acessando-a diretamente do protótipo, quando ela está rodando embarcada, via celular ou navegador.

3. Trabalhos Relacionados

Um trabalho sobre Ambientes de Desenvolvimento Integrado no apoio ao ensino da linguagem de programação Haskell foi considerado para justificar os critérios de análise que serão utilizados neste trabalho. Neste caso, a IDE Eclipse foi a mais recomendada para iniciantes, segundo diversas características julgadas desejáveis do ponto de vista de um usuário que já tem experiência em programação e está iniciando na linguagem Haskell. As particularidades citadas são: instalação, plataformas, licença de uso, manual do usuário, comunidade de usuários, comunidade de desenvolvedores, interface, compiladores ou interpretadores, depurador, editor, checagem de sintaxe, realce de sintaxe, autocompletar, ajuda rápida e indentador [Russi and Charão 2011].

Apesar de o Eclipse não servir para a mesma finalidade que a chonIDE, é aceitável utilizá-la na qualidade de modelo, pois o foco está na estrutura genérica de uma IDE e nas facilidades que essa mesma estrutura pode fornecer durante a programação, e não no propósito do desenvolvimento propriamente dito. Logo, ainda que o Eclipse não seja um ambiente para programação de SMA Embarcados, como é a chonIDE, é válido utilizá-lo como referência. Nessa mesma perspectiva, o VSCode e o IntelliJ foram considerados adequados, já que são ambientes voltados para a linguagem Java, assim como o Eclipse, e também são muito explorados no mercado de trabalho.

A partir da Tabela 1, este trabalho torna evidente o quanto a chonIDE ainda pode e precisa evoluir em versões futuras. Entretanto, mesmo em suas versões iniciais, já facilita a programação de agentes enquanto ocupa papel pioneiro no desenvolvimento de SMA Embarcados em um ambiente integrado.

Seguindo essa lógica, antes da análise da tabela, é importante expôr uma explicação dos critérios de análise envolvidos, retirados do artigo mencionado no início desta seção, para enriquecer a compreensão do leitor. A instalação do IDE deve ser fácil e rápida, para o usuário poder começar a utilizá-la o mais breve possível. É interessante que o IDE funcione nas plataformas de hardware/software mais comuns, incluindo Windows, Linux e MacOS. IDEs distribuídas sob licenças de software livre são preferíveis, por serem mais acessíveis. É desejável que o manual seja o mais completo possível, para o usuário poder sanar as dúvidas iniciais sem ter que pesquisar em fóruns. Uma comunidade numerosa é um ponto positivo, por facilitar a resolução de problemas que possam surgir. Geralmente, IDEs desenvolvidas por grupos trazem atualizações mais rapidamente que os desenvolvidos por uma só pessoa. A interface deve ser intuitiva, limpa e amigável.

É cobiçado que o depurador esteja integrado para facilitar a localização de problemas de lógica. Um editor poderoso com várias funcionalidades pode ajudar muito, por exemplo, com numeração de linhas para localização de erros reportados, múltiplas abas para acelerar a visualização de vários arquivos, linha atual destacada para facilitar a localização no arquivo, dentre outros recursos considerados abaixo. Checagem de sin-

taxe é um recurso muito útil, por avisar quando há algo errado no código antes que o usuário acione o compilador ou interpretador. Realce de sintaxe é um recurso importante, por mostrar o código de forma que fica fácil de visualizar e localizar funções, variáveis, constantes, strings, operadores, entre outros. Autocompletar auxilia na digitação do código, diminuindo os erros cometidos. Ajuda rápida são dicas mostradas quando o mouse aponta para determinadas regiões do código. Indentador é responsável pela formatação do código, auxiliando na sua organização e facilitando sua leitura [Russi and Charão 2011].

	chonIDE	Eclipse	VSCode	IntelliJ
Facilidade de Instalação	●●●○○	●●●●●	●●●●●	●●●●○
Multiplataforma	✗	✓	✓	✓
Multilinguagem	✗	✓	✓	✓
Livre para Uso	✓	✓	✓	✓
Interface clean	✓	✓	✓	✓
Debugger	✗	✓	✓	✓
Checagem de Sintaxe	✗	✓	✓	✓
Realce de Sintaxe	✗	✓	✓	✓
Autocompletar	✗	✓	✓	✓
Ajuda Rápida	✗	✓	✓	✓
Múltiplas Abas	✗	✓	✓	✓
Manual de Usuário	✗	✓	✓	✓
Visualizador de Arquivos	✓	✓	✓	✓
Contador de Linhas	✓	✓	✓	✓
Linha Atual Destacada	✗	✓	✓	✓
Indentação	✗	✓	✓	✓

Tabela 1. Comparativo de características entre as IDEs.

4. O Novo Protótipo da chonIDE

A partir da análise e comparação realizada entre a chonIDE e outras IDEs mais populares, foi possível mapear funcionalidades-base que são importantes para o desenvolvedor. O novo protótipo da chonIDE visa integrar essas principais necessidades para facilitar o processo de desenvolver e pensar um SMA.

Foram observadas três características que são comuns e consideradas como mais importantes inicialmente, seriam um console para visualização das informações de saída textuais, um depurador para visualização e acompanhamento das informações em tempo de execução e o versionamento através do Git. Essas soluções já existem de forma não integrada à IDE, sendo as duas primeiras providas pelo nativamente pelo Jason.

A disponibilização dessas funcionalidades na dinâmica de IDE possibilitaria que o desenvolvedor não precisasse sair da tela de desenvolvimento do SMA e tivesse a disposição esses recursos de visualização de informações de versionamento de forma limpa, intuitiva e prática, como proposto na Figura 2.

• ***Console integrado: Agent Tracer:*** O interpretador do Jason [Bordini et al. 2007] oferece uma interface interativa para visualização do SMA em execução, incluindo uma tela de registro de eventos e mensagens (logs) e outras ferramentas. Um dos recursos do chonOS — o sistema operacional para desenvolvimento SMA embarcado que suporta a IDE

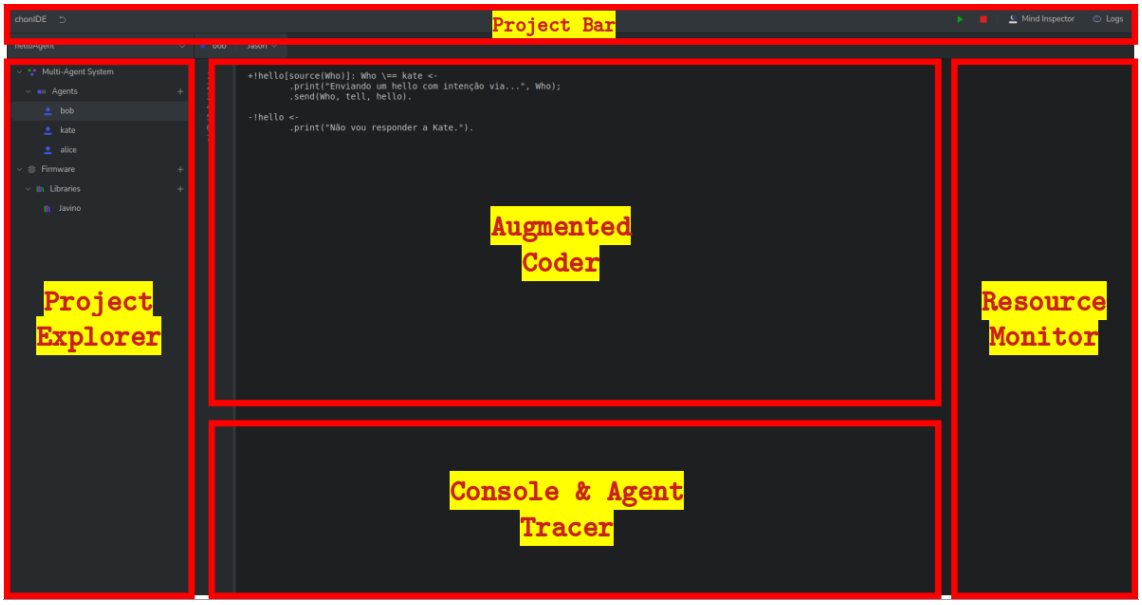


Figura 2. Os novos componentes da chonIDE.

— si captura o conteúdo textual dos logs e o escreve temporariamente em disco. Posteriormente, esse conteúdo é lido e disponibilizado em uma página web — por meio de uma aplicação chamada Wtee, escrita em Python — para a visualização desses registros em contextos embarcados e se mostrando uma solução mais reutilizável.

A integração com a IDE foi simples, a partir do aproveitamento desse recurso e visto que ambas as soluções já estão no contexto web. Ao incorporar a página do Wtee na interface da IDE, foi possível criar o componente gráfico Agent Tracer, permitindo então que o usuário não precise se deslocar da interface de desenvolvimento para visualizar os logs do SMA, simplificando e agilizando o processo de desenvolvimento.

• ***Depurador integrado: Mind Inspector integrado:*** O recurso de depuração de um SMA é existente e nativo do Jason, nomeado de Mind Inspector. É uma estrutura de páginas que exibe crenças, intenções e outras informações dos agentes. Inicialmente, na página principal, é exibida uma lista dos agentes em execução, e quando o usuário seleciona um agente, é redirecionado para uma página que se atualiza constantemente com novas informações dos ciclos de raciocínio. No entanto, essa estrutura não é integrada a IDE. Não era prático para o usuário alternar entre abas do navegador para conferir o estado do agente e funcionalmente também não era viável encaixar essa estrutura na interface do sistema. Para encontrar uma solução para essa integração, foi necessário entender como o Mind Inspector realizava esse fluxo e dispunha as informações. Para isso, foi realizado um estudo da arquitetura execução e controle do SMA no código-fonte do Jason e de seus conceitos principais, como eram dispostas crenças, intenções e planos no agente, como tudo se relacionava e onde se intercediam, além do funcionamento durante cada ciclo de execução. Como resultados do estudo, foram criados algoritmos de extração dessas informações em tempo de execução e conversão em modelos de dados diretos e intuitivos. Uma vez que os modelos de dados puderam ser instanciados com informações, a próxima etapa foi disponibilizá-los. Foi desenvolvido um mecanismo de histórico aplicável a todo agente, que armazena o estado do agente — o modelo de dados preenchido — em deter-

minado ciclo de raciocínio, e uma API Web consumida para retornar o estado do ciclo mais recente de todos os agentes do sistema ou para um agente e ciclo específico. Essa API é consumida pelo novo componente gráfico Mind Inspector na interface da IDE, localizado na barra lateral direita, que exibe, inicialmente, o estado do ciclo mais recente de todos os agentes. O usuário também pode alternar entre os ciclos de um agente específico, consolidando assim a proposta de mecânica de depuração orientada pelo ciclo do agente.

• ***Versionamento de Código:*** Devido à sua natureza distribuída e à necessidade de gerenciar múltiplas versões do projetos, os SMA Embarcados requerem um controle preciso do código-fonte. Nesse contexto, o versionamento de código desempenha um papel essencial, facilitando a colaboração entre os membros da equipe, o rastreamento de alterações e a preservação da integridade do código ao longo do desenvolvimento. A implementação proposta para a chonIDE do sistema de versionamento de código Git facilitará o processo de desenvolvimento interno e a integração com outras IDEs que suportam o desenvolvimento de SMA Embarcados. O Git é amplamente utilizado e suportado por várias ferramentas de desenvolvimento, permitindo uma colaboração eficiente entre desenvolvedores que utilizam diferentes ambientes de desenvolvimento. Essa interoperabilidade é fundamental para melhorar a padronização e a eficiência no desenvolvimento de sistemas complexos.

Para viabilizar o versionamento de código, propomos uma reestruturação das pastas e diretórios dos projetos dentro da IDE. Essa reorganização será fundamental para garantir que os arquivos estejam configurados corretamente para o controle de versão, facilitando o rastreamento das alterações ao longo do tempo e permitindo uma melhor integração com outras ferramentas e ambientes de desenvolvimento. Os componentes integrados à IDE permitirão aos desenvolvedores acessar e utilizar as funcionalidades do Git diretamente no ambiente de desenvolvimento. Isso incluirá a capacidade de iniciar e gerenciar repositórios Git, criar e gerenciar branches, realizar commits e sincronizar com repositórios remotos, tudo de maneira intuitiva e integrada. Em resumo, esta proposta de implementação do Git na chonIDE tem como objetivo melhorar o fluxo de trabalho de desenvolvimento, incentivando a colaboração entre diferentes ambientes de desenvolvimento e garantindo a segurança e a qualidade no desenvolvimento dos sistemas.

4.1. Componentização

No desenvolvimento de software, são pilares fundamentais a escalabilidade, o desacoplamento e a integração, os quais desempenham um papel importante em todo o processo de desenvolvimento. Em aplicações como uma IDE é importante que esses pilares sejam preservados, especialmente dada a amplitude de funcionalidades que esse tipo de ferramenta precisa oferecer, integrar em um mesmo contexto de forma coesa e garantir sua manutenção. Evidentemente, também reflete diretamente no desenvolvimento da interface e na experiência do usuário, dito isso, nesse contexto, foi adotado o desenvolvimento orientado a componentes.

Os frameworks de desenvolvimento front-end como o VUE.js, que é utilizado no desenvolvimento da IDE, um componente é uma unidade encapsulada, independente e replicável do código-fonte que representa visual ou interativamente um comportamento na interface. Deve ser aplicável em diferentes contextos ou conter um conjunto de funcionalidades e estados relacionados, configurando assim uma responsabilidade específica.

O processo de componentização foi conduzido em duas frentes principais: transformar as macrofuncionalidades da interface de desenvolvimento em componentes, a fim de facilitar a manutenção e a escalabilidade, e preparar o contexto que os une para integrar futuras funcionalidades com maior facilidade. Dessa forma, seria criado um sistema visual desacoplado, no qual as partes podem se expandir e mudar sem interferir no exterior, e novas partes podem ser integradas organicamente, permitindo também o desenvolvimento paralelo direcionados a esses macrocomponentes. O resultado desse processo foi a subdivisão do código-fonte em seis grandes componentes, cada um com seus próprios componentes internos: Barra Lateral Esquerda (Explorador), Agent Tracer, Codificador, Barra Lateral Direita (Mind Inspector e Firmware Boards) e Barra Superior. A Figura 3 apresenta a tela do novo protótipo[1][2] já com os componentes separados e implementados.

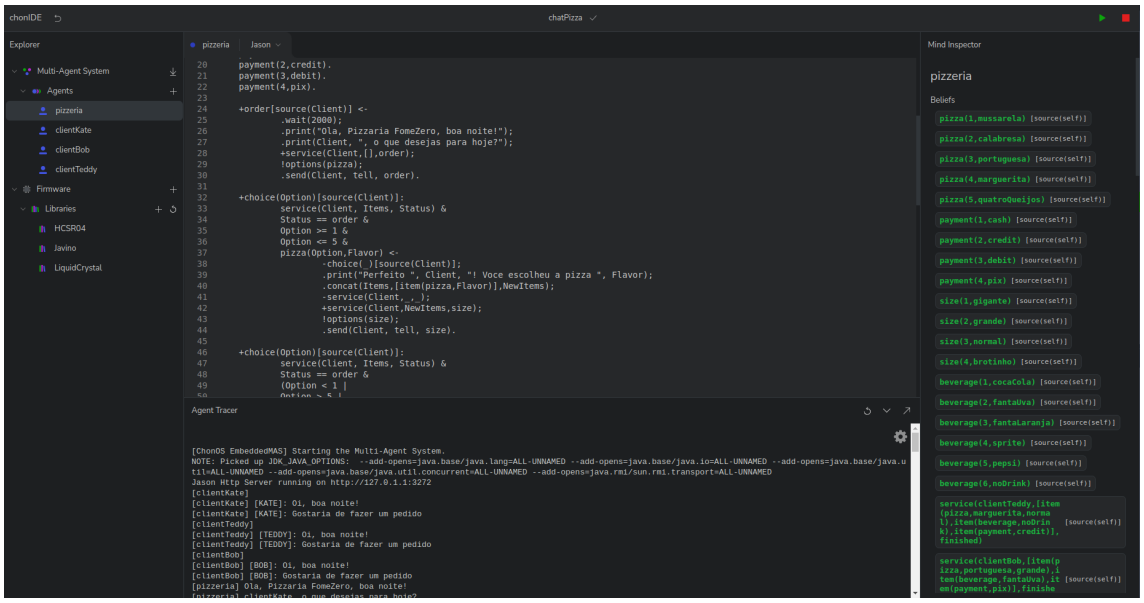


Figura 3. O protótipo em desenvolvimento da nova chonIDE.

Na Barra Lateral Esquerda, é encontrado o explorador de arquivos e pastas do projeto, onde é possível interagir com os estados relacionados ao projeto, adicionando, excluindo e modificando arquivos de agentes, firmwares e bibliotecas dentro de suas respectivas representações de pastas e arquivos. O Agent Tracer é onde o usuário pode visualizar todas as mensagens do SMA em execução no momento. O Codificador é onde ocorre a escrita e edição do código-fonte do arquivo selecionado no explorador de arquivos. Se houver um comportamento específico para o arquivo selecionado, ele será exibido no controlador de abas do codificador. Por exemplo, no arquivo do agente, é possível alternar o tipo do agente, enquanto no arquivo do firmware, é possível compilar e realizar o deploy para a placa selecionada no Firmware Boards.

A Barra Lateral Direita é alternada conforme o arquivo atualmente selecionado. Ao se tratar de um arquivo de firmware, é exibido o Firmware Boards, onde são exibidas todas as placas de firmware do sistema, e é possível selecionar para qual é desejado realizar os procedimentos de compilação ou deploy. Já ao se tratar de um arquivo de agente, é

[1]A versão online da chonIDE está disponível em https://ide.bot.chon.group:3270/chonide/login. É possível acessar utilizando o usuário *chon* e senha *ide*.

[2]O repositório da chonIDE está disponível em https://github.com/chon-group/chonIDE.

exibido o Mind Inspector, onde são exibidas todas as informações em tempo de execução de cada agente em cada ciclo, incluindo crenças, intenções, planos e outras informações. Por fim, na Barra Superior estão as informações gerais do projeto, como o nome e o indicador de salvamento automático. Além disso, são apresentados os controles gerais, como iniciar e parar o SMA do projeto e outras operações de manipulação do projeto.

5. Considerações Finais

Restrições técnicas são um desafio para os projetistas de SMA Embarcados. Um ferramental dedicado ao desenvolvimento de sistemas embarcados e distribuídos pode facilitar o processo de desenvolvimento do projetista. Dessa forma, este trabalho apresentou uma análise, restruturação e novas funcionalidades para que a chonIDE possa atenuar essas restrições em termos de agilidade e automatização de processos no desenvolvimento de SMA. Além disso, a chonIDE já traz benefícios relevantes para a integração hardware-software, através da viabilidade de projetos com a embarcação remota.

Como trabalhos futuros, pretende-se desenvolver um survey de usabilidade, aplicando a metodologia TAM (Modelo de Aceitação de Tecnologia), com uma turma da disciplina de Sistemas Especialistas do sétimo período do curso de Sistema de Informação do Cefet-RJ, a fim de sugerir pontos de aprimoramento para próximas versões da chonIDE, os quais serão refletidos em um protótipo contendo as implementações propostas, com destino a futuras avaliações, facilitando, dessa forma, ainda mais o desenvolvimento de embarcados e agentes.

Referências

Ball, S. (2002). *Embedded microprocessor systems: real world design*. Elsevier.

Bordini, R., Hübner, J., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. Wiley.

Deitel, P. J. and Deitel, H. M. (2009). *Java for programmers*. Pearson education.

Heath, S. (2002). *Embedded systems design*. Elsevier.

Hou, D. and Wang, Y. (2009). An empirical analysis of the evolution of user-visible features in an integrated development environment. In *Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research*, pages 122–135.

IEEE (1990). *IEEE Standard Glossary of Software Engineering Terminology*.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Lazarin, N. M. and Pantoja, C. E. (2015). A robotic-agent platform for embedding software agents using Raspberry Pi and Arduino Boards. In *Proceedings of the 9th Workshop-School on Agents, Environments, and Applications (WESAAC 2015)*, pages 13–20, Niteroi. UFF. `http://www2.ic.uff.br/~wesaac2015/Proceedings-WESAAC-2015.pdf`.

Manoel, F. C. P. B., dos Santos Amorim, M. C. M., and Pantoja, C. E. (2022). Um metamodelo para sistemas multiagentes embarcados considerando as dimensões multiagente, estrutural e comportamental. In *Anais do XVI Workshop-Escola de Sistemas*

de Agentes, Seus Ambientes e Aplicações (WESAAC 2022), pages 61–72, Blumenau. UFSC.

Marwedel, P. (2021). *Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things*. Springer Nature.

Ojo, M. O., Giordano, S., Procissi, G., and Seitanidis, I. N. (2018). A review of low-end, middle-end, and high-end iot devices. *IEEE Access*, 6:70528–70554.

Pantoja, C. E., Stabile, M. F., Lazarin, N. M., and Sichman, J. S. (2016). ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In Baldoni, M., Müller, J. P., Nunes, I., and Zalila-Wenkstern, R., editors, *Engineering Multi-Agent Systems*, pages 136–155, Cham. Springer International Publishing. https://doi.org/10.1007/978-3-319-50983-9_8.

Russi, D. F. and Charão, A. S. (2011). Ambientes de desenvolvimento integrado no apoio ao ensino da linguagem de programação haskell. *Revista Novas Tecnologias na Educação*, 9(2).

Schimuneck, T. (2014). Ambiente de desenvolvimento integrado para ensino e programação de microcontroladores da família mcs51.

Sichman, J. S., Demazeau, Y., and Boissier, O. (1992). When can knowledge-based systems be called agents? In *Anais do SBIA 92*, pages 172–185, Rio de Janeiro. SBC.

Souza de Castro, L. F., Manoel, F. C. P. B., Souza de Jesus, V., Pantoja, C. E., Pinz Borges, A., and Vaz Alves, G. (2022). Integrating Embedded Multiagent Systems with Urban Simulation Tools and IoT Applications. *Revista de Informática Teórica e Aplicada*, 29(1):81–90. DOI: https://doi.org/10.22456/2175-2745.110837.

Souza de Jesus, V., Lazarin, N. M., Pantoja, C. E., Manoel, F. C. P. B., Alves, G. V., Ramos, G., and Filho, J. V. (2022). Proposta de uma ide para desenvolvimento de sma embarcados. In *Anais do XVI Workshop-Escola de Sistemas de Agentes, Seus Ambientes e Aplicações (WESAAC 2022)*, pages 49–60, Blumenau. UFSC.

Souza de Jesus, V., Mori Lazarin, N., Pantoja, C. E., Vaz Alves, G., Ramos Alves de Lima, G., and Viterbo, J. (2023). An IDE to Support the Development of Embedded Multi-Agent Systems. In Mathieu, P., Dignum, F., Novais, P., and De la Prieta, F., editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*, pages 346–358, Cham. Springer Nature Switzerland. DOI: https://doi.org./10.1007/978-3-031-37616-0_29.

Wooldridge, M. (2000). Intelligent Agents. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1st edition.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John wiley & sons.

Introductory Guide to Agent-Based Simulation Development on the GAMA Platform[2]

Aline Rodrigues Santos[1], **Fernando Santos**[1]

[1]Departamento de Engenharia de Software
Universidade do Estado de Santa Catarina (UDESC)
Ibirama – SC – Brasil

aline.rs@edu.udesc.br, fernando.santos@udesc.br

Abstract. *The use of agent-based simulations is becoming common in the research field, being employed to abstract complex concepts through visual demonstrations. This has driven the emergence of platforms for developing these simulations. In this context, GAMA stands out as a good option due to its wide range of features. However, GAMA still lacks materials to guide beginner developers. With the aim of filling this gap, this paper presents an introductory guide to simulation development in GAMA. The paper describes the main functionalities and the structure for developing a simulation in GAMA. Beside that, the paper exemplifies these elements through the development of the 'Sugarscape' simulation, known in the community. Finally, the challenges that a beginner developer may encounter are reported, along with recommendations to address them.*

2

1. introduction

Agent-based simulations (ABS) have been used to replicate and study different phenomena. According to [Wilensky e Rand 2015], the adoption of ABS not only assists in visualizing complex scenarios but also in simplifying and abstracting them. An example of this approach is illustrated by [Costa 2023], who developed an ABS to demonstrate some of the main components of the complex Tupinambá society, allowing for the explanation of phenomena and demonstrating a tribal society that lived in Brazilian territory. Another example is the simulation of Covid-19 spread by [Teixeira e Santos 2020], which allowed for studying the effects of isolation measures on the pandemic.

[Kleiboer 1997] describes the different functions of ABS, two of which are research and support in teaching. This highlight the need for learning concepts and programming languages for ABS development. Although it is possible to develop ABS using general-purpose programming languages like Python or Java, the use of a specialized platform significantly simplifies learning and makes it more intuitive. For this reason, there are platforms that facilitate agent-based simulations development, such as NetLogo [Wilensky 1999] and, especially, GAMA [Taillandier et al. 2019].

The GAMA platform offers a programming language called GAML (Gama Modeling Language). Through GMAIL it is possible to develop simple or complex simulations, such as 3D models that provide control over various visual aspects such as lighting, texture and camera position or just 2D simple models, like Sugarscape. Additionally, the platform prioritizes the development of intuitive interfaces and allows for the provision of different visualizations that can be customized and updated dynamically to observe the simulation progress. GAMA also allows specifying monitors to collect and display data about the simulation during its execution. The presence of these features illustrates how the GAMA platform can streamline the development of ABS [Taillandier et al. 2019].

Despite offering various elements, a beginner in ABS interested in developing simulations on this platform may face challenges. The available documentation[1] for GAMA is extensive; however, the explanation of the essential elements for developing an ABS is dispersed across different topics, which poses comprehension challenges, especially for beginners. The learning curve can be steep for those who do not have prior experience with ABS platforms, requiring additional time and effort to familiarize themselves with the necessary elements for implementation. Therefore, there is a lack of an introductory guide that presents the essential elements for implementing ABS in GAMA using a simple and widely known simulation. Despite the GAMA documentation offering a tutorial on implementing the *Predator-Prey* simulation, it provides few explanations about the platform, its organization, the reasons for use, and the functionality of each element used.

This paper presents a guide to developing ABS in GAMA, which is based on the development of the *Sugarscape*, simulation, widely known in the ABS community. It explains the essential elements for implementing ABS in GAMA and exemplifies them through the *Sugarscape* simulation. The paper also provides recommendations for addressing some limitations of GAMA perceived during the simulation development.

The remaining of this paper is organized as follows. Section 2 presents the required background on ABS, the GAMA platform, and the *Sugarscape* simulation. In section 3 we present the introductory guide to the GAMA platform throghout the development of the *Sugarscape* simulation. Next, section 4 points out recommendations for beginners to start developing in GAMA. Finally, section 5 presents concluding remarks and future work.

2. Background

This section introduces concepts used in this paper. Firstly, it presents what ABS are. Then, the GAMA platform is explained, along with some of its various features. Later, the Sugarscape simulation is described, highlighting the main parts of its development.

2.1. Agent-based simulation

An agent-based simulation (ABS) is a practical approach to studying complex systems [Klügl e Bazzan 2012]. This approach allows researchers to predict and to optimize, these systems more effectively. Essentially, the simulation offers a tangible way to represent and explore complex situations, where agents play crucial roles, such as individuals, animals, or entities, interacting with each other and with a predefined environment.

[1] https://gama-platform.org/wiki/Home

According to [Klügl e Bazzan 2012], an ABS is composed of three elements: a set of autonomous *agents*, endowed with rules governing their behavior; the specification of *interactions* between agents and the environment, responsible for producing the overall output of the system; and the simulated *environment*, which contains all other simulation elements, such as resources and other objects without active behavior. The result of an ABS can be observed through visualization of agents behavior or through graphs presenting data collected during simulation execution.

Because they focus on individuals, ABSs offer researchers the ability to make specific adjustments to agent behavior so that they can respond adaptively to the environment and other agents, influencing the observed results in the system under study. An example of this is the ABS developed by [Pereira et al. 2011] to study human behavior in natural disaster situations and how to improve the performance of rescue teams.

Additionally, ABSs are highly scalable and modular, which means that models can be easily adapted and extended to include new elements or details [Macal 2016]. This makes this approach especially useful for simulating dynamic and evolving systems, such as ecosystems, financial markets, or social networks.

2.2. GAMA platform

Agent-Based Modeling (ABM) platforms, like Cormas and GAMA, are specialized tools for creating agent-based simulations (ABSs), particularly focusing on explicit spatial representation. Cormas, developed in 1998, stands as one of the pioneering ABM platforms [Bousquet et al. 1998]. Similarly, GAMA, built in Java, is a prominent open-source ABM platform. These platforms aim to offer a scientific approach for modeling a wide range of scenarios, making them accessible to both scientists and non-scientists alike [Taillandier et al. 2019].

GAMA provide a programming language called Gama Modeling Language (GAML). Through GAML, it is possible to develop simple or complex simulations, including 2D and 3D models that provide control over aspects such as lighting, texture, and cameras. Additionally, the platform prioritizes the development of intuitive interfaces, enabling the provision of various customizable displays for the same model, dynamically updated to visualize the simulation [Taillandier et al. 2019].

ABSs for studying various phenomena have been recently developed in GAMA. [Gaudou et al. 2020] present the COMOKIT, an ABS developed in GAMA to analyze and compare interventions to deal with the Covid-19 epidemic at the scale of a city. [Daudé et al. 2019] present the ESCAPE ABS, to study mass evacuation strategies in crisis situations. These simulations highlight the potential of the GAMA platform.

2.3. Sugarscape

The Sugarscape is a model of an artificial society proposed by [Epstein e Axtell 1996]. It is widely used in the ABS community as an example of a simulation that exhibits emergent phenomena. In essence, the Sugarscape model simulates a population that depends on limited resources existing in the environment. The population consists of *ants* searching for food (sugar) present in the environment, which is composed of different regions, some rich and others poor in sugar. Each ant in the simulation is represented by an artificial agent with distinct attributes, such as vision, energy, and metabolism. These agents

have the ability to move in the environment and collect sugar. The objective of each ant is to identify nearby regions with a higher quantity of sugar and that are free (without another ant). When a region is identified, the ant moves there. When an ant interacts with a sugar region, it consumes the resource, and this sugar may or may not have an immediate growth. The variation in resources allows for exploring population dynamics and their concentration in different parts of the environment.

In the model created by [Epstein e Axtell 1996], the environment consists of a set of 2500 cells, representing the regions, organized in a grid of dimensions 50 x 50. Each cell has a certain amount of sugar and a maximum storage capacity. The initial amount of sugar in each cell is defined to form two sugar peaks, in the northeast and southeast of the grid. An external file provides the initial amount of sugar for each cell. As the simulation progresses, the sugar consumed by the agents is replenished. Two replenishment rules are considered: *immediate replenishment*, where the amount of sugar consumed by an agent is fully replenished at each simulation step, and *partial replenishment*, where only a percentage of the initial amount of sugar is replenished at each simulation step.

Although simple, the Sugarscape simulation highlights, for example, the emergent ecological phenomenon of carrying capacity for a species—according to which a given environment can only support a finite population [Epstein e Axtell 1996]. This ABS was chosen in this work due to its simplicity and popularity. The NetLogo platform, for example, provides well-documented implementation of Sugarscape with the two mentioned sugar replenishment rules [Li e Wilensky 2009a, Li e Wilensky 2009b].

3. Development of ABS Sugarscape in GAMA

This section presents the development of the Sugarscape simulation on the GAMA platform. The essential elements for developing ABSs are explained through the Sugarscape simulation, making this section an introductory guide for ABS development in GAMA.

To develop an ABS in GAMA, it is essentially necessary to implement three elements: the global species, the regular species, and an experiment. Figure 1 presents a diagram with these elements, their relationships, and components.

It is worth noting that the diagram in Figure 1 is a contribution of this paper. The GAMA documentation provides a learning tutorial[2] that only describes the general structure of the GAML language metamodel, not offering a concrete view of the elements required in the simulation and their components. GAML is still under development, which may explain its documentation. This can create difficulties for beginner developers to understand the structure of ABSs on the platform. The following will detail the essential elements of ABSs in GAMA and how they were implemented in Sugarscape. Due to space limitations, code snippets are not included in this paper. However, we provide the complete ABS implementation online[3]. We suggest that the reader download the ABS so they can verify the implementation of the elements as they are described.

3.1. Global Species

The global species, represented by the red color in Figure 1, plays a fundamental role in the simulation, being the main entity that defines the essential characteristics of the

[2] https://gama-platform.org/wiki/LearnGAMLStepByStep

[3] https://github.com/agentbasedsimulations/sugarscape-gama-simulation

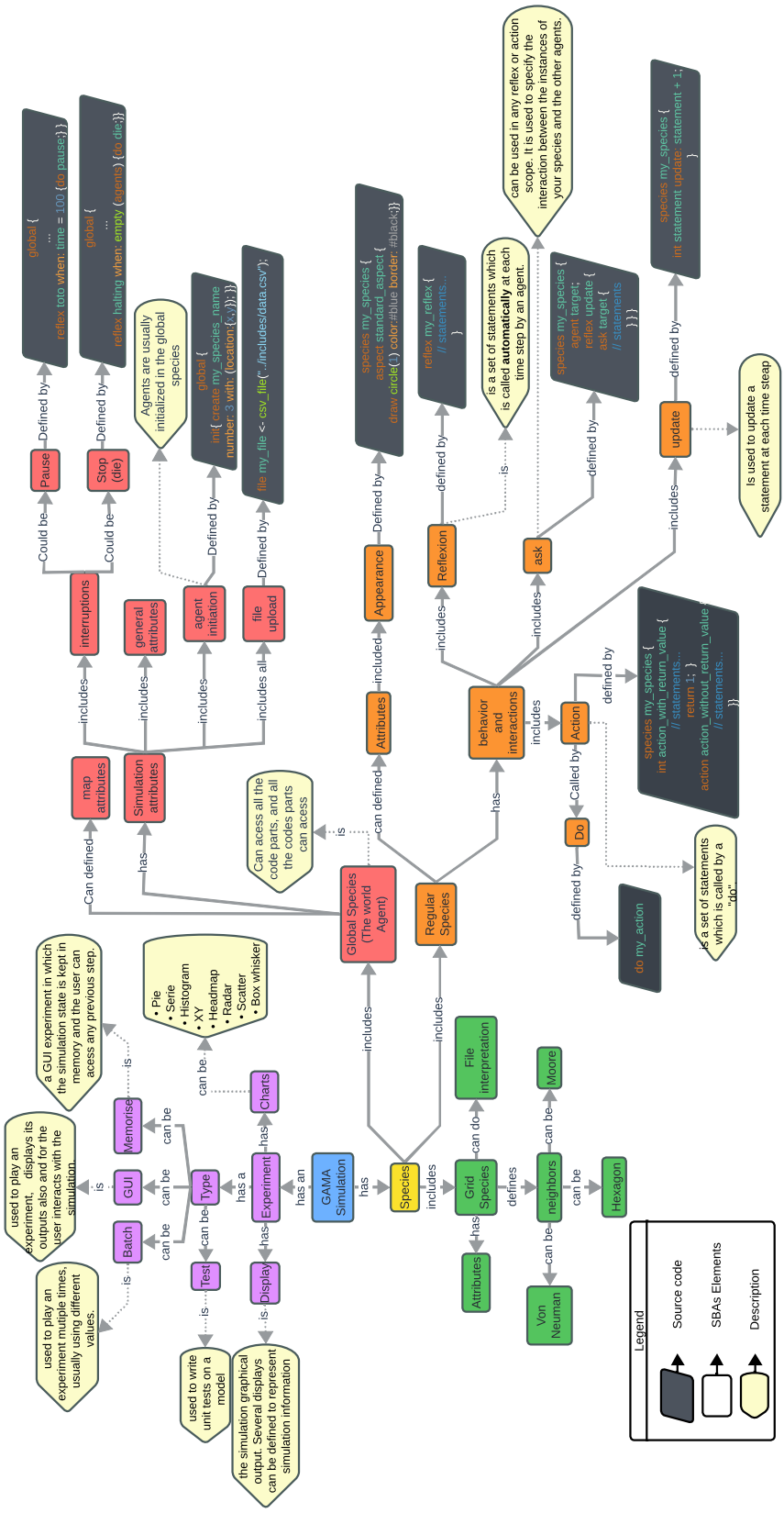


Figure 1. Essential elements of an ABS in GAMA

environment and agents. [Taillandier et al. 2019] define it as a *master species*. Only one instance of the global species can exist, and it is where characteristics such as values and methods accessible to all other species and the experiment are defined.

In addition to user-defined data, the global species incorporates attributes that define vital aspects of the simulation, such as the size of the map, as well as fundamental attributes like simulation interruptions, which can pause or terminate the simulation. These attributes can be accessed and modified by other species and the experiment, providing a flexible basis for configuring the simulation environment [Taillandier et al. 2019].

In Sugarscape, the global species plays an important role because it defines the general parameters of the simulation, such as the minimum/maximum values for some ants attributes. These parameters can be associated with visual components, allowing the user to input their values when running the simulation. Considering the heterogeneity of ants in the simulation, with frequently different attribute values among them (for example each ant may have a different range of vision or metabolic rate), the use of global parameters and their editing through visual components facilitates the execution of the simulation.

It is in the global species that the type of simulated environment should be defined. GAMA supports two types of environments: grid and graph. The grid environment resembles a matrix formed by cells, where it is possible to define specific values for specific regions of the environment. In GAMA, the attributes of each cell can be defined manually or initialized from files. When the simulation depends on external files to initialize the environment, it is in the global species that such files should be loaded. In the case of Sugarscape, we chose to initialize the environment from a CSV file, which provides the initial amount of sugar and the maximum storage capacity for each cell.[4].

The visual aspect of the simulated environment is also defined in the global species, where it is possible to implement which colors will be used for the elements of the environment. In Sugarscape, darker shades of yellow were defined to represent cells with a higher concentration of sugar and lighter ones to represent cells with a low concentration of sugar. This visual representation facilitates the interpretation of simulation results and provides important insights into the behavior of agents in relation to their environment.

Finally, in the global species we can define which data will be collected during the simulation. In Sugarscape, it is necessary to collect data on the average vision and average metabolism of ants, as implemented by [Li e Wilensky 2009a]. The collected data aggregate values from various agents, highlighting the importance of the collection being performed in the global species, which has access to all elements of the simulation.

3.2. Regular Species

The regular species, represented by the orange color in Figure 1, within the GAMA framework, are considered as a complement to the master species, each composed of specific characteristics and values that will serve as fundamental parameters for the creation and manipulation of the necessary instances during the simulation. Analogous to object-oriented languages, we can understand regular species as classes that define a set

[4]The CSV file used was adapted from [Li e Wilensky 2009a]

of properties and behaviors to be shared by their agents [Taillandier et al. 2019]. This organization is essential in GAMA to define and structure the entities that compose the simulation, providing them with identity and functionality.

When creating a species in GAMA, some attributes are automatically integrated, such as the name (by default, the same name by which the species was defined), location and shape. Attributes related to the context of the ABS under development must be explicitly defined, such as actions and behavioral characteristics of each species. Actions, in relation to object orientation, can be compared to methods, and they are what enable the interaction and behavior of the agents.

The instantiation of species (agents) for the simulation execution, as recommended by the language documentation, should be implemented in the `init` block of the global species [Taillandier et al. 2019]. This block is responsible for determining the quantity of agents to be created and, optionally, their initial location. To implement interaction between agents, GAMA provides the `ask` command. This command allows an agent to interact with instances of its own species or other species. For example, the `ask` command can be used to check which agent is the closest in a specific situation.

As a platform aimed at facilitating the creation of ABSs, GAMA offers additional resources that are not found in traditional object-oriented languages. This includes the availability of ready-to-use abilities, such as the movement ability, which provides predefined commands for agents to move and can be adjusted to change the speed and direction of agents as necessary. This flexibility allows for greater adaptation and customization of simulations according to the specific needs of each scenario.

In Sugarscape, the agents are ants. Therefore, it is necessary to define a species `ant` to instantiate the agents in the simulation execution. In the Sugarscape model, the ant species has three essential attributes: *vision*, which determines the number of accessible regions for sugar checking; *metabolism*, which symbolizes the energy expended to keep the ant alive; and initial energy, which defines the amount of energy with which the ant is born [Epstein e Axtell 1996]. In the ABS developed in this paper, minimum and maximum limits for initializing these attributes were specified. These limits are implemented in the global species. Upon creation, a random value is chosen between these limits for each characteristic of the ant using the `rnd()` command in GAMA.

In the `ant` species, the actions that the agents must perform during the simulation are also implemented. In GAMA, these actions can be implemented in the following ways: through the `Reflexion` command, which calls a method automatically at each cycle; through the `action` command, a method that can be activated from a `do` command; through the `ask` command, which refers to an interaction between agents; and finally, the `update` command, which updates an attribute at each cycle.

Another important method, is the movement in the environment. For this to occur, the map must be imagined as a matrix with several squares, each with a values. The ant must be able to check in each of these squares both the availability of sugar and the spatial availability, using the `empty` command.

Figure 2 presents a portion of the *Sugarscape* simulation implemented in the GAMA platform. In the left section, we find the *Users Models*, which allows developers to create customized ABSs, while a green button initiates the simulation. In the right

section, there is the code where agent attributes can be configured, such as the random values for vision and metabolism of each agent, illustrating the use of the `<-` operator for value assignment. Additionally, the use of commands like `update` and `reflex` is evident, as mentioned earlier. Furthermore, there is the definition of agent appearance and the termination of their life.

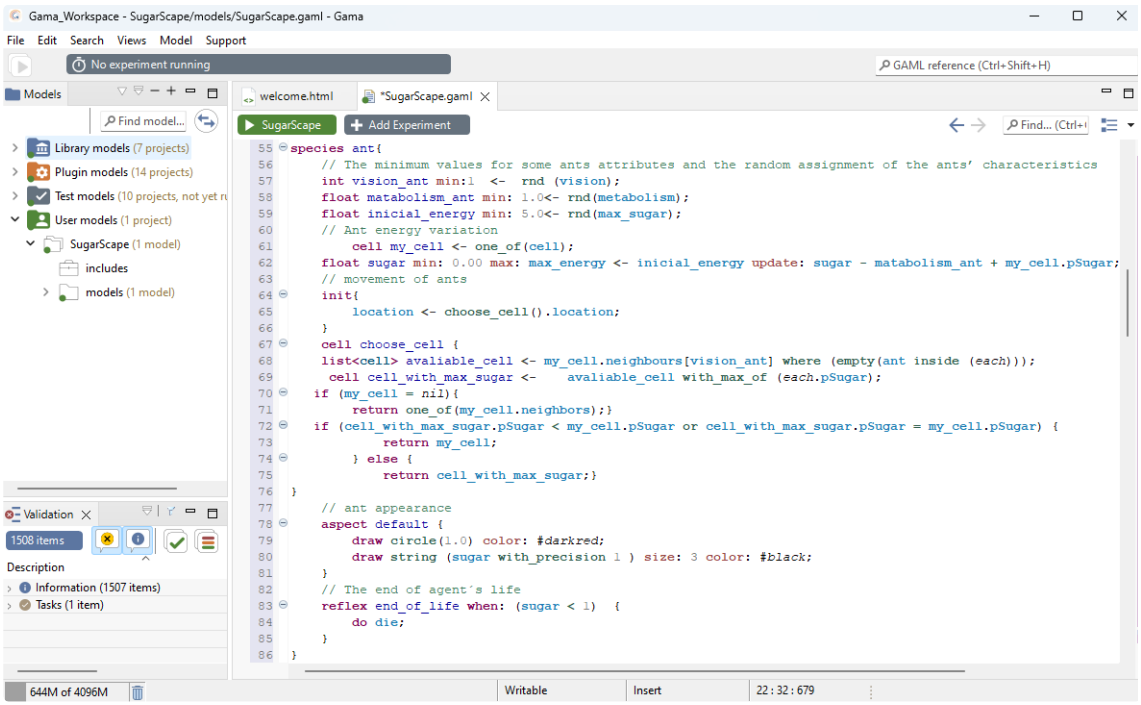


Figure 2. The SugarScape code on the GAMA platform

3.3. Grid Species

The *Grid* species, represented by the green color in Figure 1, is unique compared to other species, as it has unique attributes and behaviors that are essential for defining the environment where the simulation takes place. Unlike conventional species, it is automatically generated and is present in all simulations. The *Grid* is responsible for defining the characteristics of the map. In general, it is a structure that represents a grid or spatial mesh on which agents can move and interact in a simulation. This grid is composed of cells, where each cell can contain different attributes, such as the amount of available resources, the presence of obstacles, or other relevant information to the ABS.

By default, the *Grid* species has a series of attributes, such as the number of rows and columns, as well as the predefined color. However, all these characteristics can be modified if we define the species in the code. When manipulating the *Grid*, we are actually altering a matrix that, by default, has dimensions of 100x100. In other words, all changes will permeate every value of the matrix. Additionally, when defining the *Grid* species, it is necessary to specify which neighbor rule will be adopted for the cells. GAMA supports the following neighbor rules: *Von Neumann*, where each cell perceives 4 neighbor cells (horizontally and vertically); *Hexagon*, perceiving 6 neighbor cells; and *Moore*, perceiving 8 neighbor cells.

In Sugarscape, the *grid* species is defined with dimensions of 50x50. Each cell has attributes representing the current amount of sugar and the maximum amount of sugar it can hold. Depending on the version of the implemented simulation, the sugar growth rate can be added as an attribute, which varies based on both consumption and progressive recovery. This feature is implemented using the `update` command, providing greater dynamism to the simulation. The maximum sugar amount is initialized from the CSV file loaded in the global species. A value of zero indicates the absence of sugar in the cell. The adopted neighborhood rule is the *Von Neumann*, where neighboring cells are considered by the agent when searching for cells with sugar to move to.

3.4. Experiment

For an ABS to be effective and representative, it's desirable that it's visual and encompasses all the characteristics defined in the various species of the simulation. This requires a precise definition of the experiment, represented by the purple color in Figure 1, where the user has the ability to configure inputs, outputs, and behaviors as needed for the simulation. In GAMA, there are four main types of experiment that can be implemented to meet different needs and contexts [Taillandier et al. 2019].

- *GUI*: It offers a graphical user interface that allows interaction with the simulation. This facilitates parameter manipulation and real-time observation of results, providing a more intuitive and immersive experience.
- *Batch*: It allows running the simulation multiple times, with the possibility of pre-changing parameter values for each run. This approach is useful for comparative analyses and evaluating results in different scenarios, providing insights into the model's behavior under various conditions.
- *Test*: It focuses on performing utility tests to ensure the quality and integrity of the simulation. This approach is essential for validating the accuracy and robustness of the model, identifying any potential flaws or inconsistencies that could compromise the results.
- *Memorize*: It maintains a detailed record of each step taken during the simulation. This allows the user to review and modify any stage of the process as needed, providing greater control and flexibility in conducting the simulation.

When defining the experiment, it's necessary to specify the type of simulation adopted. Additionally, the visual and interactive configurations of the simulation are also defined. This includes determining how relevant information will be presented through graphs and monitors, which play a role in interpreting the data and making informed decisions based on the results obtained.

In Sugarscape, the type of simulation adopted is the GUI. Through the graphical interface generated by GAMA, it is possible to explore and analyze the behavior of the ants in the environment. This choice allows for direct interaction with the simulation through display elements. The displays, which are graphical representations of some elements of the simulation, facilitate understanding and experimentation with different configurations to investigate various aspects of the phenomenon under study.

In the experimental species of Sugarscape, four display elements are defined. The first display, is called *display grid*, is a visual representation of the simulation's grid, showing the cells and agents. The other three displays specify graph elements that

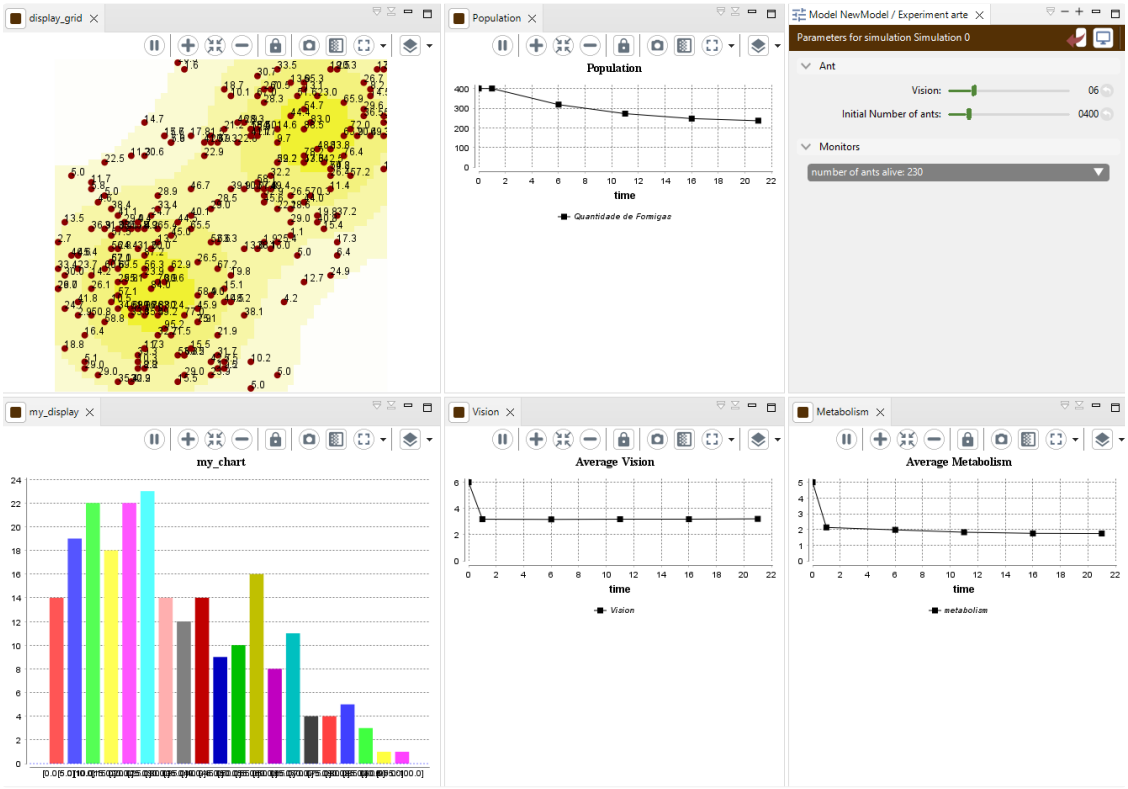


Figure 3. SugarScape simulation made on GAMA

display aggregated data from the various agents collected through the definition in the global species: *population* is a line graph that shows the number of ants in the simulation over time; *vision* is a line graph showing the average vision of the ants over time; and metabolism is also a line graph, displaying the average metabolism of the ants.

Besides display elements, the experimental species also allows for the definition of monitoring elements. These elements show values throughout the execution of the simulation, providing the user with a quick and convenient way to monitor the simulation's progress and make decisions based on the presented data. In Sugarscape, a monitor is defined to display the number of ants in the simulation.

Figure 3 show us a visual representation of the ongoing simulation is presented, comprising four distinct monitors. The first monitor, labeled *Display grid*, graphically displays the distribution of ants on the map grid, accompanied by a histogram that assesses the energy distribution among live ants. Subsequent monitors, *Population*, *Average Metabolism*, and *Average Vision*, provide the variation in the population of live agents, average metabolism, and average vision of ants over time. Additionally, the figure includes interactive parameters allowing users to adjust the agents' vision range and modify the initial quantity of ants, alongside a real-time monitor showing the quantity of live ants in the simulation, providing an instant view of the current population status.

4. Recommendations for Beginners in GAMA

Developing on the GAMA platform can present a significant challenge, even for those familiar with ABMs. This is partly because the platform is still under development and may

exhibit errors in executing its commands. Additionally, the available documentation can be confusing for beginners, lacking in information and organization, which contributes to an inadequate understanding for the developer.

During the development process of Sugarscape, it was necessary to find workarounds for some issues. One such issue involved the `neighbours` command, which was supposed to return the neighboring cells to the agent's current position. However, this command did not worked as expected, ignoring the predefined configuration specified for the grid. This issue was reported to the GAMA developers[5] through the platform's mailing list, and they acknowledged the problem with the command and suggested an alternative method to achieve the desired execution. Another problem reported to the developers[6] but still unresolved is the use of a histogram graph, which cannot be configured to adapt to the constantly varying energy levels of the agents.

Given this, a recommendation for beginners on the GAMA platform is to interact and post questions on the mailing list. However, it is important to note that responses from the GAMA developers often take a significant amount of time, and the answers may be insufficient for problem resolution. For example, the query regarding the histogram took three months to receive an inconclusive response about its application.

The scarcity of tutorials and explanatory articles about the functionality and development within the platform hinders the initial access for people interested in using GAMA for their simulations. Therefore, it is recommended to organize your questions and seek out forum discussions with other users of the platform. Often, these users are helpful and share their own experiences, which facilitates understanding and reduces the learning curve.

5. Conclusion

This work presented the elements of the GAMA platform that are essential for developing an ABS. To this end, the development of Sugarscape, a simple and widely known ABS, was reported. A diagram was built to highlight the essential structure of an ABM in GAMA, so that beginners on this platform can use this paper as an introductory guide to develop simulations.

Indeed, while the GAMA platform facilitates the development of ABMs, especially when compared to object-oriented languages, it does require dedication and study from the developer to understand its elements and perform basic operations. The absence of materials that assist the beginner developer in GAMA contributes to the difficulty of accessibility of the platform for those developing their first ABM. Therefore, works like this are relevant to contribute to the popularization, accessibility, and development of GAMA.

References

Bousquet, F. c., Bakam, I., Proton, H., e Le Page, C. (1998). Cormas: Common-pool resources and multi-agent systems. In Pasqual del Pobil, A., Mira, J., e Ali, M., editors, *Tasks and Methods in Applied Artificial Intelligence*, pages 826–837, Berlin. Springer.

[5] `https://groups.google.com/g/gama-platform/c/jOFr9ju7sS4/m/LekjUGQnAAAJ`

[6] `https://groups.google.com/g/gama-platform/c/fSDZzeQjs_E/m/wSdQZUVHAgAJ`

Costa, A. C. R. (2023). The tupinambá: An exercise in societal modeling. In *Anais do XVII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações (WESAAC 2023)*, pages 44–54.

Daudé, E., Chapuis, K., Taillandier, P., Tranouez, P., Caron, C., Drogoul, A., Gaudou, B., Rey-Coyrehourcq, S., Saval, A., e Zucker, J.-D. (2019). ESCAPE: exploring by simulation cities awareness on population evacuation. In *Proceedings of the 16th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2019)*, pages 76–93.

Epstein, J. M. e Axtell, R. L. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. MIT Press.

Gaudou, B., Huynh, N. Q., Philippon, D., Brugière, A., Chapuis, K., Taillandier, P., Larmande, P., e Drogoul, A. (2020). COMOKIT: A modeling kit to understand, analyze, and compare the impacts of mitigation policies against the COVID-19 epidemic at the scale of a city. *Frontiers in Public Health*, 8.

Kleiboer, M. (1997). Simulation methodology for crisis management support. *Journal of Contingencies and Crisis Management*, 5(4):198–206.

Klügl, F. e Bazzan, A. L. C. (2012). Agent-based modeling and simulation. *AI Magazine*, 33(3):29–40.

Li, J. e Wilensky, U. (2009a). Netlogo sugarscape 1 immediate growback model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Li, J. e Wilensky, U. (2009b). Netlogo sugarscape 2 constant growback model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10(2):144–156.

Pereira, A. H., Nardin, L. G., e Sichman, J. S. a. (2011). Coordination of agents in the robocup rescue: A partial global approach. In *2011 Workshop and School of Agent Systems, their Environment and Applications*, pages 45–50.

Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.-N., Marilleau, N., Caillou, P., Philippon, D., e Drogoul, A. (2019). Building, composing and experimenting complex spatial models with the GAMA platform. *GeoInformatica*, 23:299–322.

Teixeira, L. e Santos, F. (2020). Uma simulação com agentes para estudar a propagação da COVID-19 em ibirama(SC). In *Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional*, pages 176–187, Porto Alegre, RS, Brasil. SBC.

Wilensky, U. (1999). NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky, U. e Rand, W. (2015). *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. Mit Press.

Agentes BDI e Aprendizagem: um mapeamento sistemático e utilização com a biblioteca MASPY

Felipe Merenda Izidorio[1], **Alexandre L. L. Mellado**[1], **André Pinz Borges**[1], **Gleifer Vaz Alves**[1]

[1]Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
Ponta Grossa – PR – Brasil

{felipemerenda, mellado}alunos.utfpr.edu.br

{apborges, gleifer}@utfpr.edu.br

Abstract. *Reinforcement Learning algorithms can solve sequential decision processes through repeated interactions with an environment. This approach solves complex challenges and enables technological innovations such as Autonomous Vehicles (AVs). With this in mind, this paper presents the planning, execution and conclusions of a systematic mapping of the literature on learning algorithms for AVs. One gap identified is integrating BDI Intelligent Agent architecture with Reinforcement Learning. To address this, an example is presented using the MASPY library in Python, in which a BDI agent is programmed to use learning components.*

Resumo. *Os algoritmos de Aprendizagem por Reforço são capazes de resolver processos de decisão sequenciais por meio de interações repetidas com um ambiente. Essa abordagem permite a solução de desafios complexos e possibilita inovações tecnológicas, como os Veículos Autônomos (VAs). Com isso em mente, este artigo apresenta o planejamento, execução e conclusões de um mapeamento sistemático da literatura sobre algoritmos de aprendizagem para VAs. Uma lacuna identificada é a integração de arquitetura de Agentes Inteligentes BDI com Aprendizagem por Reforço. Para abordar isso, é apresentado um exemplo usando a biblioteca MASPY em Python, em que é programado um agente BDI que utiliza componentes de aprendizagem.*

1. Introdução

Agentes inteligentes são amplamente utilizados em tomada de decisão envolvendo sistemas autônomos, em sistemas de recomendação, navegação, locomoção autônoma ou definição do fluxo de dados de rede [Gronauer and Diepold 2022]. Esse conceito é utilizado na área de aprendizado por reforço como o componente que aprende sobre um determinado ambiente. Em específico, uma das maiores vantagens deste aprendizado é sua adaptabilidade a diferentes áreas de atuação [Shakya et al. 2023].

Baseados no Modelo de Decisão de Markov (MDP), que estabelece que os estados futuros e recompensas são independentes dos estados e ações passadas, e nas Equações

de Bellman [Bellman 1957] usadas para resolver problemas de decisões sequenciais, os algoritmos de Aprendizagem por Reforço (AR) são capazes de aprimorar as tomadas de decisões por meio de interações repetidas com o ambiente [Sutton and Barto 2018]. Estes algoritmos podem solucionar desafios complexos e possibilitar várias inovações tecnológicas, como os Veículos Autônomos (VA), onde um veículo com AR é capaz de desenvolver uma política de ações ótima com base no estado atual.

A execução de um mapeamento sistemático da literatura sobre algoritmos de aprendizagem para a condução de VAs visa analisar de maneira abrangente as pesquisas existentes, identificar tendências e desafios. Este processo inclui a geração das questões de pesquisa, definição dos critérios de inclusão e exclusão, a realização de buscas em bases de dados acadêmicas, e a aplicação de métodos padronizados para avaliar a qualidade e relevância dos estudos selecionados.

Após a realização do mapeamento sistemático foi possível identificar uma lacuna (que conforme o nosso conhecimento) não foi amplamente explorada na área, a qual é a utilização da arquitetura de Agentes Inteligentes juntamente com Aprendizagem por Reforço, tendo como referência o modelo BDI [Bratman 1987], baseado em estados mentais *belief-desire-intention* para agentes deliberativos. Esse tipo de abordagem possui a sua capacidade de tomar decisões complexas, ao considerar múltiplos objetivos e ações em ambientes dinâmicos e complexos, e suas ações são executadas com base em suas crenças, desejos e intenções, promovendo mais transparência e previsibilidade.

Visando combinar um sistema multiagente BDI com aprendizado por reforço, foi desenvolvida a biblioteca MASPY, primeiramente apresentada sem aprendizagem em [Mellado et al. 2023]. Esta biblioteca, em Python, possui as abstrações necessárias para a programação de múltiplos agentes BDI com acesso a uma classe de aprendizagem para expandir suas capacidades. Este trabalho apresenta um exemplo de aprendizado sobre a movimentação em um ambiente 2D, como o agente BDI interage com a política encontrada e como as questões analisadas no mapeamento refletem no futuro da biblioteca. MASPY apresenta uma proposta semelhante à encontrada em [Bosello 2020], porém enquanto este estende o aprendizado em Jason com foco em veículos autônomos, é esperado que MASPY possibilite um desenvolvimento genérico de agentes.

O trabalho está organizado da seguinte maneira: a seção 2 apresenta o mapeamento sistemático; a seção 3 descreve a biblioteca MASPY; a seção 4 mostra um exemplo agente BDI com aprendizagem; e a seção 5 apresenta as considerações finais.

2. Mapeamento Sistemático da Literatura

Com o propósito de identificar algoritmos de aprendizagem na condução de VAs e encontrar possíveis lacunas dentro da área, foi realizado um mapeamento sistemático da literatura. A Seção 2.1 apresenta a metodologia utilizada e as etapas realizadas, a Seção 2.2 aborda sobre o planejamento, a Seção 2.3 fala sobre a execução, a Seção 2.4 faz uma análise dos resultados e, por fim, a Seção 2.5 apresenta algumas conclusões.

2.1. Metodologia de Pesquisa

Para a execução do mapeamento sistemático da literatura foi utilizada a metodologia de Kitchenham e Chartes [Kitchenham and Charters 2007], com o objetivo de identificar e classificar a pesquisa relacionada a um tópico amplo de pesquisa. Há três etapas a serem

executadas: o planejamento, que contêm a descrição do problema e uma especificação das questões de pesquisa, a execução, no intuito de identificar as pesquisas mais relevantes realizando uma extração/síntese dos estudos e uma etapa de avaliação de qualidade dos estudos, e uma análise dos resultados encontrados, respondendo às questões de pesquisa elaboradas na etapa de planejamento. Para auxiliar a realização desse mapeamento, foi utilizado a ferramenta Parsifal [Freitas and Segatto 2021].

2.2. Planejamento

O objetivo desse mapeamento é buscar por estudos relacionados a algoritmos de aprendizagem por reforço com foco na área de condução de VAs considerando 5 questões:

QP1. Quais algoritmos de aprendizagem são usados no trabalho?
QP2. É realizada uma comparação com outros algoritmos de aprendizagem?
QP3. Quais métricas de avaliação são utilizadas no trabalho?
QP4. Qual a técnica predominante nos trabalhos de condução de VAs?
QP5. Dentro da área de condução de VAs, qual temática é a mais abordada?

A partir dessas questões é possível analisar os resultados encontrados. Antes de realizar a busca dos trabalhos relacionados, é necessário definir algumas palavras-chave, na língua inglesa, relacionadas ao objetivo do mapeamento, no caso: *Reinforcement Learning* e *Autonomous Vehicle Navigation*, juntamente com os seus sinônimos, e combiná-las com os operadores lógicos AND e OR para formar uma *string* de busca. Dessa forma, pode-se executar uma busca assertiva nos repositórios de pesquisa. A *string* resultante foi: *("Reinforcement Learning"OR "Decision Making"OR "Deep Reinforcement Learning"OR "Reinforcement Learning Algorithms") AND ("Autonomous Vehicle Navigation"OR "Autonomous Vehicle Driving"OR "Autonomous Vehicle Movement Control"OR "Self-Driving Car Driving"OR "Self-Driving Car Movement Control").*

Para a realização da busca dos estudos foram usadas três bases de busca: IEEE Digital Library, Science Direct e Scopus. O motivo da escolha dessas bases de dados se deve ao fato de possuírem um grande número de artigos científicos, e por apresentar uma ferramenta de busca eficiente capaz de selecionar os estudos mais relacionados ao trabalho por meio de uma *string* de busca e filtros, como: ano de publicação, área de pesquisa, idioma, etc. Em seguida, são definidos os critérios de inclusão (CI) e exclusão (CE):

CI1. Aborda sobre algoritmos de aprendizagem (***CI2.*** ...para agentes) (***CI3.*** ...para a condução de VAs)
CE1. Estudos duplicados / ***CE2.*** Fora de escopo / ***CE3.*** Não é estudo primário

A partir desses critérios será possível eleger os trabalhos que serão usados para a etapa de qualidade. Assim, foram definidas as questões de qualidade:

QQ1. O trabalho aborda sobre algoritmos de aprendizagem por reforço?
QQ2. O trabalho faz uma comparação com outros algoritmos?
QQ3. O trabalho se baseia em alguma arquitetura de agentes?
QQ4. O trabalho aborda sobre condução de VAs?
QQ5. O trabalho utiliza o(os) algoritmo(os) para conduzir os VAs?
QQ6. O trabalho faz uma avaliação do(os) algoritmo(os) abordado(os)?

Para cada uma das questões, há três respostas possíveis: Sim, Parcialmente ou Não, com a, respectiva, pontuação: 1, 0.5 ou 0. A soma dessas pontuações corresponde ao quão útil o trabalho é para o tema abordado.

2.3. Execução

Com a etapa de Planejamento realizada, inicia-se a etapa de Execução. Por meio das bases de dados citadas e a *string* de busca, foram obtidos 159 trabalhos sobre o tema. Na sequência, foram aplicados os critérios de inclusão e exclusão definidos anteriormente. Sendo que, com a aplicação do critério CE1, 37 trabalhos foram eliminados, resultando em 122 trabalhos. Com a aplicação do CE2 foram retirados 81 trabalhos. E por fim, com o critério CE3, 7 trabalhos foram descartados. Dessa forma, foram selecionados 34 trabalhos para leitura e realização da etapa de avaliação de qualidade.

Com os trabalhos selecionados foi realizada uma leitura de cada um deles para responder às questões de qualidade e mensurar as respectivas pontuações de qualidade (*Score*), explicado anteriormente na etapa de planejamento. Com o intuito de realizar uma investigação assertiva desses estudos obtidos, somente aqueles que obtiveram uma pontuação acima de 3 pontos foram escolhidos para servirem como base para a análise dos resultados. Esses trabalhos são mostrados na Tabela 1

ID	Título	*Score*	ID	Título	*Score*
1	[Liang et al. 2023]	5.0	12	[Sun et al. 2023]	5.5
2	[Dhinakaran et al. 2024]	5.0	13	[Hook et al. 2021]	4.0
3	[Zhu and Hayashibe 2023]	5.0	14	[Hartmann et al. 2020]	5.0
4	[Arvind and Senthilnath 2019]	5.0	15	[Jacinto et al. 2023]	4.0
5	[Dar et al.]	5.0	16	[González-Miranda et al. 2023]	4.0
6	[Tiong et al. 2023]	4.0	17	[Mackay et al. 2022]	4.0
7	[Zhai et al. 2021]	4.0	18	[Yang et al. 2023]	4.0
8	[Josef and Degani 2020]	4.0	19	[Lambert et al. 2021]	4.0
9	[Bin Issa et al. 2021]	4.5	20	[Taghavifar et al. 2024]	4.0
10	[Zhang et al. 2019]	5.0	21	[Qin et al. 2024]	5.0
11	[Gao et al. 2023]	5.0	22	[Cabezas-Olivenza et al. 2023]	4.5

Tabela 1. Avaliação de qualidade dos trabalhos selecionados

2.4. Análise dos Resultados

A seguir serão respondidas às questões de pesquisa a respeito dos trabalhos da Tabela 1.

QP1. Quais algoritmos de aprendizagem são usados no trabalho? Vários algoritmos de Aprendizado por Reforço foram citados, incluindo Q-Learning, Sarsa, Deep Q-Networks e Proximal Policy Optimization. O Deep Q-Networks se destacou, combinando as vantagens do Q-Learning com o poder das Redes Neurais. Além disso, alguns trabalhos desenvolveram variações desses algoritmos mencionados.

QP2. É realizada uma comparação com outros algoritmos de aprendizagem? Dos trabalhos analisados, 15 realizaram comparações, utilizando as métricas de avaliação para determinar quais algoritmos foram mais eficazes.

QP3. Quais métricas de avaliação são utilizadas no trabalho? Os estudos adotaram diferentes métricas de avaliação de acordo com o problema escolhido. Por exemplo, em prevenção de colisões, os autores investigaram o tempo em que um veículo se moveu antes de colidir com um obstáculo. Ao longo dos testes, o veículo conseguiu aumentar esse tempo. No entanto, todos os trabalhos utilizaram um gráfico de recompensa média por episódio para mostrar a eficácia do treinamento do veículo.

QP4. Qual a técnica predominante nos trabalhos de condução de VAs? Todos os estudos abordam Aprendizado por Reforço, porém, algumas técnicas foram combinadas com outras áreas para desenvolver frameworks ou métodos para resolver problemas específicos. Por exemplo, o artigo 9 usou Redes Neurais Convolucionais para processar imagens. O artigo 21 utilizou *Social Value Orientation*, um conceito da psicologia que descreve como as pessoas avaliam e valorizam relações sociais e tomam decisões em contextos sociais, em conjunto com Aprendizado por Reforço. No total, 17 artigos incorporaram Redes Neurais em seus algoritmos, destacando a eficácia dessa técnica na condução de VAs.

QP5. Dentro da área de condução de VAs, qual temática é a mais abordada? Dentro da vasta área de condução de VAs, os estudos focaram em temas específicos, como controle de velocidade (trabalho 14), manutenção de faixa (trabalho 16), controle de estabilidade (trabalhos 1 e 22), condução em ambientes acidentados (trabalho 8), prevenção de colisão com obstáculos estáticos e dinâmicos (trabalhos 1, 4, 16 e 20), e navegação em cenários multiagentes (trabalhos 7, 12 e 13), entre outros.

2.5. Conclusões do Mapeamento

As respostas das questões levantam considerações importantes sobre os algoritmos de aprendizagem para VAs. O Deep Q-Networks, amplamente citado, é uma opção versátil para abordar os desafios de envolvendo VAs. A comparação com outros algoritmos é crucial para avaliar sua eficácia e serve como guia para estudos futuros. Métricas específicas relacionadas ao problema são mais vantajosas para solucioná-lo. A utilização de ferramentas de outras áreas pode enriquecer o aprendizado dos agentes. A área de condução de VAs tem diversas temáticas a serem exploradas, incluindo aquelas menos abordadas, que podem ser relevantes para estudos futuros.

Vale ressaltar que nenhum dos trabalhos se baseia em alguma arquitetura de agentes (QQ2). Isso é uma lacuna presente dentro do escopo de algoritmos de aprendizagem e um estudo aprofundado em algumas arquiteturas, como o modelo BDI, é uma abordagem pertinente. Durante o processo de revisão foi descoberto o trabalho de [Bosello 2020], que apresenta uma extensão da linguagem de agentes Jason com técnicas de aprendizagem de máquina. Esse trabalho apresenta, assim como a MASPY, agentes BDI com aprendizado por reforço. Porém, enquanto este tem foco no desenvolvimento de navegação autônoma, a biblioteca MASPY almeja viabilizar a criação de sistemas genéricos para agentes, como descrito a seguir.

3. MASPY: Aprendizagem e Sistema BDI

A biblioteca MASPY é dividida em cinco classes para o desenvolvimento de sistemas multiagentes, como mostra a figura 1. A classe de agentes possui as funções para as lógicas de BDI, utilizando os componentes de crenças, objetivo e planos. A classe de ambiente fornece uma abstração onde agentes podem perceber e realizar ações sobre um contexto de forma simulada. A classe de comunicação permite que agentes troquem mensagens um com os outros. A classe de aprendizagem expande as capacidades do agente além do conjunto de regras do paradigma BDI. E a classe de administrador ajudar o programador na configuração e execução do sistema além de assegurar suas funcionalidades.

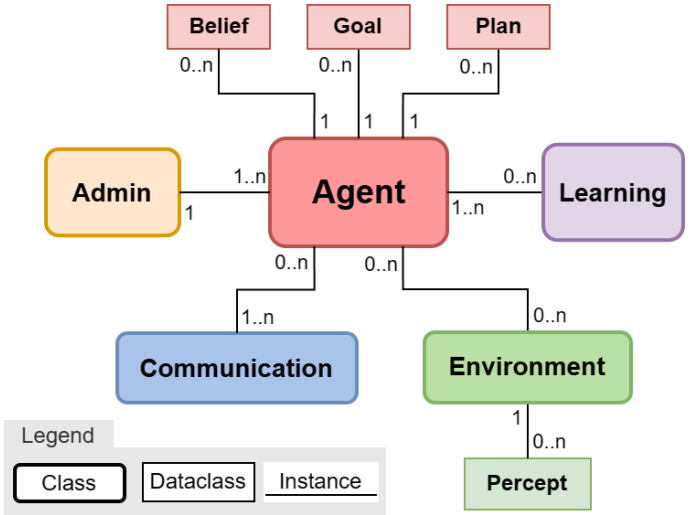


Figura 1. Diagrama das classes de MASPY

3.1. Classe de Agente

A classe de agente é a base da biblioteca MASPY. Ela compõe as abstrações para o gerenciamento de crenças e objetivos, assim como o ciclo de raciocínio que promove o comportamento BDI do agente. Durante a execução de um agente, este adquiri e perde crenças e objetivos que influenciam sua escolha de planos. Estas mudanças causam eventos que são o gatilho para que o agente realize ações.

A figura 2 apresenta como o raciocínio de cada agente transcorre durante a execução de um sistema multiagente usando MASPY. O começo do ciclo envolve a atualização de crenças. Para isso são consideradas percepções de ambiente e mensagens de outros agentes. Em seguida, os objetivos são atualizados. Neste começo do raciocínio, somente mensagens afetam os objetivos.

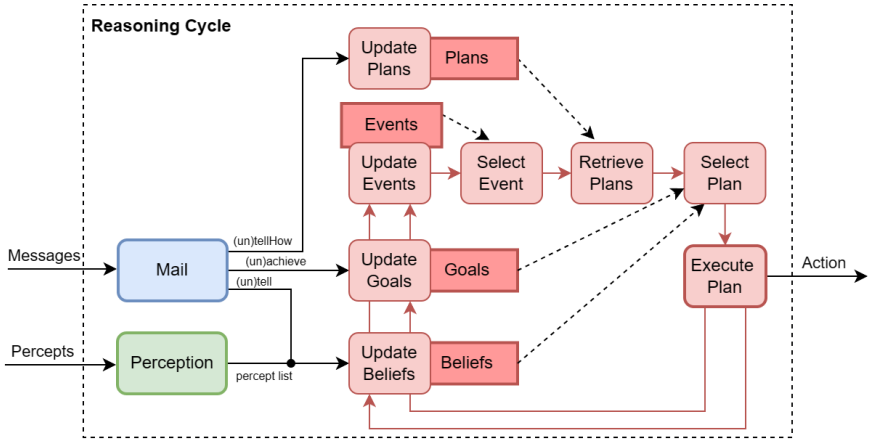


Figura 2. Ciclo de Raciocínio do Agente

As alterações feitas, seja ganho ou perda de informação, são adicionadas em uma lista de eventos. O evento mais antigo é então utilizado para recuperar planos que contém este evento como gatilho. Entre os planos encontrados, o primeiro em que seu contexto é

refletido com a lista de crenças e objetivos disponíveis é escolhido e executado. Durante a execução deste plano, ações no ambiente podem ser realizadas além de atualização de crenças e objetivos. Concluído o plano, o ciclo recomeça. Fora do ciclo principal, planos podem ser atualizados por mensagens especificas de outros agentes.

3.2. Classe de Aprendizagem

O uso da classe de aprendizagem da biblioteca MASPY é representado pela figura 3.

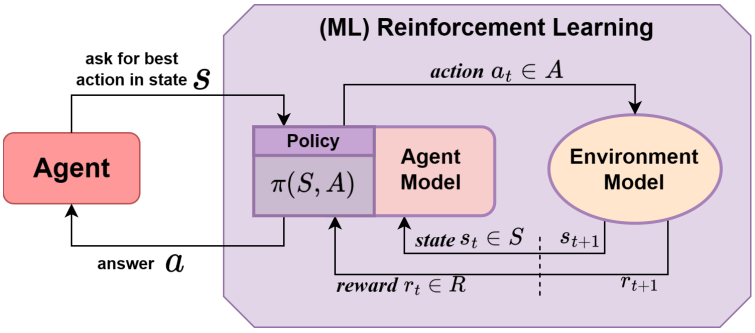


Figura 3. Diagrama de Agente utilizando um Aprendizado por Reforço

A figura 3 mostra um diagrama de como um agente pode utilizar técnicas de aprendizado por reforço. Esse agente pode requisitar o treinamento de um ambiente modelado e pedir pela melhor ação a ser tomada naquela política encontrada. Este treinamento é representado na figura pela interação entre o modelo de agente que realiza ações no ambiente, o qual retorna recompensas que atualiza a politíca do agente.

4. Exemplo de Utilização da Biblioteca

Como exemplo da utilização da MASPY com aprendizagem por reforço, será descrito um agente limpador que aprende a navegar em um ambiente 2D para alcançar todas as posições de sujeiras. O código completo está disponível em: *garbage_cleaner_learn.py*. A figura 4 apresenta um diagrama de sequência da execução e interação entre instâncias de Lrn (Aprendizado), Admin (Administrador), Agt (Agente) e Env (Ambiente).

Configuração: Lrn define seus parâmetros para o aprendizado e executa aprendizado com base nos parâmetros definidos; Lrn configura seu ambiente com o mapa da classe de ambiente; Room (Env) adiciona um *percept* para mostrar que o cômodo está sujo no momento e o Admin define seu nome como “Room”; Room adiciona um *percept* com as posições de sujeira, o mesmo que o mapa enviado para Lrn; Admin define o nome de Rbt (Agt) como “(Rbt,1)”; Rbt se conecta ao ambiente Room, define sua posição inicial e adiciona objetivo de decidir seu próximo movimento.

Execução: Admin inicia o sistema, executando a função de raciocínio de cada agente conhecido; Rbt inicia seu ciclo de raciocínio, percebendo Room; Rbt executa um plano com base em seu objetivo para decidir o movimento; Rbt solicita uma ação de Lrn que retorna a melhor ação aprendida; Rbt adiciona objetivo de mover com base na ação recebida; Essa sequência é repetida até que uma posição suja seja alcançada; Rbt percebe que está na posição de sujeira e a limpa; Room atualiza sua percepção das posições de

sujeiras; Rbt adiciona um objetivo para decidir novamente seu próximo movimento; O agente continua até que todas as posições de sujeira sejam limpas.

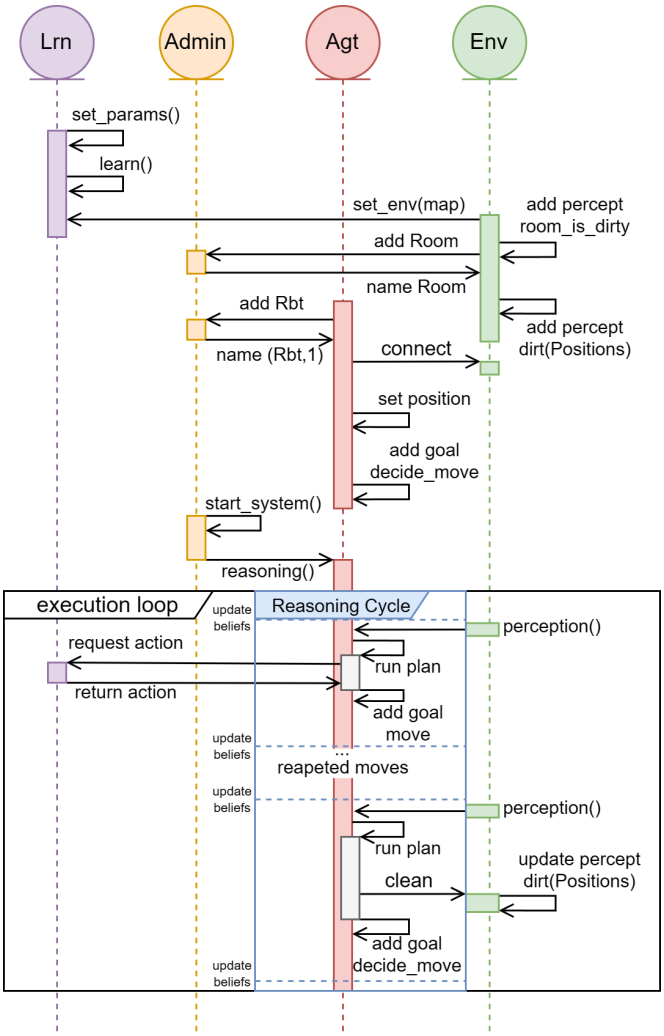


Figura 4. Diagrama de sequência da execução do exemplo

O algoritmo 1 apresenta o plano em que o agente Rbt escolhe entre mover para uma posição suja ou perceber que a limpeza acabou e deve voltar ao carregador. Para identificar que ainda existe posição suja, Rbt verifica, na linha 3, se contém uma crença "*room_is_dirty*" percebida no ambiente Room. Nas linhas 4 e 5, o agente Rbt utiliza o modelo de aprendizado pronto Lrn para requisitar a próxima ação em sua posição.

```
1     @pl(gain,Goal("decide_move"))
2     def decide_move(self,src):
3         if self.has(Belief("room_is_dirty",source="Room")):
4             state = Lrn.env.encode(*self.pos)
5             action,n_state,reward = Lrn.exec(Lrn.env, state)
6             self.add(Goal("move",(action,reward)))
7         else:
8             self.print(f"All dirt is cleaned")
9             self.add(Goal("recharge"))
```

Algoritmo 1. Plano decidir movimento

Com o algoritmo 2, Rbt executa um movimento baseado na ação e recompensa recebida de Lrn, indicado nas linhas 1 e 2. Neste exemplo, quando a recompensa é maior que zero, é determinado que o agente Rbt alcançou uma posição suja e logo ele recebe o plano de limpar esta posição. Caso contrario, Rbt requisita pela próxima ação.

```
1     @pl(gain,Goal("move",("Action","Reward")))
2     def move(self,src,action,reward):
3         match action:
4             case 0: direction = (1,0)
5             case 1: direction = (-1,0)
6             case 2: direction = (0,1)
7             case 3: direction = (0,-1)
8         self.pos = (self.pos[0]+direction[0], self.pos[1]+direction[1])
9         if reward > 0:
10            self.add(Goal("clean_dirt"))
11        else:
12            state = Lrn.env.encode(*self.pos)
13            action,n_state,reward = Lrn.exec(Lrn.env, state)
14            self.add(Goal("move",(action,reward)))
```

Algoritmo 2. Plano ação mover

5. Considerações Finais

Este artigo apresentou um mapeamento sistemático de estudos sobre a condução de veículo autônomos que utilizam aprendizagem por reforço. Essa pesquisa procurou identificar os algoritmos e técnicas mais utilizadas, os métodos para sua validação e os cenários em que são testados veículos autônomos. Também foi realizado a descrição e exemplificação de MASPY, uma biblioteca em Python para a implementação de sistemas multiagentes BDI com aprendizagem de máquina.

Entre estas questões analisadas no mapeamento encontrou-se a lacuna do desenvolvimento de agentes BDI utilizando aprendizado por reforço. MASPY, logo, apresenta a capacidade de desenvolver um sistema com os algoritmos Q-learning e Sarsa, porém não possui no momento o algoritmo mais destacado: Deep Q-Networks. Apesar de envolver um cenário de veículo autônomo, o exemplo apresenta a capacidade de movimentação em um cenário 2D utilizada para as decisões de um agente BDI.

Para o desenvolvimento futuro, é importante a validação da biblioteca MASPY. É necessário, por meio de comparações com outras bibliotecas de aprendizagem e desenvolvimento de agentes, mostrar onde se encontram as desvantagens e vantagens da MASPY. Como identificado no mapeamento, essas comparações incluem a determinação das técnicas e os cenários para a avaliação, como controle de velocidade e prevenção de colisão no caso de veículos autônomos. Capacidades futuras também incluem a implementação de novos algoritmos embutidos em MASPY para treinamento, como Deep Q-Networks, e incorporação da aprendizagem durante o próprio raciocínio dos agentes.

Referências

Arvind, C. S. and Senthilnath, J. (2019). Autonomous RL: Autonomous Vehicle Obstacle Avoidance in a Dynamic Environment using MLP-SARSA Reinforcement Learning. In *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*, pages 120–124, Singapore. IEEE.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition.

Bin Issa, R., Das, M., Rahman, M. S., Barua, M., Rhaman, M. K., Ripon, K. S. N., and Alam, M. G. R. (2021). Double Deep Q-Learning and Faster R-CNN-Based Autonomous Vehicle Navigation and Obstacle Avoidance in Dynamic Environment. *Sensors*, 21(4):1468. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

Bosello, M. (2020). Integrating bdi and reinforcement learning: the case study of autonomous driving.

Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Cambridge: Cambridge, MA: Harvard University Press.

Cabezas-Olivenza, M., Zulueta, E., Sanchez-Chica, A., Fernandez-Gamiz, U., and Teso-Fz-Betoño, A. (2023). Stability Analysis for Autonomous Vehicle Navigation Trained over Deep Deterministic Policy Gradient. *Mathematics*, 11(1):132. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

Dar, S. A., Palanivel, S., and Geetha, M. K. Autonomous Taxi Driving Environment Using Reinforcement Learning Algorithms. *International Journal of Modern Education and Computer Science*, 14(3):88.

Dhinakaran, M., Rajasekaran, R., Balaji, V., Aarthi, V., and Ambika, S. (2024). Advanced Deep Reinforcement Learning Strategies for Enhanced Autonomous Vehicle Navigation Systems. In *2024 2nd International Conference on Computer, Communication and Control (IC4)*, pages 1–4.

Freitas, V. and Segatto, W. (2021). Perform Systematic Literature Reviews.

Gao, L., Wu, Y., Wang, L., Wang, L., Zhang, J., and Li, K. (2023). End-to-end autonomous vehicle navigation control method guided by the dynamic window approach. In *2023 IEEE 6th International Electrical and Energy Conference (CIEEC)*, pages 4472–4476.

González-Miranda, O., Miranda, L. A. L., and Ibarra-Zannatha, J. M. (2023). Q-Learning for Autonomous Vehicle Navigation. In *2023 XXV Robotics Mexican Congress (COMRob)*, pages 138–142.

Gronauer, S. and Diepold, K. (2022). Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49.

Hartmann, G., Shiller, Z., and Azaria, A. (2020). Model-Based Reinforcement Learning for Time-Optimal Velocity Control. *IEEE Robotics and Automation Letters*, 5(4):6185–6192. Conference Name: IEEE Robotics and Automation Letters.

Hook, J., El-Sedky, S., De Silva, V., and Kondoz, A. (2021). Learning data-driven decision-making policies in multi-agent environments for autonomous systems. *Cognitive Systems Research*, 65:40–49.

Jacinto, E., Martinez, F., and Martinez, F. (2023). Navigation of Autonomous Vehicles using Reinforcement Learning with Generalized Advantage Estimation. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 14(1). Number: 1 Publisher: The Science and Information (SAI) Organization Limited.

Josef, S. and Degani, A. (2020). Deep Reinforcement Learning for Safe Local Planning of a Ground Vehicle in Unknown Rough Terrain. *IEEE Robotics and Automation Letters*, 5(4):6748–6755. Conference Name: IEEE Robotics and Automation Letters.

Kitchenham, B. A. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.

Lambert, R., Li, J., Wu, L.-F., and Mahmoudian, N. (2021). Robust ASV Navigation Through Ground to Water Cross-Domain Deep Reinforcement Learning. *Frontiers in Robotics and AI*, 8. Publisher: Frontiers.

Liang, J., Weerakoon, K., Guan, T., Karapetyan, N., and Manocha, D. (2023). AdaptiveON: Adaptive Outdoor Local Navigation Method for Stable and Reliable Actions. *IEEE Robotics and Automation Letters*, 8(2):648–655.

Mackay, A. K., Riazuelo, L., and Montano, L. (2022). RL-DOVS: Reinforcement Learning for Autonomous Robot Navigation in Dynamic Environments. *Sensors*, 22(10):3847. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

Mellado, A. L. L., G., F. I., Alves, G. V., and Borges, A. P. (2023). Maspy: Towards the creation of bdi multi-agent systems. In *Proceedings of the 17th Workshop-School on Agents, Environments, and Applications (WESAAC 2023)*, pages 106–117.

Qin, J., Qin, J., Qiu, J., Liu, Q., Li, M., and Ma, Q. (2024). SRL-ORCA: A Socially Aware Multi-Agent Mapless Navigation Algorithm in Complex Dynamic Scenes. *IEEE Robotics and Automation Letters*, 9(1):143–150. Conference Name: IEEE Robotics and Automation Letters.

Shakya, A. K., Pillai, G., and Chakrabarty, S. (2023). Reinforcement learning algorithms: A brief survey. *Expert Systems with Applications*, page 120495.

Sun, Q., Zhang, L., Yu, H., Zhang, W., Mei, Y., and Xiong, H. (2023). Hierarchical Reinforcement Learning for Dynamic Autonomous Vehicle Navigation at Intelligent Intersections. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pages 4852–4861, New York, NY, USA. Association for Computing Machinery.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Taghavifar, H., Wei, C., and Taghavifar, L. (2024). Socially Intelligent Reinforcement Learning for Optimal Automated Vehicle Control in Traffic Scenarios. *IEEE Transactions on Automation Science and Engineering*, pages 1–12. Conference Name: IEEE Transactions on Automation Science and Engineering.

Tiong, T., Saad, I., Teo, K. T. K., and Lago, H. B. (2023). Autonomous Vehicle Driving Path Control with Deep Reinforcement Learning. In *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0084–0092, Las Vegas, NV, USA. IEEE.

Yang, H., Yao, C., Liu, C., and Chen, Q. (2023). RMRL: Robot Navigation in Crowd Environments With Risk Map-Based Deep Reinforcement Learning. *IEEE Robotics*

and Automation Letters, 8(12):7930–7937. Conference Name: IEEE Robotics and Automation Letters.

Zhai, Y., Ding, B., Liu, X., Jia, H., Zhao, Y., and Luo, J. (2021). Decentralized Multi-Robot Collision Avoidance in Complex Scenarios With Selective Communication. *IEEE Robotics and Automation Letters*, 6(4):8379–8386. Conference Name: IEEE Robotics and Automation Letters.

Zhang, W., Gai, J., Zhang, Z., Tang, L., Liao, Q., and Ding, Y. (2019). Double-DQN based path smoothing and tracking control method for robotic vehicle navigation. *Computers and Electronics in Agriculture*, 166:104985.

Zhu, W. and Hayashibe, M. (2023). Autonomous Navigation System in Pedestrian Scenarios Using a Dreamer-Based Motion Planner. *IEEE Robotics and Automation Letters*, 8(6):3836–3843.

Controle de Tráfego Aéreo Autônomo: Simulação de Sistemas Multiagentes com o framework JaCaMo na plataforma Unity

Bernardo Viero[1], **Rafael H. Bordini**[2], **Alexandre Zamberlan**[1]

[1]Curso de Ciência da Computação – Universidade Franciscana (UFN)
Santa Maria – RS

[2]Escola Politécnica – PUCRS
Porto Alegre – RS

{bernardo.viero,alexz}@ufn.edu.br, rafael.bordini@pucrs.br

Abstract. *This research aims to unite the areas of Multi-Agent Systems (MAS) and Simulation in the management and control of autonomous air traffic. The simulated agents represent aircraft, control towers, and runways for landing and takeoff, with the ability to perceive, plan, execute, and communicate with each other. The behavior of the agents should be emergent and dynamic, meaning that the simulations should generate states or situations requiring the agents to plan, communicate, and negotiate. For this, the Java and AgentSpeak(L) languages, and the Jason interpreter are used for server construction, while the C# language and the Unity platform are used for the implementation of the simulations.*

1. Introdução

No universo complexo da gestão e controle do tráfego aéreo, a Simulação Computacional desempenha um papel crucial na tomada de decisões e na garantia da segurança das operações. A interação entre Sistemas Multiagentes e Simulação Computacional abre portas para abordagens desafiadoras na gestão do tráfego aéreo autônomo [1]. Ao unir essas áreas, é possível criar ambientes dinâmicos e emergentes, onde agentes simulados representam não apenas aeronaves, mas também torres de controle e pistas de pouso, dotados de capacidades de percepção, planejamento, execução e comunicação entre si. Isso se traduz em uma ferramenta valiosa para investigar cenários variados, avaliar estratégias de controle e mitigar riscos associados ao tráfego aéreo não tripulado. Além disso, ao utilizar tecnologias como Java, *framework* JaCaMo, AgentSpeak(L) e Unity, podem-se criar ambientes interativos e visualmente ricos, que facilitam o monitoramento e a análise do comportamento inteligente das aeronaves.

Neste contexto, este projeto propõe não apenas projetar e implementar um ambiente de simulação baseada na teoria de Sistemas Multiagente, mas também avaliá-lo de alguma forma. A integração de Jason em um ambiente de simulação Unity oferece a oportunidade de avaliar e aprimorar o desempenho do sistema em diferentes cenários, contribuindo assim para qualificar as discussões na área de controle de tráfego aéreo autônomo.

1.1. Controle de Tráfego Aéreo e Simulação

O controle de tráfego aéreo é um sistema essencial para garantir a segurança e a eficiência das operações aéreas em todo o mundo. Em sua essência, é responsável por gerenciar o

fluxo de aeronaves nos céus, coordenando suas rotas, altitudes e velocidades para evitar colisões e garantir o fluxo ordenado do tráfego aéreo [2]. O funcionamento do controle de tráfego aéreo envolve uma rede complexa de centros de controle, torres de controle e sistemas de comunicação. Nos centros de controle, controladores aéreos monitoram continuamente as posições das aeronaves por meio de radares e sistemas de navegação por satélite, como o GPS. Eles utilizam computadores e sistemas de gestão de tráfego aéreo para acompanhar o movimento das aeronaves e determinar as melhores rotas e altitudes para cada voo.

A simulação é um método que envolve a criação de um modelo de um sistema real e a condução de experimentos com esse modelo, com o objetivo de compreender os comportamentos do sistema ou avaliar estratégias para sua operação [3]. Essa prática pode ser vista como uma ferramenta computacional que permite o desenvolvimento, teste e estudo de teorias, visando a compreensão de sistemas reais e o desenvolvimento de sistemas computacionais [4]. Nesse contexto, os veículos aéreos são representados como agentes autônomos que possuem capacidade de percepção, planejamento, comunicação e tomada de decisão, com isso, são definidos como um Sistemas Sensíveis ao Contexto (*Context-Aware Computing*) [5].

1.2. Trabalhos Correlatos

Na pesquisa de Alexandre Zamberlan [6], foi conduzido um estudo que ressaltou a aplicação da teoria de Sistemas Multiagentes em Simulações Computacionais. Essas simulações foram realizadas em ambientes contendo partículas poliméricas nanoestruturadas, que consistiam em fármacos encapsulados em polímeros. No contexto, as partículas atuavam como agentes autônomos, interagindo entre si e com o ambiente. O trabalho empregou agentes e Sistemas Multiagentes com uma arquitetura reativa, visando monitorar se as partículas demonstravam comportamento de aglomeração ou não, influenciado pela carga elétrica de cada uma e pelo pH do ambiente. Todas as interações no sistema seguiram as regras de colisão da teoria Browniana do movimento.

No trabalho de Gabriel Dal Forno [7], foi ressaltado a implementação de um sistema multiagentes baseado em tráfego urbano autônomo. A arquitetura da simulação foi baseada em um simulador (ambiente Unity e linguagem C#) e um sistema multiagentes (ambiente Jason) com a utilização de *Swarm Intelligence* [8]. Com isso, o sistema multiagentes comportou-se como um servidor e o simulador como um cliente, comunicando-se via protocolo TCP/IP. A metodologia utilizada, para a implementação do sistema multiagentes, foi através do PROMETHEUS [9]. Já os resultados alcançados foram pertinentes, como o estabelecimento da comunicação eficaz entre o simulador e o sistema multiagentes através de *sockets* e o uso de *JavaScript Object Notation* (JSON), e a implementação dos planos que lidam com os eventos ativadores dos agentes em Jason.

O projeto de Carlos Pantoja [10] desenvolveu um agente inteligente para controlar uma aeronave em um simulador de voo, com foco na integração do Jason com o simulador de voo X-Plane. O agente foi responsável por todas as tarefas de voo, incluindo decolagem, navegação e pouso. O agente foi programado usando técnicas de aprendizado de máquina e controle de sistemas. Foi fixado um aeroporto para que o agente pudesse realizar a decolagem, e também foi necessário a criação de planos para que o agente pudesse decolar de qualquer aeroporto sem uma prévia configuração. O agente apresentou um

bom controle da aeronave na maioria das simulações, porém em algumas ocasiões houve desvios do percurso durante a decolagem e inclinações horizontais que não foram corrigidas. Esses comportamentos foram atribuídos à simplicidade do raciocínio do agente e às condições climáticas variáveis do simulador. No entanto, a codificação de planos e o acesso a informações adicionais permitiram melhorias significativas no controle da aeronave.

2. Metodologia

Na condução da pesquisa, adotou-se a metodologia Scrum, conforme descrita por Silveira [11], juntamente com a técnica Kanban para gerenciar as atividades assumidas. Os ciclos de trabalho, *sprints*, ocorrem semanalmente, baseando-se nas funcionalidades mapeadas e integradas ao Trello para Kanban. Para desenvolver o sistema multiagente (servidor da simulação), optou-se por utilizar o interpretador Jason, empregando as linguagens AgentSpeak(L) e Java, fundamentados na metodologia JaCaMo. Para o ambiente de simulação (cliente), foi selecionado o software Unity, utilizando C# como linguagem de programação. Esta escolha foi motivada pela variedade de classes disponíveis no Unity, que são úteis na criação de ambientes simulados, como colisores, vetores de movimento e facilidades para implementação de paralelismo. Por fim, foi decidido usar *sockets* para comunicação, por meio do protocolo TCP, entre a aplicação de simulação e a aplicação do sistema multiagentes, devido à sua fácil implementação em ambas as linguagens utilizadas.

Para a pesquisa, aplica-se um ambiente de avaliação e teste que trate das metas e planos dos agentes (torre de comando, pista, aviões); dos valores das variáveis para aviões (altura, nível de combustível e se o aeroporto é uma escala desse avião em voo); e da quantidade de aviões a decolar e/ou pousar. Com isso, busca-se avaliar se o auto gerenciamento dos agentes, por meio de negociação e planejamento descentralizado, minimiza as filas de espera e/ou promove um fluxo mais eficiente de decolagens e pousos. A avaliação do funcionamento dos agentes deve ser realizada por meio dos cenários definidos pelo usuário. Cada cenário é testado com diferentes valores para as variáveis mapeadas, que geram volumes de tráfego aéreo, visando entender o sistema.

Como a metodologia JaCaMo integra três plataformas: Jason, CArtAgO e Moise, há alguns passos a seguir:

Identificação dos Agentes e suas características:

- Agente Avião com propriedades como altitude, nível de combustível e se tem escala naquele aeroporto. Comportamentos como decolar, voar, pousar;
- Agente Pista com propriedade como fila de aviões prontos para decolar;
- Agente Torre de Controle com a responsabilidade de gerenciar o fluxo de aviões, tanto para decolagem quanto para pouso.

Definição de Artefatos (usados pelos agentes para interagir com o ambiente):

- Artefato Fila de Decolagem que gerencia a fila de aviões esperando para decolar;
- Artefato Fila de Pouso que gerencia a fila de aviões esperando para pousar.

Estrutura Organizacional (Moise):

- Papéis: controlador de pouso (gerenciado pela Torre de Controle); controlador de decolagem (gerenciado pela Torre de Controle); piloto (papel assumido pelos agentes avião);

- Normas: aviões devem negociar entre si e receber autorização para decolar ou pousar; a Torre de Controle deve garantir comunicação a todos os agentes.

Com base nas definições discutidas, o diagrama da Figura 1 fornece uma visão geral do sistema multiagente usando a metodologia JaCaMo, mostrando como os agentes interagem com os artefatos para realizar o controle descentralizado do tráfego aéreo. O diagrama da Figura 2, mostra o fluxo dos processos (modelagem de funcionamento do simulador) para a execução da simulação com relação ao usuário, ao simulador e ao Sistema Multiagente.

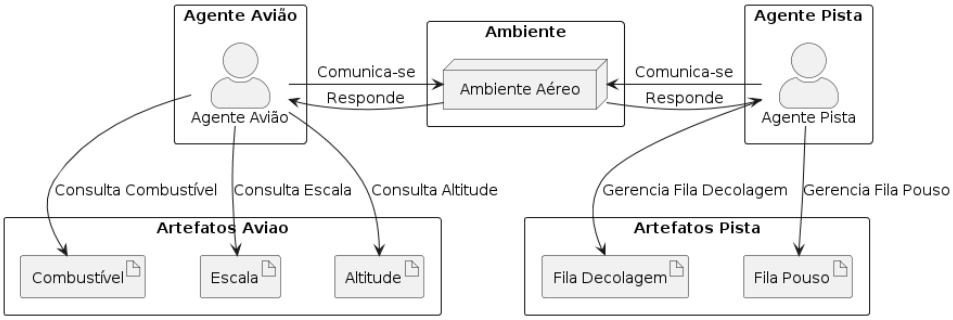


Figura 1. Diagrama de visão geral do sistema multiagente [12].

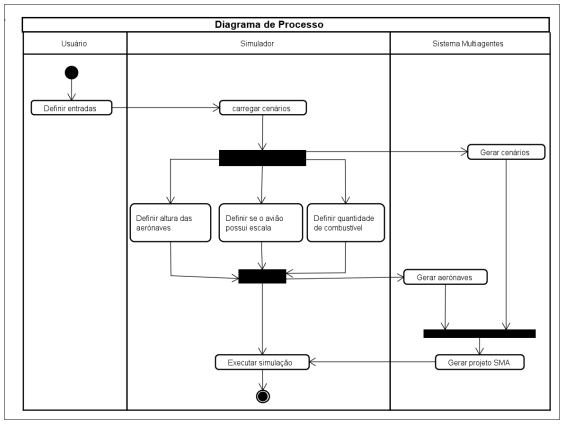


Figura 2. Diagrama do processo da simulação [12].

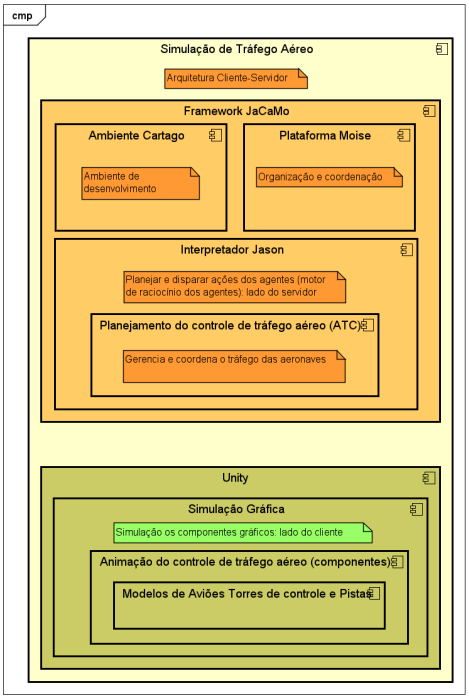


Figura 3. Diagrama de componentes [12].

As simulações gráficas, ao receberem comunicação do JaCaMo por meio da plataforma Unity, podem gerenciar graficamente esse ambiente. Isso inclui a exibição de aeronaves com rotas planejadas e detecção de colisões, torres de controle e pistas de pouso e decolagem. Conforme observado na Figura 3, a arquitetura adotada pelo projeto

é de natureza cliente-servidor. O servidor corresponde à implementação do *framework* JaCaMo, encarregado de acionar ações por meio do interpretador Jason para a linguagem AgentSpeak(L) conforme as circunstâncias de cada agente. Desse modo, o sistema de controle de tráfego aéreo é capaz de gerenciar e coordenar o fluxo das aeronaves. Por sua vez, o cliente assume a responsabilidade de receber tais ações provenientes do servidor e realizar simulações gráficas por meio da plataforma Unity. Em outras palavras, toda a parte de animações do tráfego aéreo, bem como os modelos de aeronaves e pistas, é renderizada por essa plataforma.

Decidiu-se o uso de *socket* para comunicação, via protocolo TCP (tanto do servidor ao simulador, quanto do simulador ao servidor), entre a aplicação de simulação e a aplicação do Sistema Multiagente, devido à sua implementação fácil em ambas as linguagens empregadas, além de interconectar dois sistemas heterogêneos (servidor em Java, com AgentSpeak(L) e Jason; cliente em C#, com Unity). A Figura 4 ilustra a dinâmica entre o simulador e o Sistema Multiagente em relação à comunicação entre as tecnologias. A Figura 5 apresenta uma proposta inicial de interface gráfica para o sistema, destacando a visualização da simulação.

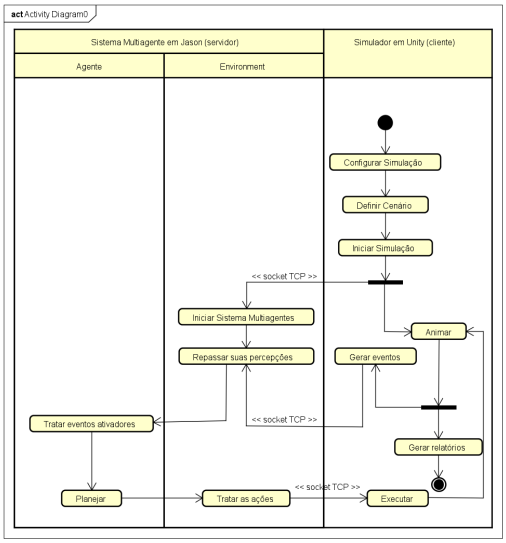


Figura 4. Diagrama de atividades para integração do simulador [12].

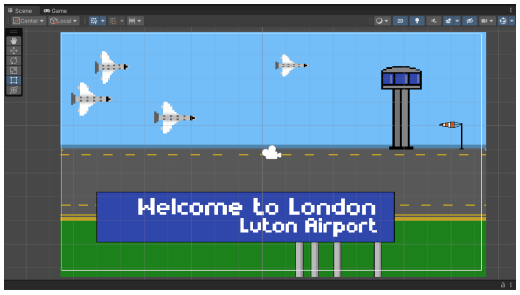


Figura 5. Protótipo de simulação [12].

Com o uso do sistema de comunicação baseado em *socket*, onde o Sistema Multiagente abriga o servidor e o ambiente de simulação abrange o cliente, a troca de informações entre os dois lados é realizada por meio de *strings* no formato JSON. Esse formato foi escolhido devido à disponibilidade de bibliotecas de codificação e decodificação em ambas as linguagens utilizadas. As informações enviadas do cliente para o servidor contêm dados relativos aos agentes e outros objetos. Enquanto as instruções destinadas às aeronaves da simulação, definidas pelos agentes do Sistema Multiagente, são transmitidas do servidor para o cliente. O protocolo TCP foi selecionado para a implementação, pois a perda de uma instrução/ação do Sistema Multiagente poderia causar falta de sincronia entre os dois lados e resultar em erros que poderiam levar a colisões ou acidentes na simulação.

3. Considerações finais

Este trabalho propõe um projeto de um sistema de simulação de tráfego aéreo por meio de sistemas multiagentes. Especificamente, o sistema modelado consiste de aviões em um aeroporto e uma torre de controle que devem ser capazes de se comunicar e auto-organizar, visando a geração de um fluxo de tráfego eficiente. Além disso, os agentes do sistema devem aprender a interagir com outros agentes independentes.

A pesquisa encontra-se na fase final de modelagem e especificação do sistema de simulação via SMA. Toda a funcionalidade de comunicação entre o servidor (Jason) e o ambiente de simulação (Unity) já está preparada, aguardando a implementação dos comportamentos dos agentes e dos cenários de simulação.

Referências

[1] Wenhui Fan et al. "Multi-Agent Modeling and Simulation in the AI Age". Em: *TSINGHUA SCIENCE AND TECHNOLOGY* (2021). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9409754.

[2] Elmira Zohrevandi et al. "Design and evaluation study of visual analytics decision support tools in air traffic control". Em: *Computer Graphics Forum*. Vol. 41. 1. Wiley Online Library. 2022, pp. 230–242.

[3] "Como construir modelos de simulação válidos e confiáveis". Em: *Conferência de Simulação de Inverno 2022 (WSC)*. IEEE, 2022, pp. 1283–1295.

[4] Adelinde M. Uhrmacher e Danny Weyns. *Multi-Agent Systems: Simulation and application*. Computational analysis, synthesis, and design of dynamic models series. Boca Raton, FL, USA: CRC Press, 2009.

[5] Mir Salim Ul Islam, Ashok Kumar e Yu-Chen Hu. "Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions". Em: *Journal of Network and Computer Applications* 180 (2021), p. 103008.

[6] Alexandre Zamberlan et al. "Multi-Agent Systems, Simulation and Nanotechnology". Em: *Multi Agent Systems-Strategies and Applications*. IntechOpen, 2020.

[7] Gabriel Dal Forno, Rafael Heitor Bordini e Alexandre Zamberlan. *Simulação de tráfego autônomo com Inteligência Coletiva e Emergente*. Pimenta Cultural - Upgrade: jogos, entretenimento e cultura, 2024. URL: https://www.pimentacultural.com/livro/upgrade-jogos-2/ (acesso em 20/05/2024).

[8] Eric Bonabeau, Marco Dorigo e Guy Theraulaz. *From Natural to Artificial Swarm Intelligence*. USA: Oxford University Press, Inc., 1999. ISBN: 0195131584.

[9] Lin Padgham e Michael Winikoff. *Developing Intelligent Agent Systems: A practical guide.* John Wiley & Sons, 2004.

[10] Carlos Eduardo Pantoja e Tielle da Silva Alexandre. "Um Agente Inteligente para Simulação de Voo Usando Jason e X-Plane". Em: *8th Software Agents, Environments and Applications School (WESAAC)* (2014).

[11] Paulo Silveira et al. *Introdução à Arquitetura de Design de Software*. Ed. por Elsevier. 2011.

[12] Bernardo Viero. *images-tfg*. 2024. URL: https://github.com/bernardoviero/TFG/tree/main/images-tfg (acesso em 18/06/2024).

Comunicação entre agentes no *framework* Embedded-BDI*

Vitor Luis Babireski Furio [1], Maiquel de Brito[1], Carlos Roberto Moratelli[1]

[1]Universidade Federal de Santa Catarina (UFSC)
Rua João Pessoa, 2750 – 89036-002 – Blumenau – SC – Brasil

vitorluis.babireskifurio@gmail.com,{maiquel.b,carlos.moratelli}@ufsc.br

Abstract. *The Embedded-BDI framework provides resources for the implementation and execution of BDI agents on hardware with limited resources, typical of cyber-physical systems, without providing resources for implementing communication among the agents. This work extends the framework by adding features to specify communication between agents using high-level elements related to the BDI model and to transmit messages using protocols and resources consolidated in cyber-physical systems. The experimentally evaluated results demonstrate that the extension adds communication capability to agents.*

Resumo. *O* framework Embedded-BDI *provê recursos para a implementação e execução de agentes BDI em* hardwares *com recursos limitados, típicos de sistemas ciberfísicos, sem fornecer recursos para implementar comunicação entre agentes. Este trabalho estende o* framework *adicionando recursos para especificar a comunicação entre agentes usando elementos de alto nível relacionados ao modelo BDI e para transmitir as mensagens usando protocolos e recursos consolidados em sistemas ciberfísicos. Os resultados, avaliados experimentalmente, demonstram que a extensão proposta adiciona capacidade de comunicação aos agentes.*

1. Introdução

Sistemas ciberfísicos (ou CPS, do inglês *Cyber-Physical Systems*) combinam programas de computador, embarcados em diferentes dispositivos, com sensores e atuadores, para, assim, interagir com o ambiente físico. Aplicações complexas podem requerer que CPS tenham características de agentes BDI, como (i) autonomia, para atuar sem a intervenção de um operador humano, (ii) proatividade, para tomar a iniciativa de atingir objetivos de longo prazo, (iii) reatividade, para adaptar-se a novas circunstâncias de operação, e (iv) habilidades sociais, para interagir com outros CPS. A programação orientada a agentes (ou AOP, do inglês *Agent-Oriented Programming*) fornece modelos e ferramentas para o desenvolvimento de agentes que, normalmente, requerem mais recursos computacionais (processamento, memória etc) do que dispõem os *hardwares* típicos de CPS. Para tratar desta limitação, o *framework Embedded-BDI*[1] [Santos 2022] fornece ferramentas para implementar e executar agentes BDI neste tipo de *hardware*.

Agentes desenvolvidos com o *framework Embedded-BDI* possuem autonomia, proatividade e reatividade. Este trabalho propõe uma extensão para que o *framework* forneça suporte a comunicação entre agentes. Os resultados são avaliados experimentalmente através de uma aplicação desenvolvida com o *framework* estendido.

[1]https://embedded-bdi.github.io/index.html

2. Fundamentação

2.1. Agentes, AOP e o *framework Embedded-BDI*

Um *agente* é um sistema que percebe o ambiente em que está situado e atua de forma autônoma para satisfazer os objetivos para os quais foi projetado [Wooldridge 2009]. Um agente que segue a arquitetura BDI atua a partir de crenças, que são as informações que ele possui sobre o ambiente, desejos (ou objetivos), que são os estados do mundo que ele gostaria de atingir, e intenções, que são os estados do mundo que ele está atuando para atingir [Rao and Georgeff 1995]. Agentes BDI executam um ciclo de raciocínio contínuo em que (i) percebem o ambiente e atualizam suas crenças a partir disso, (ii) avaliam, a partir das crenças atuais, quais objetivos são factíveis, passando a ter intenções de realizá-los e (iii) executam planos de atuação para satisfazer as intenções.

AgentSpeak [Rao 1996] é uma linguagem para programação de agentes BDI especificando crenças, objetivos e planos. O *framework Embedded-BDI* utiliza uma versão simplificada de *AgentSpeak* para programação de agentes e para execução em *hardwares* limitados, típicos de CPS. A Figura 2 ilustra um código nesta versão. Ela contém todos os elementos da linguagem original. Porém as crenças e objetivos são expressos através de proposições, enquanto no *AgentSpeak* original, são expressos através de predicados. O código *AgentSpeak*, escrito em arquivos com extensão *.asl*, é traduzido para C++ e compilado junto com outros módulos escritos em C++ que implementam (i) funções para conectar ações e atualização de crenças do agente aos sensores e atuadores físicos; e (ii) uma *engine BDI*, que executa o ciclo de raciocínio BDI. Estes componentes são compilados e embarcados no *hardware* (Figura 1).

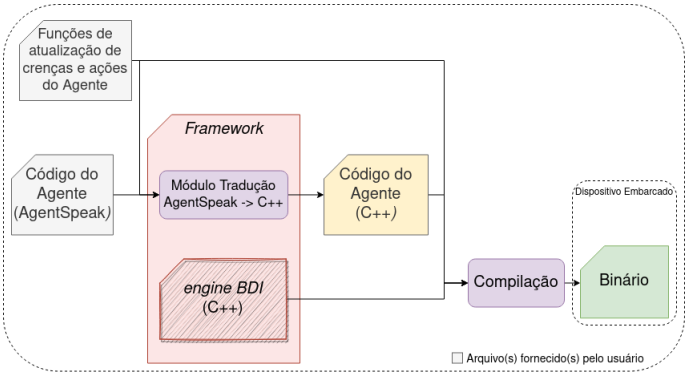


Figura 1. Elementos envolvidos na compilação do agente [Santos 2022]

Para tratar das limitações de memória típicas em CPS, não há alocação dinâmica de memória na *engine BDI*. Por isso, todas as crenças que o agente pode vir a possuir, bem como todos os objetivos que o agente pode buscar atingir, são instanciados quando ele é incializado.[2] Estas crenças e objetivos são aqueles que constam no código fonte do agente (.asl). Por exemplo, o agente implementado pelo código da Figura 2 possuirá somente a crença *happy* e poderá ter somente os objetivos *start* e *hello*. Durante a execução do agente, os objetivos instanciados podem ser ativados e desativados e as crenças instanciadas são associadas ao valor *verdadeiro* quando o agente possui a crença e *falso* caso contrário. Para economizar memória, os identificadores de crenças e objetivos são substituídos por números inteiros no código C++ gerado com a tradução do .asl, seguindo a ordem em que aparecem neste último. Para o código ilustrado na Figura 2, os identificadores *start*, *happy* e *hello* são mapeados, respectivamente, para os valores 0, 1 e 2 (mais detalhes na Seção 3.3).

[2]Há outros elementos que também são instanciados na inicialização do agente, mas que não são relevantes neste trabalho.

```
1  !start. // desejo (ou objetivo)
2  +!start <- +happy. //plano para satisfazer o objetivo ``start''
3  +happy <- !!hello.  //plano para tratar a nova crenca ``happy''
4  +!hello <- say_hello. //plano para satisfazer o objetivo ``hello''
```

Figura 2. Exemplo de código fonte de agente

2.2. MQTT

O *Message Queuing Telemetry Transport* (MQTT) é um protocolo de mensagens leve, projetado para dispositivos IoT (Internet das Coisas). Ele segue o modelo de comunicação de publicação/assinatura. Dispositivos *clientes* se conectam a um servidor central (ou *broker*), que gerencia a comunicação entre eles. Os clientes publicam mensagens em *tópicos*, que são canais de comunicação hierárquicos. Outros clientes podem se inscrever nesses tópicos de interesse. Quando uma mensagem é publicada em um tópico, o *broker* a recebe e a distribui para todos os clientes inscritos nesse tópico.

3. Desenvolvimento

A adição de capacidade de comunicação entre agentes ao *framework Embedded-BDI* requer integrar o ciclo de raciocínio à infraestrutura de comunicação disponível, conciliar as mensagens transmitidas através dessa infraestrutura com as representações do modelo BDI e adaptar as funcionalidades de comunicação adicionadas ao *framework* às suas premissas para otimização de uso de recursos computacionais (cf. seções 3.1 a 3.3).

3.1. Integração do ciclo de raciocínio à infraestrutura de comunicação

Os diferentes *hardwares* usados em CPS têm interfaces e métodos de acesso a redes de comunicação próprios. Para prover alguma generalidade ao *framework*, propõe-se basear a transmissão em algum protocolo para o qual os principais *hardwares* tenham bibliotecas apropriadas. Neste trabalho, considera-se o protocolo MQTT. A integração do ciclo de raciocínio do agente com a infraestrutura de comunicação sobre a qual as mensagens são transmitidas é feita integrando a *engine BDI* ao serviço MQTT. Ela incorpora funcionalidades para ler e escrever em tópicos mantidos pelo *broker*, que é externo ao agente. Ao inicializar, a *engine BDI* se inscreve em (ou *assina*) dois tópicos MQTT. Um deles tem o nome do próprio agente e destina-se a armazenar as mensagens enviadas para o agente em particular. O segundo tópico, de nome *broadcast*, destina-se a armazenar as mensagens que os agentes enviam a todos os demais. No início de cada ciclo de raciocínio, a *engine BDI* detecta novas mensagens nestes tópicos, as interpreta e, na sequência do ciclo, atualiza o estado interno do agente.

3.2. Conciliação das mensagens transmitidas com a semântica do modelo BDI.

As mensagens trocadas entre os agentes têm, inevitavelmente, relação com as abstrações do modelo BDI que representam seu estado interno, podendo, por exemplo, afetar suas crenças e objetivos. É necessário, assim, ligar as mensagens trocadas a essas abstrações. Como é usual em AOP, considera-se, neste trabalho, que os agentes comunicam-se utilizando uma linguagem de comunicação entre agentes (ou ACL, do inglês *agent communication language*). A ACL considerada neste trabalho é KQML (ou *Knowledge Query and Manipulation Language*) [Finin et al. 1994]. Nela, as mensagens são compostas não

só pelo conteúdo a ser comunicado, mas também por uma *performativa* que expressa a intenção que o emissor possui ao enviar a mensagem. As performativas relevantes neste trabalho são *tell*, que denota a intenção do emissor de que o receptor possua uma determinada crença, e *achieve*, que denota a intenção do emissor de que o receptor tenha um determinado objetivo. Por exemplo, um agente que envia o conteúdo *happy* acompanhado da performativa *tell* tem a tem a intenção de que o destinatário passe a ter a crença *happy*. De forma semelhante, um agente que envia o conteúdo *do_something* acompanhado da performativa *achieve* tem a intenção de o receptor passe a ter o objetivo *do_something*.

A comunicação através de KQML e a sua ligação com o ciclo de raciocínio do agente requerem adaptações no *framework* para tratar do envio e recepção de mensagens. Quanto ao envio, considera-se que a capacidade de enviar uma mensagem é parte da implementação do próprio agente, incorporada à *engine BDI*, e não depende de elementos do ambiente. Assim, com inspiração na linguagem *Jason* [Bordini et al. 2007], que também estende *AgentSpeak*, adiciona-se um novo construto, chamado *ação interna*, à linguagem de programação. A linguagem de programação é estendida para incluir as ações internas *send* e *broadcast*, que permitem que o agente envie uma mensagem a outro agente ou a todos os demais, respectivamente. Elas são especificadas na forma de instruções no codigo .asl, como, por exemplo, *.broadcast(tell,happy)*, em que agente emissor envia um *broadcast* com conteúdo *happy* e performativa *tell*; ou *.send(bob,achieve,do_something)*, em que o agente emissor envia uma mensagem diretamente ao agente *bob*, com conteúdo *do_something* e performativa *achieve*.[3] Em tempo de execução, a *engine BDI* executa tudo o que é necessário para que instruções como estas resultem na escrita em tópicos MQTT. A *engine BDI* é estendida também para, ao receber mensagens via MQTT, tratá-las de forma adequada em função da sua performativa: ao processar uma mesagem com a performativa *tell*, atualiza a crença correspondente e, no caso da performativa *achieve*, atualiza os objetivos do agente.

3.3. Adaptação às representações internas do *framework Embedded-BDI*

A representação interna de crenças e objetivos através de números inteiros (cf. Sec. 2.1) impõe um desafio à comunicação entre agentes pois as mesmas crenças e objetivos podem ser mapeados para valores diferentes em agentes cujo código .asl é também diferente. Por exemplo, no caso do caso o agente codificado na Figura 2, a ação interna *.broadcast(tell,happy)*, ao ser traduzida para C++, resultará em uma instrução para enviar uma mensagem com a performativa *tell* e o conteúdo 1, que é associado à crença *happy* naquele agente. Não e possível afirmar, no entanto, que a crença será associada valor 1 no agente receptor. Para tratar deste problema, optou-se em transmitir, na mensagem, o identificador das crenças e objetivos em vez de seu valor inteiro correspondente. Para isso, a *engine BDI* contém uma estrutura de dados chamada *msg_list* que armazena a lista de identificadores e os números inteiros correspondentes.

O processo de envio da mensagem segue as seguintes etapas: (i) busca-se na *msg_list* o identificador correspondente ao número da crença ou objetivo a enviar, e (ii) envia-se esse identificador ao destinatário pelo protocolo MQTT. Para tratar as mensagens recebidas, verifica-se inicialmente se o conteúdo da mensagem, que é um identificador de

[3]No código C++ gerado a partir da tradução do asl, os identificadores de crenças e objetivos (como *happy* e *do_something* neste exemplo) são substituídos por números inteiros (cf. Sec. 2.1). Assim, as mensagens enviadas contém uma performativa e um número inteiro.

crença ou objetivo, está instanciado na *engine BDI*. Caso afirmativo, processa-se a mensagem, convertendo seu conteúdo para o número inteiro correspondente, e então atualizando o estado da crença ou objetivo em função da performativa enviada na mensagem.

4. Resultados

A comunicação entre agentes é ilustrada através de um exemplo envolvendo os agentes *bob* e *alice*.[4] *Bob*, cujo código é ilustrado na Figura 3-a, inicia com o objetivo *start* e a crença *its_night* (linhas 1–2). Ele tem um plano para satisfazer o objetivo *start* (linha 3), no qual envia um *broadcast* com a intenção de que os receptores tenham o objetivo *hello*. *Alice*, cujo código é ilustrado na Figura 3-b, recebe a mensagem e adquire tal objetivo. Para satisfazê-lo, executa o plano das linhas 1–3, em que (i) executa a ação *say_hello* e (ii) envia uma mensagem a *bob* com a intenção de que este tenha o objetivo *is_day*. Ao receber a mensagem, *bob* adquire tal objetivo e, para satisfazê-lo, executa o plano das linhas 6–8, em que (i) executa a ação *say_night* e envia para *alice* uma mensagem com a intenção de que ela possua a crença *its_night*. *Alice* recebe a mensagem, adquirindo a referida crença e, como reação à aquisição, executa o plano da linha 4, disparando a ação *say_its_night*. O *log* da execução dos agentes é exibido nas figuras 4 e 5.

O exemplo demonstra capacidade de comunicação entre agentes. A troca de mensagens é viabilizada pelo protocolo MQTT. Mas, em vez de programar-se mera troca de mensagens naquele protocolo, programa-se comunicação entre agentes, utilizando construtores de alto nível que relacionam os atos de comunicação às intenções do agente ao enviar a mensagem e com as representações do modelo BDI que compõem o estado interno do agente. As ações dos agentes neste exemplo são realizadas por meio de mensagens impressas na tela em vez de atuações feitas por atuadores físicos. Isso é suficiente para avaliar a possibilidade de comunicação entre agentes, considerando-se que a utilização de atuadores físicos é suportada pelo *framework* desde sua versão inicial [Santos 2022].

```
1 !start.
2 its_night.
3 +!start
4   <- say_hello;
5      .broadcast(achieve,hello).
6 +!is_day
7   <- say_night;
8     .send(alice,tell,its_night).
```

(a) bob

```
1 +!hello
2   <- say_hello;
3      .send(bob,achieve,is_day).
4 +its_night <- say_its_night.
```

(b) alice

Figura 3. Códigos dos agentes

5. Conclusões

Os resultados satisfazem o objetivo deste trabalho, que é adicionar funcionalidades para comunicação entre agentes ao *framework Emebdded-BDI*. Comunicação entre agentes em sistemas embarcados é tratada em outros trabalhos (ex.: [de Jesus et al. 2023, Ortiz et al. 2022]). Quanto ao *framework Emebdded-BDI*, a funcionalidade de *broadcast*

[4]O código completo do exemplo, bem como instruções para execução, encontram-se em `http://github.com/VitorFurio/Embedded-bdi`

```
Message arrived on topic broadcast: ACHIEVE/hello
Hello Everyone, I'm Alice!
Sending message to topic bob: ACHIEVE/is_day
Message arrived on topic alice: TELL/its_night
Meh, it's night, I'm going to sleep...
```

Figura 4. Saída do console de Alice

```
Hello, I'm Bob!!
Sending message to topic broadcast: ACHIEVE/hello
Message arrived on topic bob: ACHIEVE/is_day
It's night now, Alice.
Sending message to topic alice: TELL/its_night
```

Figura 5. Saída do console de Bob

é adicionada em [William 2022]. No entanto, aquela abordagem é fortemente acoplada a um *hardware* específico, enquanto este trabalho inclui um certo nível de generalidade ao usar MQTT. Testes preliminares indicam que os agentes implementados com a nova versão do *framework* podem ser executados em dispositivos ESP32 ou equivalentes. Trabalhos futuros incluem testes com outros dispositivos, bem como a medição dos recursos computacionais demandados pelas funcionalidades adicionadas neste trabalho.

Referências

Bordini, R. H., Hübner, J. F., and Wooldridge, M. J. (2007). *Programming Multi-Agent Systems in AgentSpeak Using Jason*. J. Wiley.

de Jesus, V. S., Lazarin, N. M., Pantoja, C. E., Manoel, F. C. P. B., Alves, G. V., and Viterbo, J. (2023). A middleware for providing communicability to embedded MAS based on the lack of connectivity. *Artif. Intell. Rev.*, 56(Supplement 3):2971–3001.

Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994). Kqml as an agent communication language. In *CIKM'94*, page 456–463, New York, USA. ACM.

Ortiz, G., Zouai, M., Kazar, O., de Prado, A. G., and Boubeta-Puig, J. (2022). Atmosphere: Context and situational-aware collaborative iot architecture for edge-fog-cloud computing. *Computer Standards Interfaces*, 79:103550.

Rao, A. S. (1996). Agentspeak(l): BDI agents speak out in a logical computable language. In *MAAMAW 1996*, volume 1038 of *LNCS*, pages 42–55. Springer.

Rao, A. S. and Georgeff, M. P. (1995). BDI agents: From theory to practice. In *ICMAS'1995*, pages 312–319.

Santos, M. (2022). Programação orientada a agentes bdi em sistemas embarcados. Dissertação (mestrado), Universidade Federal de Santa Catarina, Florianópolis, Brasil.

William, J. (2022). Multi-agent systems on ultra low power platforms. Dissertação (mestrado), ETH Zurich, Suiça.

Wooldridge, M. J. (2009). *An Introduction to MultiAgent Systems, Second Edition*. Wiley.

Aplicando Sistemas Multi-agentes Embarcados no monitoramento de deslizamentos

Gabriel A. Klein, Matheus de S. P. Silva, Washington A. Pedro, Nilson M. Lazarin

Bacharelado em Engenharia Elétrica – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ) – Nova Friburgo, RJ – Brasil

{washington.pedro,matheus.perdigao,gabriel.klein}@aluno.cefet-rj.br

Abstract. *Embedded Multi-agent Systems, based on the Belief-Desire-Intention model, have been consolidated recently. This paper explores its application in a landslide monitoring system without human intervention. To this end, a prototype capable of sending perceptions of the physical environment to the cognitive agent was built. Furthermore, through an Internet of Things (IoT) network, different SMAs are integrated, allowing propagation of alerts in risk situations.*

Resumo. *Sistemas Multi-agentes Embarcados, baseados no modelo belief-desire-intention (BDI), têm se consolidado nos últimos anos. Neste trabalho exploramos sua aplicação em um sistema de monitoramento de deslizamento, sem a intervenção humana. Para tal, foi construído um protótipo capaz de enviar ao agente cognitivo as percepções do ambiente físico. Além disso, por meio de uma rede de internet das coisas (IoT), diferentes SMAs são integrados, permitindo uma rápida propagação de alerta em situações de risco.*

1. Introdução

A Região Serrana do estado do Rio de Janeiro é formada por um relevo com colinas dissecadas, escarpas serranas e grande domínio montanhoso. Além de existir altos índices de chuvas fortes, a região acumula um histórico de grandes desastres ambientais, tal como o ocorrido na cidade de Nova Friburgo em 2011 que ocasionou centenas de mortes e teve milhares de feridos, além disso, esse desastre foi considerado pela ONU o 8° maior em número de deslizamentos ocorridos nos então últimos cem anos. Além disso, mais de 50% da área total do estado do Rio de Janeiro está sob risco de deslizamentos, com ocorrências constantes de deslizamentos durante períodos chuvosos. Este cenário alarmante destaca a importância de pesquisas que visem o desenvolvimento de tecnologias avançadas para o monitoramento de áreas de risco, visando à mitigação de desastres e à proteção das comunidades afetadas. [Marques and Baesso 2021, Filho et al. 2024].

Em [Leão and Souza 2018], é apresentado um sistema de inclinação e um data logger, cujo foi aplicado em uma universidade. O sistema utiliza apenas o monitoramento de inclinação por meio de acelerômetros e por meio de um módulo APC, realiza a transmissão dos dados via rádio. Em [Junior et al. 2016] a fibra óptica é utilizada como sensor. O sistema de monitoramento consiste em fixar a fibra óptica entre dois pontos. No intermédio são feitos laços de fibra, quando existe movimentação no solo a fibra se estica,

apertando o laço e ocasionando uma perda no sinal de luz. O sensor funciona realizando a medição da potência óptica que atravessa o percurso de fibra, onde existe um laser para emitir a luz e um receptor para recebê-la. Esse nível de potência é transmitido via rádio a uma unidade de coleta de dados remota, onde é analisado.

O que este trabalho se difere com maior expressividade em relação aos citados, se dá pela utilização de mais variáveis de monitoramento, pois utiliza dados de vibração do solo e também sua umidade. Além disso, o sistema proposto abre mão de um monitoramento humano, sendo capaz de analisar os dados de maneira automática e ainda realizar comunicação com os órgãos competentes para que medidas sejam tomadas. Ademais, a proposta em questão apresenta baixo custo e fácil instalação, permitindo maior abrangência de áreas de monitoramento. A implementação dessas tecnologias pode não apenas melhorar a precisão das previsões, mas também proporcionar uma base para o desenvolvimento de políticas públicas eficazes e estratégias de gestão de risco.

Os Sistemas Multi-Agentes (SMA) são sistemas formados por entidades de software que possuem capacidade de tomada de decisão segundo as percepções do ambiente em que estão inseridos; trabalham em conjunto, em função de realizar seus objetivos; e são programados utilizando uma linguagem de programação orientada a agentes [Bordini and Vieira 2003]. Nos últimos anos, o uso de agentes racionais embarcados em plataformas de hardware de baixo custo têm se consolidado [Oliveira et al. 2021, Vachtsevanou et al. 2023, Lazarin. et al. 2023]. Essa integração hardware-software permite que agentes cognitivos possam perceber e agir no ambiente físico, de forma descentralizada, reduzindo a dependência de conectividade, realizando a tomada de decisão diretamente na *edge computing* [Pantoja et al. 2023]. A linguagem AgentSpeak(L) permite a criação de agentes racionais, através do modelo de cognição Belief-Desire-intention (BDI), que é estruturado, resumidamente, da seguinte forma: "*As crenças representam aquilo que o agente sabe sobre o estado do ambiente e dos agentes naquele ambiente (inclusive sobre si mesmo). Os desejos representam estados do mundo que o agente quer atingir (dito de outra forma, são representações daquilo que ele quer que passe a ser verdadeiro no ambiente) [...]. As intenções representam sequências de ações específicas que um agente se compromete a executar para atingir determinados objetivos*" [Bordini and Vieira 2003].

Na construção de um SMA, podem ser empregados agentes de diferentes arquiteturas para realização de tarefas específicas. Um agente Jason [Hübner et al. 2004] é um agente comum que desenvolve seu processo de raciocínio com base em informações que o ambiente provê, seus desejos e crenças, e mensagens recebidas de outros agentes. Um agente Comunicador [de Jesus et al. 2019] é uma extensão de um agente Jason, com a capacidade de se comunicar com agentes de outros SMAs externos, além de se locomover para outros SMAs, através de uma rede IoT. Um agente ARGO [Pantoja et al. 2016], por sua vez, possui a capacidade de se manifestar no meio físico, perceber informações e modificá-lo, via sensores e atuadores conectados a um microcontrolador.

Este trabalho visa explorar a aplicação dessas diferentes arquiteturas de agentes cognitivos, na construção de SMAs Embarcados, que de forma independente de ação humana, conseguem monitorar e informar em tempo real diferentes variáveis do solo e clima, alertando a população sobre a necessidade de evadirem locais em risco.

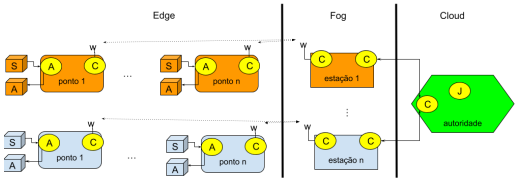
2. Monitoramento de área de risco com Sistemas Multi-agente

Este trabalho explora a utilização de Sistemas Multi-agente (SMA) em três diferentes camadas computacionais para automatizar o monitoramento de áreas de risco. Na borda computacional (*edge computing*) são utilizados SMAs Embarcados que percebem e atuam no ambiente físico. Na névoa computacional (*fog computing*) são utilizados SMAs que filtram as percepções recebidas de cada ponto de monitoramento e coordenam as ações de atuação de toda uma área de monitoramento. Por fim, na nuvem computacional (*cloud computing*) é utilizado um SMA que registra todas as ocorrências e coordena diversas áreas de monitoramento. Na Figura 1a é apresentada a arquitetura da proposta.

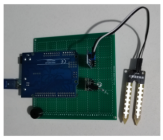
Os pontos de monitoramento estão conectados a uma rede sem fio até suas respectivas estações de monitoramento. Cada estação, por sua vez, está conectada através da internet até a autoridade responsável. No ponto de monitoramento são empregados dois agentes: um *Argo* que realiza a percepção e atuação no ambiente físico, por meio de dois sensores e atuadores acoplados em um microcontrolador, sendo eles um sensor o qual retorna um valor percentual da umidade do solo ($0 - 100\%$) e um de inclinação à base de mercúrio que fornece um valor binário ($true|false$); e um *Comunicador*, que realiza a comunicação entre o ponto de monitoramento e a próxima camada, enviando as percepções do ambiente e recebendo as ordens de atuação. Em cada estação de monitoramento são empregados dois agentes *Comunicadores*: um que gerencia a comunicação com os diversos pontos de monitoramento e outro que gerencia a comunicação com a camada superior. Por fim, na autoridade de monitoramento são empregados dois agentes: um *Comunicador* que gerencia a comunicação com todas as áreas de monitoramento e um agente *Jason* que recebe todos os eventos, decide e emite os alertas de atuação.

A lógica do trabalho proposto possui um ciclo de atuação no qual os pontos de monitoramento registram e enviam constantemente os valores observados de umidade e de inclinação do solo. Ao final da comunicação entre agentes, estas percepções são analisadas pelo agente Jason localizado na autoridade de monitoramento que, a partir de critérios previamente estabelecidos, decidirá se a percepções recebidas apresentam um cenário de normalidade; de risco de deslizamento ou de perigo iminente. Nos dois últimos casos, o mesmo fará automaticamente um chamado ao corpo de bombeiros, como retornará uma ordem ao SMA localizado na região perigosa para acionar o alarme de evacuação.

Além disso, foi construído um protótipo de *recurso* (*hardware capaz de manifestar as capacidades do agente no ambiente exógeno* [Lazarin. et al. 2023]) para ser conectado ao SMA Embarcado do ponto de monitoramento. Para atuar no ambiente foi utilizado um *buzzer* ativo. Esses sensores e atuadores foram conectados a



(a) Integração entre os diferentes SMAs.



Ações suportadas	
perigo	Ativa o *buzzer* emitindo um apito a cada 1 segundo.
critico	Ativa o *buzzer* emitindo um som de sirene.
desativar	Desativa o *buzzer*.
Percepções informadas	
umidade(N)	N é um inteiro (entre 0 e 100).
movimento(T)	T é um termo (*true* ou *false*).

(b) Recurso construído para o SMA do ponto de monitoramento.

Figura 1. Integração entre os SMAs e o recurso construído.

uma plataforma Arduino. Por fim, o firmware foi programado usando a biblioteca Javino [Lazarin and Pantoja 2015] para receber os comandos de atuação enviados pelo agente *Argo* e enviar as percepções do ambiente sobre umidade e movimento, conforme apresentado na Figura 1b.

3. Prova de conceito

Buscando avaliar o uso de agentes BDI, foi realizada uma etapa de teste. O ponto de monitoramento nesse teste foi simulado com um pequeno recipiente contendo areia e terra, onde os sensores foram inseridos. No decorrer do teste foi-se injetando água no recipiente, fazendo com que a umidade no local aumentasse. Foram também causadas vibrações manualmente no recipiente a fim de ativar o sensor de inclinação.

Para embarcar o SMA no ponto de monitoramento, foi utilizada uma Raspberry Pi Zero W (1GHz, single-core CPU, com 512MB RAM) executando o chonOS [Lazarin et al. 2023]. Neste SMA, o agente *Argo* percebe o ambiente continuamente e envia uma mensagem ao agente *Comunicador*, informando as alterações nas percepções de umidade e inclinação do solo no ambiente. Este agente, por sua vez, envia essas percepções para o SMA em execução numa estação próxima, por uma rede sem fio. Para executar o SMA da estação foi utilizada uma Raspberry Pi 3 Model B (1.2GHz, quad core, com 1GB RAM), também executando o chonOS. Neste SMA, um agente *Comunicador* lida com os diversos pontos de monitoramento na rede sem fio, recebendo todos os dados referentes à umidade e à inclinação do solo de cada ponto, e encaminha as mensagens recebidas para outro agente *Comunicador* dentro do SMA. Este agente, por sua vez, está conectado à internet e encaminha essas percepções para o agente *Comunicador* do SMA da autoridade de monitoramento, em execução na nuvem.

Para executar o SMA da autoridade de monitoramento, foi utilizado um *container* com a chonIDE [Souza de Jesus et al. 2023] instalada. Neste SMA, o agente *Comunicador* recebe e encaminha ao agente *Jason* todas as percepções recebidas. O agente *Jason* analisa as informações e toma suas decisões com base em quatro cenários: caso o agente receba um valor de umidade entre $60 - 90\%$, entenderá que o ponto de monitoramento se encontra em uma situação de perigo, e emitirá um alerta para todos os pontos de monitoramento conectados à área da qual os dados foram enviados. O segundo cenário ocorre quando o valor de umidade recebida excede os 90%, e o terceiro cenário é caso o agente receba a informação que o solo vibrou, nestes, o agente entenderá que o risco de deslizamento é iminente, então emitirá um alerta de evacuação. Por fim, o quarto cenário é aquele em que nenhuma destas condições são atendidas (umidade $\leq 60\%$ e movimento = false), neste caso, o agente não envia o comando, cancelando o alerta.

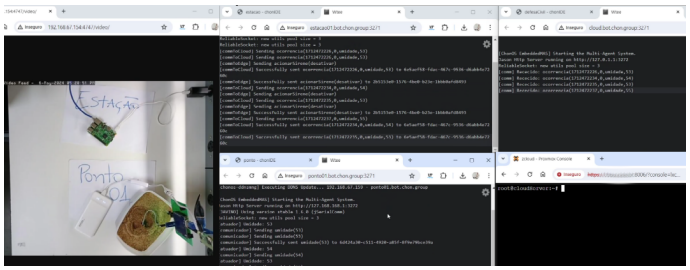


Figura 2. Aplicando Sistemas Multi-agente no monitoramento de área de risco.

No início do teste, ilustrado na Figura 2, foi-se adicionando água ao recipiente até que a umidade ultrapassasse 60%, nesse ponto o SMA da autoridade de monitoramento entendeu que a atual situação era de perigo e emitiu um alerta de perigo (intermitente) ao ponto de monitoramento. Em seguida foi causada uma vibração ao sensor de inclinação para simular uma movimentação do solo, ao receber a informação, a autoridade de monitoramento emitiu um alerta de evacuação (contínuo) para o SMA embarcado. Por fim, foi-se injetando mais água até o ponto em que a umidade ultrapassasse 90%, fazendo com que o SMA da autoridade emitisse novamente o alerta de evacuação. Os alertas foram emitidos corretamente de acordo com cada situação, fazendo do teste um sucesso. Os códigos da camada de raciocínio dos agentes e da camada de firmware do recurso construído, além do esquemático da camada de hardware e o vídeo do teste realizado, estão disponíveis na página de reprodutibilidade[1] deste trabalho.

4. Conclusão

Este trabalho apresentou e explorou a possibilidade de aplicação de SMAs Embarcados para o monitoramento de áreas de risco de deslizamento, apresentando um protótipo de baixo custo que pode ser implementado em diferentes pontos de monitoramento e integrados via estações conectadas à internet que encaminham as percepções do ambiente físico para um sistema multi-agente em uma nuvem computacional. O protótipo proposto, foi capaz de emitir alertas nos casos de alta umidade do solo e captura de movimento do solo de maneira eficaz, sendo considerado um sucesso. Ainda que a precisão dos dados de vibração seja baixa, em decorrência da ocasionalidade de descargas atmosféricas e tráfego de veículos de grande porte, os resultados foram satisfatórios demonstrando eficácia na transmissão de informações e emissão de alertas, vale ressaltar que trata-se de um projeto piloto de baixo custo.

Desse modo, trabalhos futuros podem se propor a aperfeiçoar a captação e análise de vibrações, não contendo apenas valor lógico mas sim analógico. Para tal, um giroscópio como o MPU-6050 torna-se eficaz, pois possui três eixos com seis graus de liberdade. Ademais, pluviômetros podem ser integrados às estações(utilizando sensores FD-10), de maneira a ser possível monitorar os níveis de precipitações e relacionar com as demais variáveis presentes nos pontos de monitoramento. Outrossim, mais elementos de notificações de alerta podem ser adicionados, impulsionando a comunicação com a população, à exemplo o uso de agentes Mailer [De Souza 2023], a fim de enviar e-mail e/ou SMS para que assim a população tenha ciência dos riscos por diferentes meios.

Referências

Bordini, R. H. and Vieira, R. (2003). Linguagens de Programação Orientadas a Agentes: Uma Introdução Baseada em AgentSpeak(L). *Revista de Informática Teórica e Aplicada*, 10(1):7–38.

de Jesus, V. S., Manoel, F. C. P., and Pantoja, C. E. (2019). Protocolo de Interação Entre SMA Embarcados Bio-Inspirado na Relação de Predatismo. In *Proceedings WESAAC 2019*, pages 95–106, Florianópolis.

[1] `https://papers.chon.group/WESAAC/2024/monitoriamentoAreaRisco/`

De Souza, J. P. B. (2023). *Comunicação entre SMA Embarcados: Uma Arquitetura Baseada em Protocolos da Camada de Aplicação*. Bacharelado em Sistemas de Informação, Cefet/RJ, Rio de Janeiro.

Filho, E. R. d. O., Silva, M. C. d., Abreu, M. R. d., and Júnior, L. F. M. (2024). A RELAÇÃO ENTRE A ENGENHARIA CIVIL E OS DESASTRES NATURAIS NO BRASIL. *Revista Científica Doctum Multidisciplinar*, 4(11).

Hübner, J. F., Bordini, R. H., and Vieira, R. (2004). Introdução ao desenvolvimento de sistemas multiagentes com Jason. In *Anais da XII Escola de Informática da SBC*.

Junior, I. F. d. F., Filho, G. B., and Celaschi, S. (2016). Monitoramento de Deslizamento de Terra Usando Sensor Simples de Fibra Óptica. In *SBrT2016*.

Lazarin., N., Pantoja., C., and Viterbo., J. (2023). Swapping Physical Resources at Runtime in Embedded MultiAgent Systems. In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART*. SciTePress.

Lazarin, N., Pantoja, C., and Viterbo, J. (2023). Towards a Toolkit for Teaching AI Supported by Robotic-agents: Proposal and First Impressions. In *Anais do XXXI Workshop sobre Educação em Computação*, pages 20–29, Porto Alegre, RS, Brasil. SBC.

Lazarin, N. M. and Pantoja, C. E. (2015). A Robotic-agent Platform for Embedding Software Agents Using Raspberry Pi and Arduino Boards. In *Proceedings WESAAC 2015*, pages 13–20, Niteroi. UFF.

Leão, J. C. and Souza, P. H. D. (2018). Sistema inteligente de monitoramento de deslizamento de solos. *Revista Gestão & Sustentabilidade Ambiental*, 7:508.

Marques, C. and Baesso, D. C. (2021). Desastres e vulnerabilidade na região serrana do Rio de Janeiro (RSRJ). *Ideias*, 12(00):e021019.

Oliveira, P. F., Novais, P., and Matos, P. (2021). Using Jason Framework to Develop a Multi-agent System to Manage Users and Spaces in an Adaptive Environment System. In *Ambient Intelligence – Software and Applications*, pages 137–145, Cham. Springer.

Pantoja, C. E., Jesus, V. S. d., Lazarin, N. M., and Viterbo, J. (2023). A Spin-off Version of Jason for IoT and Embedded Multi-Agent Systems. In Naldi, M. C. and Bianchi, R. A. C., editors, *Intelligent Systems*, pages 382–396, Cham. Springer Nature.

Pantoja, C. E., Stabile, M. F., Lazarin, N. M., and Sichman, J. S. (2016). ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In *Engineering Multi-Agent Systems*, pages 136–155, Cham. Springer.

Souza de Jesus, V., Mori Lazarin, N., Pantoja, C. E., Vaz Alves, G., Ramos Alves de Lima, G., and Viterbo, J. (2023). An IDE to Support the Development of Embedded Multi-Agent Systems. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*, pages 346–358, Cham. Springer.

Vachtsevanou, D., William, J., dos Santos, M. M., de Brito, M., Hübner, J. F., Mayer, S., and Gomez, A. (2023). Embedding Autonomous Agents into Low-Power Wireless Sensor Networks. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*, Cham. Springer.

Reconhecimento facial para detecção de Doença Renal Crônica

João Victor Texeira Degelo[1], Johan Su Kwok[1], Gabriel Zambelli Scalabrini[1], Anarosa Alves Franco Brandão[1], Rogério da Hora Passos[2], Maristela Carvalho da Costa[3]

[1]Departamento de Engenharia de Computação e Sistemas Digitais
Universidade de São Paulo (USP)
Caixa Postal 61.548 – 05508-010 – São Paulo – SP – Brazil

[2]Davita Tratamento Renal e Hospital Israelita Albert Einstein
São Paulo – SP – Brazil

[3]Instituto do Coração e Hospital Santa Catarina
São Paulo – SP – Brazil

{degelo, johansukwok, gabrielzsb.comp, anarosa.brandao}@usp.br,

Abstract. *Chronic Kidney Disease (CKD) affects a huge number of people along the world. Only considering the Brazilian landscape, it is estimated that over 10 milion Brazilian citizens have CKD. Among them, 90 thousands are in use of dialisis treatment. Therefore, CKD impacts the society in both ways: as a public healthy problem and as an economics problem. It is commom sense among physicians that the faces of CKD patients have common characteristics. In this way, recognizing such characteristics could help physicians in diagnosing CKD in an early stage. Nevertheless, there is a gap in medical research concerning the diagnosis of CKD using multiagent systems combined with facial recognition, an accessible and non-invasive method. This article aims to develop a multiagent system designed to assist in diagnosing CKD by using patient images and health information as input.*

Resumo. *A Doença Renal Crônica (DRC) afeta um grande número de pessoas em todo o mundo. Considerando apenas o cenário brasileiro, estima-se que mais de 10 milhões de cidadãos brasileiros tenham DRC. Entre eles, 90 mil estão em tratamento dialítico. Portanto, a DRC impacta a sociedade de duas maneiras: como problema de saúde pública e como problema econômico. É senso comum entre os médicos que os rostos dos pacientes com DRC possuem características comuns. Dessa forma, reconhecer tais características poderia auxiliar os médicos no diagnóstico precoce da DRC. No entanto, existe uma lacuna na investigação médica relativa ao diagnóstico da DRC utilizando sistemas multiagentes combinados com técnicas de inteligência artificial combinadas com reconhecimento facial, um método acessível e não invasivo. Este artigo tem como objetivo desenvolver um sistema multiagente projetado para auxiliar no diagnóstico de DRC usando imagens de pacientes e informações de saúde como entrada.*

1. Introdução

A Doença Renal Crônica (DRC) é uma condição de longa duração que afeta os rins, responsáveis pela filtragem do sangue, remoção de resíduos metabólicos e impurezas, e regulação do equilíbrio de fluidos e pH do corpo. A DRC é caracterizada pela perda gradual e irreversível da função renal ao longo do tempo. Estima-se que aproximadamente dez milhões de brasileiros tenham a doença, dos quais cerca de 90 mil estão em diálise, estágio terminal da doença [Ministério da Saúde 2019]. Os gastos com o tratamento da DRC e de outras condições relacionadas correspondem a 12,97% das despesas em saúde [Alcalde and Kirsztajn 2018].

As doenças, em geral, não afetam apenas internamente o corpo, mas também podem se refletir em características faciais dos pacientes, que podem servir como indicadores diagnósticos, especialmente para condições endócrinas, metabólicas, genéticas e neuromusculares. Embora o desenvolvimento da tecnologia de reconhecimento facial tenha sido gradual, sua aplicação na medicina clínica tem crescido rapidamente nos últimos dez anos [Qiang et al. 2022]. No caso da DRC, a presença prolongada de resíduos metabólicos no sangue pode gerar alterações em diversos órgãos, incluindo a pele, e o rosto [Malkina 2023, Lupi et al. 2011]. Entre essas alterações, destacam-se:

- Mudanças na coloração da pele [AAD 2019]: O acúmulo de toxinas no corpo, devido à falha na filtragem pelos rins, pode resultar em alterações na coloração da pele, como manchas escuras, tonalidade cinza, cistos, manchas semelhantes a espinhas ou pele amarelada, com inchaços e linhas profundas.
- Icterícia [Levine 2022]: A partir do estágio 4 da DRC, pode ocorrer icterícia, incluindo o amarelamento do branco dos olhos.
- Inchaço facial [AAD 2019]: Com a disfunção renal, a remoção de fluidos e sais é prejudicada, resultando em inchaços pelo corpo, incluindo a face.
- Xerose [AAD 2019]: Em casos avançados de DRC, que exigem diálise e transplante renal, a pele do paciente pode tornar-se excessivamente seca, desenvolvendo rachaduras e escamas.

A DRC também pode afetar a aparência das mãos e dos pés [AAD 2019], apresentando, além do inchaço e da presença de bolhas:

- Coloração branca na parte superior das unhas e coloração normal ou marrom avermelhada na parte inferior (unhas de Lindsay ou meio-e-meio).
- Unhas pálidas.
- Faixas brancas atravessando uma ou mais unhas (unhas de Muehrcke).

O objetivo deste artigo é desenvolver uma solução multiagente inteligente que será implementada na forma de aplicativo, no qual médicos nefrologistas possam utilizar imagens dos pacientes e outras informações de saúde para calcular a probabilidade de ocorrência de Doença Renal Crônica. O diagnóstico será realizado por um sistema multiagente inteligente, treinado através de algoritmos de visão computacional. Esse sistema receberá como entrada imagens das faces e das mãos dos pacientes, bem como informações sobre seu estado de saúde, buscará nas imagens fenótipos relacionados à DRC, e informará como saída a probabilidade deste paciente possuir a doença. Esta solução apresenta uma abordagem acessível e não invasiva para a detecção de DRC, tendo potencial de melhorar significativamente o processo de detecção e auxílio ao diagnóstico, especialmente em locais com alta incidência de DRC ou com escassez de médicos especialistas.

2. Trabalhos relacionados

Inicialmente foi realizado um estudo na literatura para identificar abordagens que utilizam soluções inteligentes para apoiar a detecção de Doença Renal Crônica. Em [Kumar et al. 2023], foi adotado o uso de processamento de imagem em exames de ultrassom dos rins para apoio ao diagnóstico de DRC. Já [Divya et al. 2021] utiliza imagens da íris de pacientes para treinar um modelo de aprendizado de máquina que prediz doenças renais. Ainda, [Dashtban et al. 2023] emprega dados de prontuários eletrônicos de 350.067 indivíduos para identificar subtipos de Doença Renal Crônica com o auxílio de algoritmos de inteligência artificial.

Uma revisão da literatura sobre uso de reconhecimento facial para diagnóstico de doenças [Qiang et al. 2022] destaca práticas recomendadas para a captura de fotos, incluindo padronização da expressão facial dos pacientes, posicionamento da câmera e condições ambientais. Além disso, são apresentados os principais algoritmos utilizados no diagnóstico e as circunstâncias em que cada um deles é mais apropriado. Apesar da extensa literatura sobre o tema, não foram encontradas abordagens que utilizem reconhecimento facial diretamente na detecção de Doença Renal Crônica. Por conta disso, foi realizado um estudo mais aprofundado para entender melhor a aplicação de modelos de reconhecimento facial na detecção de doenças.

Em outra revisão da literatura [Wu et al. 2021], foram compilados 141 estudos que utilizaram reconhecimento facial para o diagnóstico de doenças. Também foi realizada uma análise estatística sobre os fatores que influenciam a eficiência dos modelos de inteligência artificial. Os autores definem o conceito de Facial Recognition Intensity (FRI), um indicador que descreve a diferença entre as feições faciais da doença estudada e as do grupo de controle. O FRI é calculado pela fórmula $N_f * P_{max}$, onde N_f representa o número de fenótipos faciais relevantes para a doença, ou seja, o número de características faciais afetadas pela doença, e P_{max} representa a penetração máxima dessas características, isto é, o percentual máximo de indivíduos acometidos pela doença que apresentam essa característica. O estudo identifica uma forte correlação entre o FRI e a eficiência do modelo. Entretanto, fatores como resolução da imagem utilizada no treino, tamanho do dataset utilizado para treino e o algoritmo de inteligência artificial utilizado não apresentam uma correlação tão significativa. Essas informações são de extrema relevância, pois impactam diretamente a abordagem que deve ser adotada neste projeto.

3. Metodologia do projeto

Na etapa inicial do projeto, foi elaborada a documentação para submissão ao Comitê de Ética da EACH-USP, com o objetivo de viabilizar a coleta de dados. Após a autorização do Comitê de Ética, a coleta de dados dos pacientes com Doença Renal Crônica será iniciada. Essa coleta ocorrerá na Clinefro - Clínica de Nefrologia de Juazeiro, onde serão registradas fotografias faciais e das mãos dos pacientes, além do preenchimento de um questionário para obter informações sobre a saúde dos pacientes, como presença ou histórico familiar de doenças como anemia, diabetes, hipertensão ou doença renal, e sobre seus hábitos, como consumo de água, alimentos ricos em sódio ou gorduras, prática de atividade física, e consumo alcoólico e tabagismo. Ao final, estes mesmos dados serão coletados de um grupo controle, que não possui a DRC. A presença de um grupo ode controle é essencial para o bom treinamento de um algoritmo de aprendizado supervisionado, visto que diminui possíveis vieses que possam surgir.

Serão realizadas diversas análises nos dados coletados pelo Agente de Diagnóstico com o objetivo de entender com maior profundidade o grupo participante da pesquisa, além de identificar eventuais dados desbalanceados que podem impactar o modelo desenvolvido. Após análise e tratamento dos dados, eles serão usados para alimentar o modelo de aprendizado no Agente de Aprendizado.

Serão inicialmente testados diversos modelos e algoritmos, como por exemplo: extração de características do rosto e das mãos em conjunto de um modelo XGBoost, utilização de Redes Neurais Convolucionais, e utilização de modelos pré treinados como FaceNet. Durante esta etapa, serão utilizadas técnicas de validação cruzada para selecionar os modelos com melhor desempenho. Com a solução final definida, seu desempenho será avaliado no conjunto de teste, buscando entender os aspectos que mais impactaram no diagnóstico.

Por fim, será desenvolvido um aplicativo móvel para disponibilizar a solução inteligente aos médicos nefrologistas. Os usuários poderão submeter as informações do paciente, junto de suas imagens, para obter em tempo real um diagnóstico de doença renal crônica a partir dessas informações por meio do Agente de Diagnóstico.

4. Arquitetura

O sistema será composto por três agentes (Agente de Diagnóstico, Agente Coletor de Dados e Agente de Aprendizado), um banco de dados e o aplicativo. As comunicações serão realizadas por meio de requisições HTTPS através de uma API protegida contra acesso indevido. A arquitetura está representada no diagrama na Figura 1.

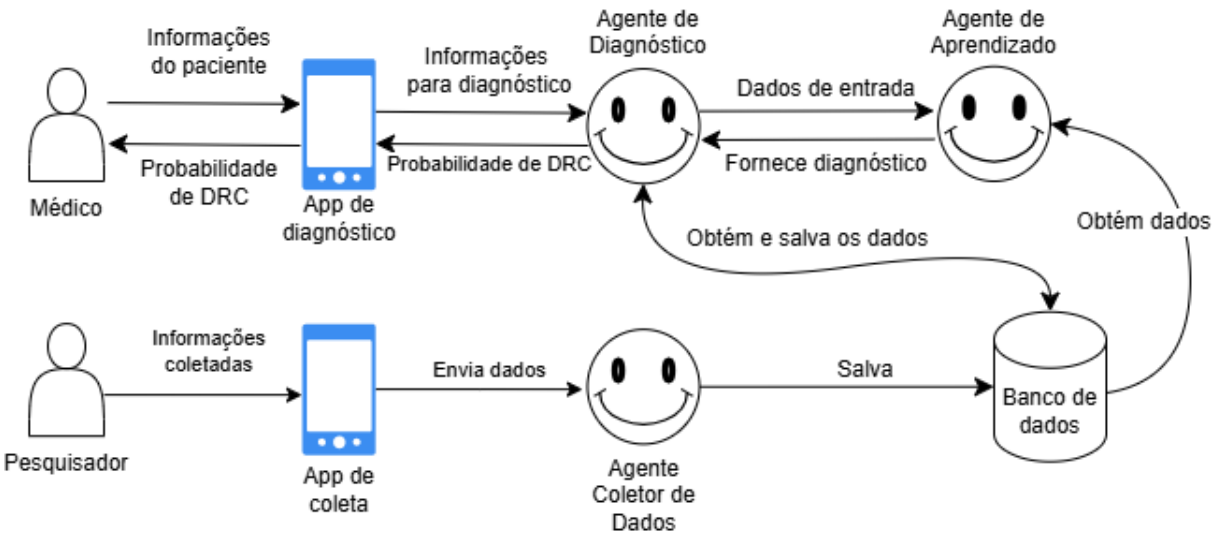


Figura 1. Diagrama de arquitetura do sistema multiagentes

Na etapa inicial de coleta de dados, pesquisadores, enfermeiros e médicos coletarão informações sobre a saúde e os hábitos dos pacientes descritas anteriormente. O Agente Coletor de Dados realizará o tratamento dessas informações e as salvará em um banco de dados na nuvem.

Em seguida, o Agente de Aprendizado obterá esse conjunto de fotos e informações

dos pacientes e, através de algoritmos de aprendizado de máquina e reconhecimento facial, treinará um modelo, que fornecerá a probabilidade do paciente possuir doença renal crônica com base nessas informações. Esse modelo terá como principais métricas a acurácia e o recall, sendo o objetivo obter um modelo final com baixo número de falso negativos.

O Agente de Diagnóstico, por sua vez, receberá do aplicativo de diagnóstico as mesmas informações dos pacientes mencionadas anteriormente, realizará o tratamento desses dados, salvará no banco de dados e fornecerá essas informações ao Agente de Aprendizado, que retornará a resposta ao aplicativo dos médicos.

5. Implementando a arquitetura

A arquitetura proposta será implementada na forma de um aplicativo destinado aos profissionais de saúde para diagnóstico de Doença Renal Crônica. Para demonstrar as funcionalidades desse aplicativo, foi desenvolvido um mockup inicial. A Figura 2a representa a tela para a qual o profissional de saúde é redirecionado após preencher seu usuário e senha. Nessa tela, é possível cadastrar novos pacientes, visualizar os pacientes já cadastrados ou buscar por uma pessoa específica.

Ao selecionar um paciente, o profissional de saúde é redirecionado para a tela representada na Figura 2b, que contém as informações pessoais do paciente e seu histórico de diagnósticos, como ilustrado na Figura 2c, utilizando o modelo de reconhecimento facial desenvolvido. O sistema permite a alteração das informações cadastradas do paciente ou a iniciação de um novo diagnóstico utilizando fotografias.

Ao buscar mais detalhes sobre um diagnóstico específico na aba ”Histórico”, o usuário é redirecionado para as telas representadas nas Figuras 2d e 2e. Nessas telas, é possível observar as informações sobre os hábitos do paciente, bem como as imagens da face e das mãos que foram utilizadas pelo modelo para gerar a probabilidade final.

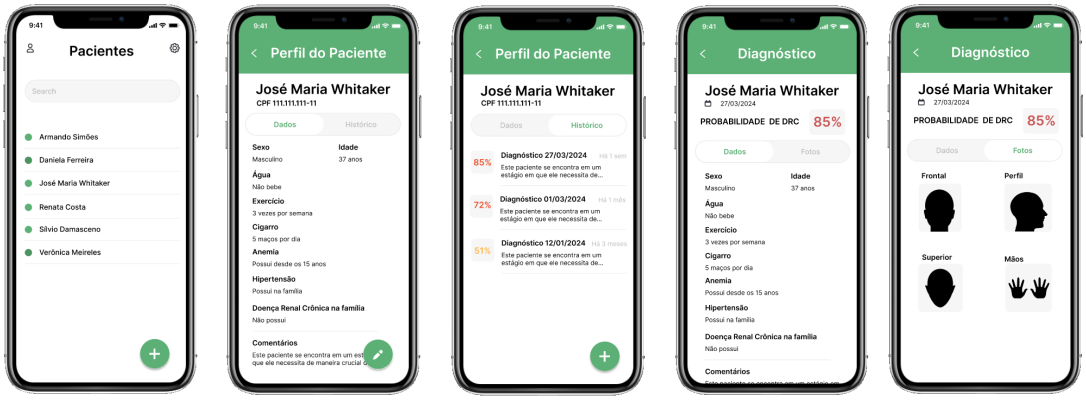


Figura 2. Telas de (a) lista de pacientes, (b) perfil do paciente, (c) histórico do paciente, (d) diagnóstico e (e) fotos do paciente

6. Conclusão

A revisão da literatura revelou que, embora existam diversos estudos que utilizam inteligência artificial para o diagnóstico de DRC, nenhum deles explora o potencial do

reconhecimento facial. Os resultados esperados podem preencher essa lacuna, proporcionando uma ferramenta eficiente e de fácil utilização para auxiliar pacientes em regiões de recursos escassos, e a identificação precoce da doença.

O projeto envolve a coleta de dados detalhados de pacientes com DRC, por meio de um aplicativo. Esses dados serão utilizados para treinar e validar o modelo de IA dentro de um sistema multiagente. A arquitetura proposta assegura a segurança e integridade dos dados, enquanto o aplicativo desenvolvido permitirá que os profissionais de saúde utilizem a tecnologia de maneira prática e eficiente.

Este estudo não apenas contribui para o avanço tecnológico no diagnóstico da DRC, mas também tem o potencial de melhorar a qualidade de vida dos pacientes ao possibilitar diagnósticos mais rápidos e precisos. Futuramente, a expansão do uso de reconhecimento facial para outras condições médicas pode transformar a prática médica, tornando-a mais preventiva e personalizada.

Referências

AAD (2019). *Kidney disease: 11 ways it can affect your skin*. American Academy of Dermathology Association.

Alcalde, P. R. and Kirsztajn, G. M. (2018). Expenses of the brazilian public healthcare system with chronic kidney disease. *Brazilian Journal of Nephrology*.

Dashtban, A. et al. (2023). Identifying subtypes of chronic kidney disease with machine learning: development, internal validation and prognostic validation using linked electronic health records in 350,067 individuals. *eBioMedicine*.

Divya, C. D. et al. (2021). An efficient machine learning approach to nephrology through iris recognition. *eBioMedicine*.

Kumar, K. et al. (2023). A deep learning approach for kidney disease recognition and prediction through image processing. *Applied Sciences*.

Levine, H. (2022). The 5 stages of kidney disease. Health.

Lupi, O. et al. (2011). Manifestações cutâneas na doença renal terminal. *Anais Brasileiros De Dermatologia*, 86(2):319–326.

Malkina, A. (2023). Doença renal crônica. Manual MSD.

Ministério da Saúde (2019). *14/3 – Dia Mundial do Rim 2019: Saúde dos Rins Para Todos*. Ministério da Saúde, Ceará.

Qiang, J. et al. (2022). Review on facial-recognition-based applications in disease diagnosis. *National Library of Medicine*.

Wu, D. et al. (2021). Facial recognition intensity in disease diagnosis using automatic facial recognition. *National Library of Medicine*.

Agent-Based Modeling and Simulation for Integrating Social and Natural Dynamics in Water Resources Management

D. Sousa[1,2], **C. Coelho**[3], **C. Alves**[1], **C. Ralha**[3]

[1]Department of Civil and Environmental Engineering – University of Brasília
Zip Code 70.297-400, Brasília, Brazil

[2]Ryan Hanley Consulting Engineers – D01W596, Dublin, Ireland

[3]Computer Science Department – Institute of Exact Sciences – University of Brasília
Zip Code 70.904-970, Brasília, Brazil

{deborahsousa.eng,cassiocouto88,cmaalves,celiaghedini}@gmail.com

Abstract. *Water resources modelling based solely in natural dynamics may presented limitations due to neglection of social factors. This article presents an exploratory study of agent-based modeling and simulation for managing water use conflicts between farmers and regulators and among farmers. Water users' behavior is guided by the Belief, Desire, and Intention (BDI) reasoning model, aiming at welfare, often through crop sales and income generation. Heterogeneous behavior is captured through diverse cooperation profiles linked to different compliance levels with water use rules set by the regulatory agent. The resulting framework offers insights into environmental flows and water consumption equity among users, testing opportunities for new water policies.*

1. Introduction

The allocation of water resources has evolved toward sustaining communities by ensuring the water availability for human-related uses (e.g., drinking water for human consumption, navigation, irrigation, industrial processes), as well as for natural purposes (e.g., environmental flow, biodiversity preservation, nutrients, and hydrological cycle).

Thus, the management of water resources shall involve methodologies for comprehending the hydrology cycle and its interaction with society. Modeling is one of those methodologies applied to support decision-making on water allocation. In particular, it enables assessments under broad spatial and temporal scales of particularly complex systems, such as the hydrologic, social, and economic systems.

Therefore, using models for water resource investigations and social systems has become increasingly relevant [Blair and Buytaert 2016]. As a result, incorporating sociological aspects into water systems modeling has been the subject of an increasing number of scientific investigations, aligned with the emergence of transdisciplinary fields such as socio-hydrology [Sivapalan et al. 2012] and socio-hydrogeology [Re 2015].

Agent-Based Modeling and Simulation (ABMS) is a commonly used approach to represent human behavior in water-related contexts, and in particular, in applications

grounded in the socio-hydrological concept [Al-Amin et al. 2018, Pouladi et al. 2019, Du et al. 2023, Sousa et al. 2024]. Moreover, the repercussion of the agent-based approach is partly due to the development of increasingly resourceful and user-friendly ABMS implementation and simulation platforms, which enables the development of ABMS by users who are not necessarily specialized in computer science [Taillandier et al. 2019]. In the context of environmental applications, the use of free ABMS platforms such as NetLogo [Tisue and Wilensky 2004], CORMAS [Le Page et al. 2000], and GAMA [Taillandier et al. 2019] have been extensively used.

The use of the Belief-Desire-Intention (BDI) agent reasoning model [Bratman 1987] has been widely used in ABMS representations [Kock 2008, Taillandier et al. 2012, Liang et al. 2016]. In particular, the BDI model can be advantageous for water resources-related applications. The complexity of decision-making processes similar to human decision-making can be represented in a more computationally tractable structure [Kock 2008].

In this context, this work presents an exploratory study on the role of ABMS in advancing the management of water resources, particularly through the explicit representation of decisions of water users and regulators toward conflict resolution. The rest of the paper is organized as follows. Section 2 presents the applied methodology, Section 3 the applied model results, and Section 4 the conclusion with future work.

2. Methodology

The conception of the agent-based model was outlined using the TROPOS methodology from the agent-oriented software engineering area, developed by [Bresciani et al. 2004]. The TROPOS diagrams were delineated using the piStar tool [Pimentel and Castro 2018]. A legend for the basic elements used is provided in Figure 1. More methodology details can be found in [Sousa 2023] and in the GitHub project's repository[1].

Figure 1 presents the agent-based model, including the main actors of the problem in the Early Requirements stage of TROPOS, along with their objectives, resources, and basic tasks. Two agent types represent the dynamic of water management: water users and a regulatory entity. A local regulatory authority agent and a farmer agent are instances of these agent types, illustrating the water use for irrigated crop production.

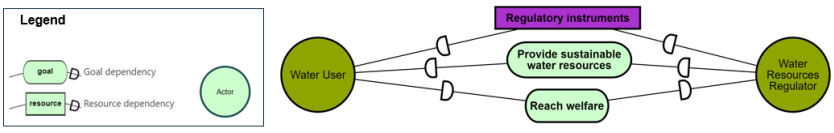


Figura 1. TROPOS early requirements diagram.

Farmer agents are human-based decision-makers according to the BDI paradigm. Water users' main goal is to reach welfare, which in the case of the instantiated farmer agent, may be achieved by selling their crop production and generating income. Heterogeneous behavior is taken into account, considering that each farmer agent acts according

[1] https://github.com/deborahsousa/Coupled-ABM-Hydrol-Formoso

to a profile related to compliance with water use rules issued by the local regulatory agent, whose goal is to provide sustainable water resources to users.

The present framework assumption is that a farmer's water withdrawal decision may be influenced by four beliefs: cooperation profile, water use rules, water available for use, and neighborhood effect. Each belief leads to the desire to comply or not with any restriction or attention rule that has entered into force and culminates in the intention (or not) of consuming water through the activation of their irrigation pumps. The local water regulator defines their management strategies based on a set of rules to be issued according to the season and the water availability in the region.

Regarding the implementation perspective, the GAMA Platform (version 1.8.2) and its built-in BDI plugin (*simple_bdi*) were used to develop the agent-based model since the geographic location of water users plays a crucial role, with upstream and downstream users having different impacts on the model. Moreover, GAMA was chosen due to its diverse and extensive applications, including decision support systems, urban systems, urban mobility, epidemiology, adaptation to climate change, and disaster management. GAMA uses its own high-level, agent-oriented language, the GAma Modeling Language (GAML).

3. Results and Discussion

The model is applied to the Formoso river basin situated in the state of Tocantins in the north of Brazil (Figure 2). The basin is characterized by intensive agriculture and water scarcity, with drought events in recent years. As a result, a drought mitigation regulatory plan was established through restriction and attention rules applied to farmers' irrigation pumps [IAC 2018]. As part of this mitigation plan, an online dataset was launched. The GAN Platform[2] is a pioneer online water management tool in Brazil as it displays both water availability and demand data for each pump in the Formoso river basin.

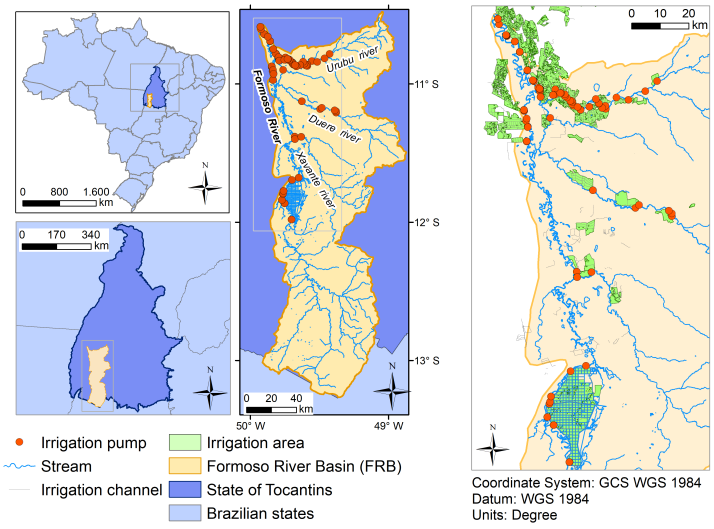


Figura 2. Geographical context of the Formoso river basin.

The application of agent-based models in the Formoso river basin has been explored in other works [da Silva et al. 2023, Sousa et al. 2024]. As presented in

[2]https://gan.iacuft.org.br/monitoramento-mapa

[Sousa et al. 2024], we consider heterogeneous behavior for farmer agents categorized according to three profiles: Cooperative (CP), Intermediate (CI), and Non-Cooperative (NC). This categorization focuses on the willingness to cooperate on the water consumption data transmission to the GAN platform. Each profile is linked to a compliance level about water usage rules issued by the regulatory agent.

The geographical set in GAMA of the irrigation pumps, water network, and irrigable lands are presented in Figure 3, as well as the results of the total daily water volume consumed by all farmers and in each profile (CP, CI, and NC). Note the pattern of water consumption is higher for NC farmers than for CP farmers. These results implications can be evaluated according to diverse metrics such as economic (overall basin income), ecologic (environmental flows), and social (income equity) variables.

A proposal for advancing water resources management toward conflict resolution is to integrate the ABMS results of water withdrawal into a hydrological model as presented in Figure 4.Additionally, the integration of role-playing games (RPG) and ABMS may enhance conflicts resolution of natural resources management [Adamatti et al. 2005].

Finally, we conclude that this agent-based framework advances water resources management by enabling the simulation of scenarios of different water policies and water use rules and including the explicit representation of water user decisions, as well as the compliance evaluation, surveillance, and application of financial penalties.

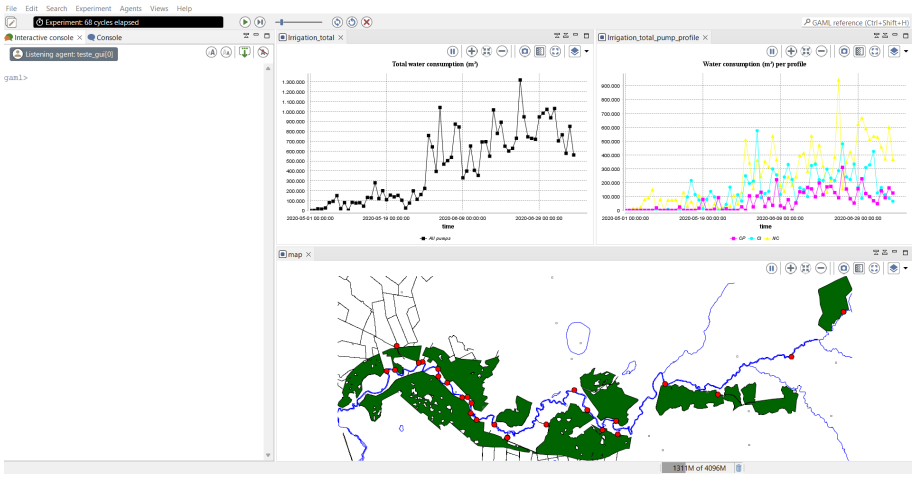


Figura 3. Graphical user interface visualization of the model in GAMA platform.

4. Conclusion

This work presents an exploratory study for advancing water resources management based on an increasingly trend observed in the literature of integrating social beahviour into natural dynamics. The presented approach uses ABMS to represent the behavior of actors of relevance in water-related issues. The main contribution of this work is providing a framework of agent-based coupled with hydrological modeling, which enables the evaluation of the importance of practical cooperativeness among users through collective compliance with usage rules. Additionally, the framework can be used by policy-makers when making decisions to safeguard sufficient water for farmers and other instances of users. The extension of the presented framework may happen through the simulation of additional scenarios, such as with an active and proactive role by the regulatory entity.

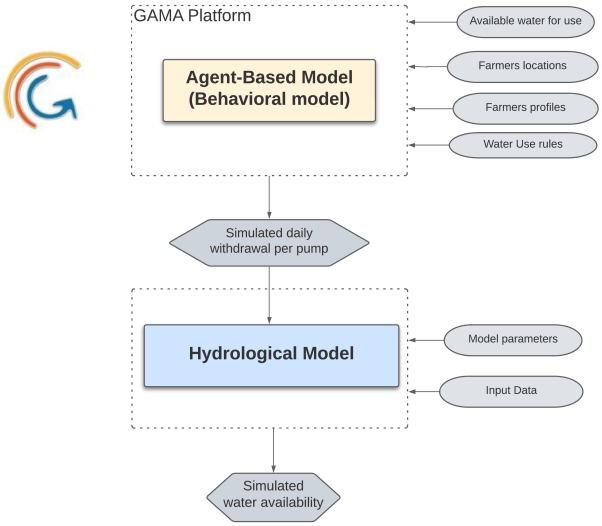


Figura 4. Proposal of coupling ABMS and hydrological models.

Acknowledgements

The authors are grateful for the comments and suggestions received by the two anonymous reviewers. The first author (D.S.) thanks funding received from the Coordination for the Improvement of Higher Education Personnel (CAPES) (grants 88887.144867/2017-00 and 88887.495084/2020-00). Prof. Célia G. Ralha thanks the research productivity grant number 309688/2021-3 in the Computer Science area from the Brazilian National Council for Scientific and Technological Development (CNPq). Opinions and findings are those of the authors and do not necessarily reflect the views of the funding agencies.

Referências

Adamatti, D., Sichman, J., Rabak, C., Bommel, P., Ducrot, R., and Camargo, M. (2005). Jogoman: A prototype using multi-agent-based simulation and role-playing games in water management. In *Proceedings of the SMAGET-CABM-HEMA Conference*.

Al-Amin, S., Berglund, E. Z., Mahinthakumar, G., and Larson, K. L. (2018). Assessing the effects of water restrictions on socio-hydrologic resilience for shared groundwater systems. *Journal of Hydrology*, 566:872–885.

Blair, P. and Buytaert, W. (2016). Socio-hydrological modelling: a review asking ”why, what and how?”. *Hydrology and Earth System Sciences*, 20:443–478.

Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*, volume xii.

Bresciani, P., Perini, A., Giorgini, P., and Giunchiglia, F. (2004). Tropos: An agent-oriented software development methodology. *AAMAS*, 8:203–236.

da Silva, E. P., de Sousa, D. S., Minoti, R. T., de Maria Albuquerque Alves, C., and Vergara, F. E. (2023). Modelagem sócio-hidrológica para avaliação do efeito comportamental cooperativo em sistemas hídricos: Uma aplicação à bacia do rio formoso, tocantins. In *XXV SBRH - Simpósio Brasileiro de Recursos Hídricos*.

Du, E., Wu, F., Jiang, H., Guo, N., Tian, Y., and C., Z. (2023). Development of an integrated socio-hydrological modeling framework for assessing the impacts of shelter

location arrangement and human behaviors on flood evacuation processes. *Hydrology and Earth System Science*, 27:1607–1626.

IAC (2018). *Gestão de Alto Nível: Plano do Biênio (2018-2019). Instituto de Atenção às Cidades (IAC). Universidade Federal do Tocantins (IFT)*. Disponível em: https://central.to.gov.br/download/43038. Acessado em: 15/05/24.

Kock, B. E. (2008). Agent-based models of socio-hydrological systems for exploring the institutional dynamics of water resources conflict. Department of Civil and Environmental Engineering, Massachusetts Institute of Technology. Cambridge, MA, EUA.

Le Page, C., Bousquet, F., Bakam, I., Bah, A., and Baron, C. (2000). Cormas: A multiagent simulation toolkit to modelnatural and social dynamics at multiple scales. In *Proceedings of Workshop "The ecology of scales"*.

Liang, X., Chen, H., Wang, Y., and Song, S. (2016). Design and application of a CA-BDI model to determine farmers' land-use behavior. *SpringerPlus*, 5(1):1581.

Pimentel, J. and Castro, J. (2018). piStar Tool – a pluggable online tool for goal modeling. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 498–499.

Pouladi, P., Afshar, A., Afshar, M. H., Molajou, A., and Farahmand, H. (2019). Agent-based socio-hydrological modeling for restoration of urmia lake: Application of theory of planned behavior. *Journal of Hydrology*, 576:736–748.

Re, V. (2015). Incorporating the social dimension into hydrogeochemical investigations for rural development: the bir al-nas approach for socio-hydrogeology. *Hydrogeology Journal*, 23:1293–1304.

Sivapalan, M., Savenije, H. H. G., and Blöschl, G. (2012). Socio-hydrology: A new science of people and water. *Hydrological Processes*, 26(8):1270–1276.

Sousa, D., Alves, C., Vergara, F., Coelho, C., and Ralha, C. (2024). Agent-based modelling for representing water allocation methodologies in the irrigation system of the formoso river basin, brazil. *Proceedings of IAHS 2022*, 385:71–77.

Sousa, D. S. (2023). Modelagem baseada em agentes acoplada à modelagem hidrológica para avaliação de estratégias coletivas de alocação de água: o caso do rio urubu (to). Dissertação de Mestrado. Programa de Pós-Graduação em Tecnologia Ambiental e Recursos Hídricos, Universidade de Brasília. Brasília, DF, Brasil.

Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.-N., Marilleau, N., Caillou, P., Philippon, D., and Drogoul, A. (2019). Building, composing and experimenting complex spatial models with the gama platform. *GeoInformatica*, 23(2):299–322.

Taillandier, P., Thérond, O., and Gaudou, B. (2012). A new BDI agent architecture based on the belief theory. Application to the modelling of cropping plan decision-making. *Int. Environmental Modelling and Software Society (iEMSs). ffhal-00714325f.*

Tisue, S. and Wilensky, U. (2004). Netlogo: Design and implementation of a multi-agent modeling environment. In *SwarmFest 2004*.

Investigação de Modelos Organizacionais para a Criação de Máquinas Éticas

Tielle da Silva Alexandre [1], **Carlos Eduardo Pantoja** [2], **Flávia Cristina Bernadini** [1]

[1]Instituto de Computação - Universidade Federal Fluminense - Niterói, RJ – Brasil

[2]Centro Federal de Educação Tecnológica (Cefet/RJ) Rio de Janeiro, RJ – Brasil

tiellesa@id.uff.br

Abstract. *A Inteligência Artificial desempenha um papel importante na transformação de cidades em ambientes inteligentes, possibilitando o desenvolvimento de novas tecnologias integradas ou substitutivas, como veículos autônomos, embora levante questões éticas e de responsabilidade. Os Sistemas Multiagentes consistem em agentes autônomos que cooperam e competem para resolver problemas coletivamente, com objetivos individuais e coletivos, semelhante à organização social humana. Máquinas Éticas, que seguem princípios e normas éticas, podem ser implementadas usando SMA, já que esses sistemas são formados por agentes que podem estar sob a mesma sociedade compartilhando regras e normas em prol da sociedade em si. Esses sistemas, com sua estrutura social análoga à humana, são candidatos para a criação e avaliação dessas máquinas éticas. O objetivo deste trabalho é investigar modelos organizacionais para criação máquinas éticas usando SMA embarcados.*

1. Introdução

A Internet das Coisas (IoT) é uma rede que conecta objetos na Internet, possibilitando o compartilhamento de dados e controle dos dispositivos [Management Association 2017]. O avanço da IoT permitirá a transformação das cidades em ambientes inteligentes dotados de soluções tecnológicas que melhoram a qualidade de vida de seus habitantes. Segundo [Albino et al. 2015], uma cidade inteligente poderia ser definida como uma cidade que fornece tecnologias integradas aos seus cidadãos para que tenham qualidade de vida. Por exemplo, uma cidade inteligente pode ser equipada com uma rede que conecta dispositivos de IoT e permite a troca de informações entre eles. Considerando a interconectividade de tais dispositivos, outras soluções poderão ser integradas às cidades inteligentes, onde os semáforos poderão ser autônomos priorizando a passagem do transporte público e dos pedestres, assim como um sistema de segurança inteligente poderá monitorar a cidade vinte e quatro horas por dia, acionando os serviços de segurança ou emergência quando um incidente for detectado [Boden 1996, Nalini 2019].

Nesse cenário, a Inteligência Artificial (IA) pode desempenhar um papel importante na transformação das cidades, possibilitando o desenvolvimento de novas tecnologias e soluções que podem ser integradas às soluções existentes ou até mesmo substituí-las. A presença de veículos autônomos no trânsito já é uma realidade em diversas cidades,

como em São Francisco, na Califórnia [Allen et al. 2006]. No entanto, questões sobre responsabilidade em casos de acidentes e outros dilemas, como a possibilidade de os veículos autônomos violarem as regras de trânsito para evitar um acidente, ainda estão em aberto [Metz 2019, Paulo 2023]. Isso ressalta a importância de que o desenvolvimento de soluções com IA seja acompanhado por discussões contínuas sobre ética e moralidade, garantindo que essas tecnologias sejam implementadas de forma responsável.

Concomitantemente, nos últimos 70 anos, a IA vem sendo desenvolvida acadêmica e industrialmente, tendo diversas definições ao longo da história. Por exemplo, em sistemas de IA, agentes podem ser projetados para realizar tarefas como reconhecimento de padrões, tomada de decisões, processamento de linguagem natural, controle de sistemas autônomos, entre outras. Neste trabalho, adotamos o grupo de definições de agentes inteligentes que consideram que todo sistema computacional e inteligente pode ser visto como um agente inteligente [Russell and Norvig 2016] e que um agente inteligente é uma entidade racional lógica ou física dotado de autonomia e proatividade que está situado em um ambiente ao qual interage e modifica a fim de atingir um objetivo [Weiss 1999].

Em um Sistema Multiagentes (SMA), agentes podem estar organizados em grupos de agentes autônomos que cooperam e competem para a resolução de problemas que não podem ser resolvidos de forma isolada [Briot and Demazeau 2001]. Assim, todo agente possui objetivos individuais e também objetivos coletivos pertencentes à sua sociedade. Dessa forma, uma organização estabelece restrições aos comportamentos dos agentes estabelecendo um comportamento grupal coeso. Tais organizações de agentes podem ser vistas como uma organização societal de agentes, que pode ser comparada à organização social dos seres humanos, que é estruturada e guiada por leis, normas e princípios éticos e morais. Por exemplo, uma universidade possui grupos de pessoas com diferentes papéis (e.g., professor, aluno e secretários) e com diferentes objetivos pessoais e profissionais, que podem ou não estar alinhados aos objetivos organizacionais da universidade. Tais grupos devem executar suas atividades na organização (universidade) respeitando as normas e o código de ética da sociedade na qual a organização está inserida,

Além disso, a organização também pode especificar um código de ética específico para sua comunidade (e.g., código de ética do servidor ou do aluno). A universidade cumpre seu propósito a partir do comportamento coeso de grupos de pessoas, cujos comportamentos individuais são regidos por um conjunto de regras. Segundo [Nalini 2019], a ética pode ser definida como um sistema de princípios, regras ou diretrizes que regem os comportamentos ou ações de um indivíduo ou grupo de indivíduos. Já o campo que estuda os valores e princípios éticos ou morais para que uma IA se comporte de maneira ética pode ser denominado de IA Ética ou Máquina Ética [Allen et al. 2006]. Considerando a analogia dos SMAs com a organização social humana, estes se tornam candidatos para implementação e avaliação de Máquinas Éticas. O objetivo deste trabalho é discutir uma proposta de investigação de modelos organizacionais para criação de máquinas éticas utilizando SMA. Este trabalho está dividido da seguinte forma: na Seção 2 é vista a metodologia do trabalho; e por fim, na Seção 3 serão apresentada os resultados esperados.

2. Metodologia

Na literatura, há diversos trabalhos que abordam a Máquina Ética em SMAs e o foco deles está na interação entre os agentes do sistema e sua organização. Atualmente, há duas abor-

dagens: centrada nos agentes e centrada na organização [Lemaître and Excelente 1998]. Na primeira abordagem, o SMA não possui uma representação explícita de sua organização. Espera-se ainda que, por meio da interação entre os agentes e o ambiente, possa emergir comportamentos complexos (auto-organização). Por exemplo, no sistema MANTA é simulada uma colônia de formigas e observou-se que o formigueiro apresentou estratégias de controle populacional e mecanismos de divisão do trabalho mesmo que tais comportamentos não tenham sido intencionalmente programados no código das formigas [Gilbert and Conte 2006]. Nessa abordagem, a estrutura organizacional existe apenas na memória dos agentes e a falta de uma descrição explícita da organização pode dificultar o raciocínio sobre ela.

Na abordagem centrada na organização, é possível obter uma descrição da organização que a sociedade está adotando sem precisar observar seu comportamento. Por exemplo, a descrição da organização de uma universidade existe em manuais, códigos de ética, organogramas, etc. Nessa abordagem, os autores propõem modelos organizacionais para descrever de forma explícita a organização levando em consideração as normas e os princípios que a norteiam. Esses modelos possuem como objetivo restringir o comportamento individual dos agentes buscando produzir um comportamento global direcionado a um objetivo social [Hübner et al. 2002]. Diversos modelos organizacionais têm sido propostos na literatura e cada um oferece uma proposta diferente para representar a organização societal dos agentes em qualquer contexto.

O modelo Moise+ é um modelo organizacional que considera a organização de um SMA segundo três dimensões: a estrutura (papéis e hierarquias), o funcionamento (planos globais) e as normas (obrigações) da organização [Hübner et al. 2002]. O aspecto deôntico liga os aspectos estruturais e o funcionais indicando quais as responsabilidades dos papéis nos planos globais. A estrutura organizacional envolve a definição de papéis, relações entre papéis e os grupos que possibilitam a definição dos níveis individual, social e coletivo. No nível individual, os papéis dos agentes são definidos juntamente com as obrigações das tarefas pertencentes a cada papel (e.g., o professor deve mediar o conhecimento). No nível social, as ligações entre os papéis (e.g., o professor tem autoridade sobre o aluno) e também as relações de compatibilidade entre os papéis (um agente não pode assumir um papel de aluno na turma onde é professor) são estabelecidas. No nível coletivo, um grupo agrega agentes com objetivos comuns e estes grupos podem ser usados na definição de várias estruturas organizacionais.

A área de Máquina Ética para SMA está em evolução e inúmeras discussões estão em aberto, tais como a escolha da melhor abordagem societal para um SMA e a capacidade deste em avaliar sua própria organização para propor mudanças organizacionais. Contudo, a construção de dispositivos físicos usando um SMA embarcado ainda não leva em consideração regras e princípios éticos e tais dispositivos atuam no ambiente sem nenhuma restrição comportamental. Para o desenvolvimento de uma Máquina Ética para SMA embarcado, os passos necessários são propostos nas próximas subseções.

2.1. Escolha de um modelo organizacional que considere construções éticas

Nesta etapa, os modelos organizacionais para SMA, que restringem o comportamento dos agentes segundo as regras e condutas éticas de uma organização, podem ser investigados e testados para um ambiente embarcado. Devido à relevância de trabalhos que empregaram

o modelo Moise+, este pode ser o primeiro modelo organizacional a ser testado para um SMA embarcado. Na investigação dos modelos organizacionais para SMA, pode-se identificar a necessidade de modificá-los para atender aos seguintes pontos:

- **A aderência dos modelos organizacionais aos princípios éticos para uma Máquina Ética.** Segundo [Huang et al. 2022], os princípios para a criação de uma Máquina Ética podem ser classificados em: transparência, equidade e justiça, responsabilidade e prestação de contas, não maleficência, beneficência, solidariedade, sustentabilidade, confiabilidade e dignidade. Os modelos organizacionais existentes podem ser avaliados à luz desses princípios e novas versões podem ser propostas para adequá-los;
- **O modelo organizacional escolhido deve permitir a existência de níveis organizacionais comuns na sociedade em que vivemos.** Por exemplo, um indivíduo deve seguir as regras e condutas éticas da sua profissão (nível micro), da empresa onde trabalha (nível meso) e também seguir as leis federais do governo (nível macro). Dessa forma, o comportamento de uma Máquina Ética deve estar também restrita por essas três perspectivas. Logo, o modelo organizacional para um SMA embarcado precisa garantir que os agentes tenham seus comportamentos restringidos por esses três níveis organizacionais e o caminho para alcançar esse objetivo ainda precisa ser construído;
- **Permitir que atualizações de leis, regras e condutas éticas de uma sociedade sejam refletidas para um SMA em tempo de execução.** Quando ocorre uma mudança nas leis, regras ou condutas, essas alterações devem ser repassadas para os modelos organizacionais em tempo de design. Contudo, quando se trata de SMA embarcado é preciso se pensar em um sistema adaptável ao ambiente em tempo de execução de forma que as mudanças ocorridas em qualquer nível sejam percebidas pelo SMA durante sua execução. Nesse sentido, é necessário pensar em soluções que possibilitem essa atualização em tempo de execução.

2.2. Especificação de uma organização com as regras e condutas éticas em um determinado contexto

Nesta etapa, será necessário o mapeamento de um conjunto de regras e condutas éticas reais de uma organização para o modelo organizacional, ou seja, as regras e condutas éticas de uma organização humana devem ser traduzidas para uma linguagem inteligível por um SMA e esquematizadas no modelo organizacional. Para facilitar esse processo, o pensamento computacional ético pode ser usado previamente pelos projetistas de SMA para a identificação de dilemas éticos em determinado contexto, a avaliação e a escolha da opção mais ética em cenários simulados e a antecipação de impactos negativos e positivos da IA aos indivíduos. Esse pensamento contribui para que os projetistas de SMA compreendam as possíveis consequências éticas das ações e tomadas de decisão que podem ser realizadas pela IA. Consequentemente, esses projetistas podem especificar melhor as restrições comportamentais do SMA no modelo organizacional. Nesse cenário, um ferramental pode ser desenvolvido para apoiar o mapeamento e o pensamento computacional ético dos projetistas de SMA, zelando para que o desenvolvimento e o uso da tecnologia ocorram de maneira ética e responsável.

2.3. Desenvolvimento de um SMA que integre as especificações da organização ao gerenciamento das ações e tomadas de decisão em um dispositivo

Esta etapa aborda o desenvolvimento do SMA pelo projetista considerando as especificações das regras e condutas éticas contidas no modelo organizacional. Em determinado contexto, o SMA é desenvolvido para atuar no ambiente de forma autônoma, no entanto, tendo como limites as restrições comportamentais da organização. Nesse sentido, experimentos podem ser conduzidos para averiguar se o comportamento do SMA no ambiente é adequado às questões éticas. Para isso, os cenários com questões éticas definidos por meio do pensamento computacional ético podem ser simulados e, eventualmente, ajustes na especificação do modelo organizacional podem ser identificados. Nesses experimentos, o atendimento aos princípios de uma Máquina Ética também devem ser avaliados.

2.4. Um ferramental que apoie o desenvolvimento de uma Máquina Ética para SMA, a embarcação da especificação de sua organização e propicie o ensino e aprendizagem do pensamento computacional ético

O ferramental existente para o desenvolvimento de um SMA não apoia o projetista na criação de uma Máquina Ética para SMA embarcado e há várias lacunas que precisam ser preenchidas para garantir avanços nessa área.

- Os modelos organizacionais que inserem preocupações éticas ao SMA ainda não foram testados e adaptados para SMA embarcados.
- Não há práticas de ensino e aprendizagem do pensamento computacional que auxiliem os projetistas de SMA a desenvolverem habilidades críticas para a especificação das regras e condutas éticas nos modelos organizacionais.
- Também não há uma IDE que possibilite o desenvolvimento de um SMA embarcado de forma integrada com um modelo organizacional.

Essas lacunas podem ser resolvidas com o desenvolvimento de um ferramental para apoiar a criação de Máquinas Éticas para SMA embarcado. Há ferramentas que suportam o desenvolvimento de um SMA como a *Cognitive Hardware on Network - Integrated Development Environment* (ChonIDE) e o *Visual Studio Code* (VS Code) [Souza de Jesus et al. 2023]. No entanto, a ChonIDE é uma plataforma *web* e *open source* que apoia especificamente o desenvolvimento de um SMA embarcado considerando as camadas de arquitetura que necessitam de programação. Para facilitar a programação de uma Máquina Ética para SMA embarcado, uma camada adicional pode ser integrada a ChonIDE permitindo que o projetista especifique um modelo organizacional e embarque diferentes níveis de regras e condutas éticas em um dispositivo. Além disso, a IDE pode abarcar as técnicas para o desenvolvimento do ensino e aprendizagem do pensamento computacional. Diante do exposto, propomos os seguintes objetivos para serem atingidos para a criação de Máquinas Éticas Embarcadas:

1. Mapeamento da literatura para investigar os modelos organizacionais e os princípios éticos para a criação de uma Máquina Ética para SMA;
2. Realizar uma análise preliminar da aderência dos modelos organizacionais aos princípios éticos;
3. Escolher um modelo organizacional para a condução de testes com regras e condutas éticas de uma organização real;
4. Realizar testes preliminares para embarcar o SMA integrado com o modelo organizacional.

3. Resultados Esperados

Este estudo pretende levantar e comparar os modelos organizacionais existentes na literatura para a criação de uma Máquina Ética para SMA e também compreender como os níveis organizacionais podem ser especificados nesses modelos. Além disso, este estudo objetiva identificar possíveis lacunas na representação das regras e condutas éticas em uma organização e avaliar a aderência desses modelos aos princípios éticos. Um estudo prático também será conduzido para entender o comportamento de um SMA sob as restrições comportamentais éticas de uma organização e realizar testes preliminares para embarcar o SMA integrado com o modelo organizacional.

Referências

Albino, V., Berardi, U., and Dangelico, R. M. (2015). Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of urban technology*, 22(1):3–21.

Allen, C., Wallach, W., and Smit, I. (2006). Why machine ethics? *IEEE Intelligent Systems*, 21(4):12–17.

Boden, M. A. (1996). *Artificial intelligence*. Elsevier.

Briot, J.-P. and Demazeau, Y. (2001). *Principes et architecture des systèmes multi-agents*. Hermès-Lavoisier.

Gilbert, N. and Conte, R. (2006). Manta: New experimental results on the emergence of (artificial) ant societies. In *Artificial societies*, pages 172–191. Routledge.

Huang, C., Zhang, Z., Mao, B., and Yao, X. (2022). An overview of artificial intelligence ethics. *IEEE Transactions on Artificial Intelligence*, 4(4):799–819.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2002). Moise+: towards a structural, functional, and deontic model for mas organization. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 501–502.

Lemaître, C. and Excelente, C. B. (1998). Multi-agent organization approach. In *Proceedings of II Iberoamerican Workshop on DAI and MAS*, pages 7–16. Toledo.

Management Association, I. (2017). *The Internet of Things: Breakthroughs in Research and Practice: Breakthroughs in Research and Practice*. Critical explorations. IGI.

Metz, C. (2019). Is ethical ai even possible. *The New York Times*, 1(3).

Nalini, B. (2019). The hitchhiker's guide to ai ethics medium.

Paulo, N. (2023). The trolley problem in the ethics of autonomous vehicles. *The Philosophical Quarterly*, 73(4):1046–1066.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Pearson.

Souza de Jesus, V., Mori Lazarin, N., Pantoja, C. E., Vaz Alves, G., Ramos Alves de Lima, G., and Viterbo, J. (2023). An IDE to Support the Development of Embedded Multi-Agent Systems. In Mathieu, P., Dignum, F., Novais, P., and De la Prieta, F., editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*, pages 346–358, Cham. Springer Nature Switzerland. DOI: https://doi.org./10.1007/978-3-031-37616-0_29.

Weiss, G. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press.

Uma aplicação embarcada de sistemas multiagentes normativos para operação e regulação de processo de manufatura*

Bruno A. Stefano[1], **Jaime S. Sichman**[1]

[1]Laboratório de Sistemas Inteligentes (LTI)
Escola Politécnica (EP)
Universidade de São Paulo (USP)
São Paulo, SP – Brasil.

{bruno.stefano,jaime.sichman}@usp.br

Abstract. *This paper presents the implementation of an embedded normative multi-agent system for the operation and regulation of a manufacturing process. A simulated scenario, based on a real scenario, is used for the implementation. Autonomous agents and norms were developed exemplifying the possibility of use of this technology in an industrial environment. This application enables the automation of more complex industrial processes, collaboration between different stages of the manufacturing system and automated production line regulation.*

Resumo. *Este artigo apresenta a implementação de um sistema multiagentes normativo embarcado na operação e regulação de um processo de manufatura. Uma simulação baseada em um cenário real é utilizada para a implementação. Foram desenvolvidos agentes autônomos e normas que exemplificam a possibilidade de uso dessa tecnologia em um ambiente industrial. Essa aplicação possibilita a automação de processos industriais mais complexos, a colaboração entre diferentes etapas do sistema de manufatura e regulação automatizada da produção.*

1. Introdução

As técnicas mais recentes de manufatura, como a manufatura aditiva e a customização em massa exigem a inserção de novas tecnologias nos chãos de fábrica. A importância de aspectos como sustentabilidade e governança se mostra cada vez mais presente em cada etapa da cadeia produtiva. Processos industriais mais complexos possibilitam a utilização de novas formas de automação e controle. Agentes autônomos aparecem como uma potencial opção para esse tipo de desafio tecnológico. Sua heterogeneidade e capacidade de raciocínio ajuda a adaptar a automação a uma indústria que muda cada vez mais rápido. A utilização de normas que regulam o comportamento dos agentes pode trazer para dentro dos sistemas de manufatura as regulações da indústria. Ainda assim, esse tipo de tecnologia é pouco explorado para aplicações industriais e de manufatura.

Este trabalho apresenta a aplicação de um sistema multiagentes normativo na operação de um processo de manufatura. Uma simulação baseada em uma situação real é

proposta para a implementação da aplicação. Os agentes são embarcados e se comunicam diretamente com os dispositivos da linha de produção. Normas são utilizadas para regular a operação dos agentes e o processo de manufatura como um todo. O objetivo dessa demonstração é, então, mostrar a possibilidade da utilização de um sistema multiagentes normativo embarcado na operação e regulação de um processo de manufatura.

A estrutura deste trabalho segue com a apresentação dos trabalhos relacionados na seção 2. O cenário motivador é exposto na seção 3. A seguir, na seção 4, é descrita a demonstração realizada. A seção 5 conclui o trabalho e apresenta seus possíveis próximos passos.

2. Trabalhos Relacionados

Existem, na literatura, duas principais opções para o desenvolvimento de sistemas multiagentes embarcados. Uma delas é o *Jason Embedded* [Pantoja et al. 2023], uma versão spin-off do framework *Jason* [Bordini et al. 2007]. Essa versão traz ao *Jason* a capacidade de controlar e ler sensores e atuadores, além de um filtro das percepções provenientes dos sensores, proposta inicialmente em [Stabile and Sichman 2015]. Esta funcionalidade é proporcionada pela arquitetura *Argo* [Pantoja et al. 2016], que utiliza um middleware executar a linguagem Java em processadores Arduino. Uma outra capacidade dessa versão é a de se comunicar com outros agentes e outros sistemas multiagentes, com a possibilidade de realizar migração de agentes, o que abre caminho para o desenvolvimento de sistemas multiagentes abertos, ou Open MAS.

A outra principal opção encontrada para o desenvolvimento de sistemas multiagentes embarcados é o framework *Embedded-Mas* [Dias and Brito 2023], cuja versão mais recente foi utilizada em conjunto com o framework de organizacional *Moise*. O *Embedded-Mas* possibilita a integração do *Jason* com o sistema operacional ROS. Dessa forma, os agentes *Jason* são munidos com capacidade de leitura nos tópicos do ROS para percepção e escrita nos tópicos e chamadas de serviços ROS para atuação. O framework *Moise* traz ao sistema multiagentes a possibilidade de programar uma camada organizacional: torna-se possível definir papéis e grupos para os agentes, criar objetivos e missões para os papéis e grupos, além de atribuir obrigações aos papéis e grupos em relação aos objetivos.

O *Jason Embedded* originalmente não traz nenhum aspecto normativo em sua concepção, ao menos até a escrita deste trabalho. O *Embedded-Mas* tampouco aborda questões normativas. Quando integrado ao Moise, entretanto, permite a definição de obrigações através dos aspectos organizacionais de papéis e missões. Entretanto, em ambas as abordagens o aspecto normativo é incompleto ou inexistente até o momento, pois a definição de normas é utilizada diretamente relacionada ao aspecto organizacional. Por outro lado, faltam mecanismos de regulação e inexiste a possibilidade de expressar a regulação nas definições do sistema.

3. Cenário Motivador

O cenário proposto para demonstração é baseado na Fábrica do Futuro (USP)[1], um ambiente de manufatura próximo do real. É um laboratório de pesquisa, ensino e demonstração

[1]https://sites.usp.br/fabricadofuturo/.

de conceitos e tecnologias da Indústria 4.0. O laboratório utiliza uma montagem de skate para ilustrar os processos industriais.

Este estudo foca na etapa de controle de qualidade por visão computacional do processo de montagem do skate. Para realizar customização em massa, cada skate produzido tem alguns pontos de customização. Um cliente pode, por exemplo, escolher a cor de cada roda da prancha. Após a colocação das rodas do skate, um sistema de visão computacional faz a identificação de suas cores e valida a customização [Zancul et al. 2020].

Na demonstração a ser apresentada, o processo de validação da customização será coordenado por agentes autônomos. O sistema de visão computacional será simulado por um software que gera os dados das cores das rodas do skate. A demonstração, portanto, não será executada em um ambiente de manufatura real. Com os dados de visão computacional gerados, um sistema multiagentes normativo então é responsável por analisar o pedido do cliente e o que de fato foi executado na fábrica e concluir se a montagem das rodas foi feita corretamente. Além do raciocínio dos agentes, normas serão utilizadas para ilustrar a possibilidade de regulação desse processo [2].

4. Demonstração

O sistema de visão computacional, baseado no da Fábrica do Futuro (USP), é aqui simulado por um microprocessador Arduino Mega. Um código específico para esse propósito foi desenvolvido. Tal código gera pseudo aleatoriamente uma dupla de dados (cor predominante, percentual da área da imagem ocupada por essa cor) para cada roda do skate, identificadas como frente, trás, esquerda e direita. As cores são escolhidas pseudo-aleatoriamente em uma lista com as opções vermelho, verde, azul, amarelo, preto e branco. A porcentagem da área ocupada pela cor também é gerada pseudo-aleatoriamente, porém sempre maior que 80%. Através da biblioteca Javino [Pantoja and Lazarin 2015], o Arduino consegue se comunicar com os agentes do sistema multiagentes. O Arduino tem então o papel de enviar as leituras do sistema de visão computacional geradas para serem interpretadas como percepções pelos agentes.

O sistema multiagentes normativo, responsável pela validação da customização, é formado por agentes de arquitetura BDI implementados na plataforma JaCaMo [Boissier et al. 2020]. Esse sistema é executado em um processador Raspberry Pi modelo 3B, que possui conexão serial com o Arduino. Especificamente, dois agentes são utilizados para essa tarefa. Um agente *validador* é responsável por se comunicar com o sistema de visão computacional simulado, raciocinar a partir dos dados obtidos e informar suas conclusões. O outro agente de *controle de qualidade* informa as especificações dos pedidos dos clientes e aguarda a conclusão da validação realizada. Uma norma que regula a operação do validador é implementada para incentivar que os pedidos sejam validados dentro de um limite de tempo. A Figura 1 apresenta uma visão geral da demonstração.

O agente validador tem o importante papel de se comunicar com o sistema de visão computacional simulado pelo Arduino. Os dados gerados pela visão computacional são integrados como crenças do agente através da arquitetura de agentes Argo [Pantoja et al. 2016]. A partir das preferências de cor recebidas do controle de qualidade

[2]Disponível em: https://github.com/brunoastefano/demonstrations/tree/master/smart-factory-simulation.

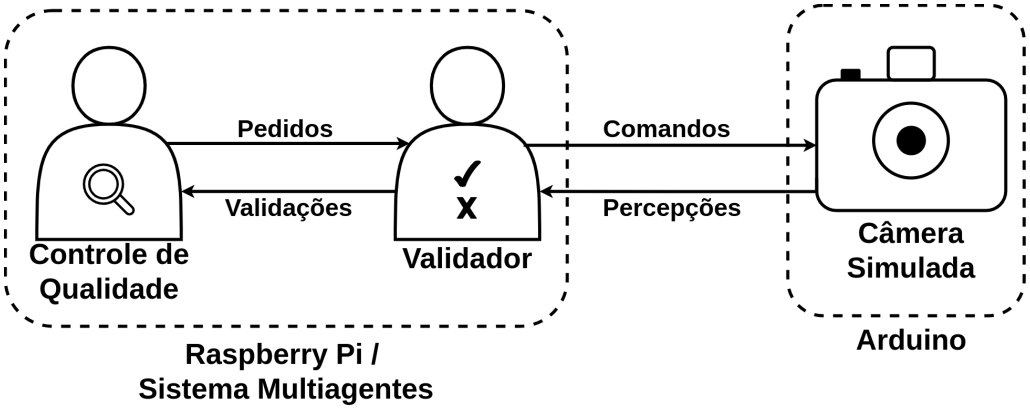


Figura 1. Visão geral da demonstração.

e de um nível de confiança mínimo previamente estipulado, que indica o mínimo percentual de área coberto pela cor para que ela seja aceita, o agente validador determina a conformidade de cada roda do skate. Com as atualizações das crenças sobre a conformidade de cada roda do skate, o agente validador consegue raciocinar sobre a conformidade geral do skate. Com a conclusão sobre a conformidade do skate, o agente validador pode então informar ao agente de controle de qualidade sobre a conformidade do skate com as preferências do cliente. A Figura 2 traz um trecho do código do raciocínio do agente sobre a conformidade do skate.

```
1  +frontLeftMeasure(ColorMeasure,ConfidenceMeasure):
2  frontLeftSettings(ColorSetting) & minimumConfidence(MinCon) &
3  ColorMeasure == ColorSetting & ConfidenceMeasure > MinCon <-
4    +compliant(frontLeft);
5    !updateCompliance;
6    .
7
8  +!updateCompliance: compliant(frontLeft) & compliant(frontRight) &
9  compliant(backLeft) & compliant(backRight) <-
10   +compliant(skateboard);
11   !communicateCompliance;
12   .
```

Figura 2. Fragmento do código do raciocínio do agente sobre a conformidade do skate.

O agente de controle de qualidade gera as preferências do cliente de modo pseudo-aleatório. Esses dados são crenças do agente que serão comunicadas ao agente validador à medida que forem sendo atualizados. Como se trata de um ambiente simulado, essa lógica representa os pedidos dos clientes que vão chegando à linha de produção. Junto das especificações do pedido, o agente de controle qualidade também informa que um pedido está ativo e incrementa um contador de pedidos. O controle de qualidade deve então ser informado sobre a conclusão da validação, através do resultado. Assim, o agebte de controle de qualidade incrementa um contador de resultados e informa ao agente validador que não há mais um pedido ativo.

A regulação do sistema é feita através de uma norma implementada utilizando a linguagem de programação normativa NPL(s) [Yan et al. 2024], ilustrada na Figura 3. O

agente de controle de qualidade utiliza uma abordagem agente-cêntrica para o processamento de normas através de uma nova arquitetura de agentes capaz de processar a NPL(s) [Yan et al. 2024]. A cada novo pedido ativo, a norma representada em NPL(s) exibida na Figura 3 entra em vigor e ela exige que o controle de qualidade receba um novo resultado de validação em até 10 segundos. Caso o resultado não chegue, uma sanção é aplicada e o agente de validação perde 5% de reputação. Essa perda de reputação pode ser usada para desqualificar este agente, ou até para agilizar seu processo de validação.

```
1 norm n1: orders(N) & N > 0
2   -> obligation(quality_agent, n1, results(N), `10 seconds`)
3     if unfulfilled: sr1(camera_agent,5).
4 sanction-rule sr1(Ag,Value) -> sanction(Ag,loseReputation(Value)).
```

Figura 3. Norma escrita em NPL(s) para regulação da operação dos agentes.

5. Conclusão

A aplicação implementada, uma simulação baseada em caso real, demonstra o uso de sistemas multiagentes normativos embarcados em um cenário de manufatura. A utilização dessa tecnologia em cenários industriais abre possibilidades para automação de processos de raciocínio e tomada de decisão. A implantação de normas permite a regulação dos processos automatizados, facilitando a integração dos sistemas computacionais com diretrizes da indústria, legislação e outros tipos de regulação. A comunicação entre os agentes simplifica a conexão entre diferentes etapas do processo de produção, que normalmente são feitas por interfaces entre diferentes sistemas ou até por intervenção humana.

Como essa primeira demonstração foi realizada em um ambiente simulado, trabalhos futuros devem explorar a aplicação dessas tecnologias em um cenário real. Assim é possível extrair seus reais benefícios para os processos de manufatura. Para enriquecer a implementação, é viável a maior exploração do impacto das sanções no comportamento dos agentes. Isso pode ser feito com o uso de sanções positivas (recompensas) e implementação das reações dos agentes às sanções. Além disso, o desenvolvimento de mais agentes e mais normas pode deixar mais completos a representação e o controle do sistema de manufatura.

Agradecimentos

O presente trabalho foi realizado no âmbito do projeto FAPESP/ANR NAIMAN, financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Processo Número 2022/03454-1 e também contou com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Codigo de Financiamento 001.

Referências

Boissier, O., Bordini, R. H., Hübner, J. F., and Ricci, A. (2020). *Multi-Agent Oriented Programming*. MIT Press.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*, volume 15. John Wiley & Sons.

Dias, P. and Brito, M. d. (2023). Coordenação de robôs ROS com Agentes BDI e MOISE. In *Anais do XVII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações - WESAAC 2023*, pages 67–78, Pelotas, RS.

Pantoja, C. and Lazarin, N. (2015). A Robotic-agent Platform for Embedding Software Agents Using Raspberry Pi and Arduino Boards. In *Proceedings do 9o. Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações (WESAAC 2015)*, pages 13–20.

Pantoja, C. E., de Jesus, V. S., Lazarin, N. M., and Viterbo, J. (2023). A spin-off version of Jason for IoT and embedded multi-agent systems. In Naldi, M. C. and Bianchi, R. A. C., editors, *Intelligent Systems - 12th Brazilian Conference, BRACIS 2023, Belo Horizonte, Brazil, September 25-29, 2023, Proceedings, Part I*, volume 14195 of *Lecture Notes in Computer Science*, pages 382–396. Springer.

Pantoja, C. E., Jr., M. F. S., Lazarin, N. M., and Sichman, J. S. (2016). ARGO: an extended Jason architecture that facilitates embedded robotic agents programming. In Baldoni, M., Müller, J. P., Nunes, I., and Zalila-Wenkstern, R., editors, *Engineering Multi-Agent Systems - 4th International Workshop, EMAS 2016, Singapore, Singapore, May 9-10, 2016, Revised, Selected, and Invited Papers*, volume 10093 of *Lecture Notes in Computer Science*, pages 136–155. Springer.

Stabile, M. F. and Sichman, J. S. (2015). Evaluating perception filters in BDI Jason agents. In *2015 Brazilian Conference on Intelligent Systems, BRACIS 2015, Natal, Brazil, November 4-7, 2015*, pages 116–121. IEEE Computer Society.

Yan, E., Nardin, L. G., Hübner, J. F., and Boissier, O. (2024). An Agent-Centric Perspective on Norm Enforcement and Sanctions. In *COINE - International Workshop on Coordination, Organizations, Institutions, Norms and Ethics for Governance of Multi-Agent Systems*, Auckland, Nova Zelândia.

Zancul, E., Martins, H. O., Lopes, F. P., and Da Silva Neto, F. A. (2020). Machine Vision applications in a Learning Factory. *Procedia Manufacturing*, 45:516–521.

Detecção de Incêndio apoiada por Agentes Cognitivos

Guilherme A. Leite, Charles J. Heringer, Diogo M. Fernandes, Nilson M. Lazarin

Bacharelado em Engenharia Elétrica – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ) – Nova Friburgo, RJ – Brasil

{guilherme.leite,charles.heringer,diogo.fernandes}@aluno.cefet-rj.br

Abstract. *Given the history and current statistics of fires in Brazil, this work analyzes the possibility of using artificial intelligence of things in fire detection, supported by embedded cognitive agents and communication via the IoT network. It differs from previous approaches by allowing dynamic adjustments to detection parameters without firmware reprogramming. Furthermore, a case study aims to show the positive impact of Multi-agent Systems for social good.*

Resumo. *Dado o histórico e as estatísticas atuais de incêndios no Brasil, este trabalho analisa a viabilidade do uso de inteligência artificial na detecção de incêndios, utilizando agentes cognitivos embarcados e comunicação via rede IoT. Diferencia-se de abordagens anteriores ao permitir ajustes dinâmicos nos parâmetros de detecção, sem reprogramação de firmware. Além disso, é apresentado um estudo de caso que visa demonstrar o impacto positivo dos Sistemas Multi-agente na segurança pública e prevenção de incêndios.*

1. Introdução

A segurança contra incêndios em ambientes residenciais, comerciais e industriais é um tema relevante para pesquisas em diversas áreas, dado que a história nacional é composta por vários infelizes incidentes, tais como o incêndio da boate Kiss, o incêndio no Gran Circus Norte-Americano, o incêndio do Edifício Joelma ou ainda o ocorrido no alojamento do Flamengo [Fernandes 2020]. Além disso, só no ano de 2020 foram registrados mais de 700 incêndios por dia no Brasil, segundo a Secretaria Nacional de Segurança Pública (SENASP) [Folha Metropolitana 2023]. Sendo que, mais de 50% dos incêndios domésticos daquele ano foram causados por sobrecarga elétrica, segundo a Associação Brasileira de Conscientização dos Perigos da Eletricidade (Abracopel) [Gandra 2021].

O risco de incêndio faz parte do nosso cotidiano e tradicionalmente os sistemas de detecção de incêndio já atuam na mitigação desses riscos. A motivação deste trabalho é criar um sistema de alerta para combate ao incêndio, utilizando tecnologias avançadas para aumentar a segurança e minimizar danos. No entanto, os avanços na tecnologia permitem a evolução desses sistemas, e um dos desenvolvimentos mais notáveis é a possibilidade da integração da inteligência artificial (IA) na detecção de incêndios. De fato, os sensores de incêndio, impulsionados por IA, podem contribuir para o aprimoramento da precisão e eficiência desses sistemas. Por exemplo, em [Perilla et al. 2018] é apresentado um framework para integração de Internet das Coisas (IoT) a sistemas de segurança contra incêndio, uma revisão das possíveis tecnologias IoT aplicáveis para o cenário, além de

uma análise qualitativa (discussões em grupos específicos e entrevistas), para determinar os recursos IoT desejados para segurança contra incêndio. Já em [Mahzan et al. 2018] é apresentado um protótipo de alarme residencial de incêndio composto pela plataforma Arduino (ATmega328), sensor de temperatura LM35 e módulo GSM (*Global System for Mobile Communications*) capaz de enviar mensagem de alerta aos proprietários, em caso de detecção de calor acima de 40°C na residência. Por outro lado, em [Yadav and Rani 2020] é apresentado um sistema de alarme de incêndio baseado em Arduino capaz ativar um pulverizados de água para combater o fogo, quando os sensores detectam fumaça ou que a temperatura está acima de 104°F, além de notificar o proprietário via GSM.

Neste trabalho exploramos a convergência entre a tecnologia de detecção de incêndio e o paradigma da Inteligência Artificial das Coisas (*AI of things* - AIoT) [Zhang and Tao 2021], através de agentes cognitivos embarcados [Pantoja et al. 2016] para identificar ameaças de incêndio e otimizar o tempo de socorro. Diferente de [Mahzan et al. 2018] e [Yadav and Rani 2020], apresentamos uma comunicação via uma rede IoT, que permite a integração com outros sistemas cognitivos em execução em órgãos governamentais, possibilitando que os parâmetros de detecção de incêndio possam ser dinamicamente definidos por uma autoridade, sem a necessidade de reprogramação de firmware. Diferente de [Perilla et al. 2018], analisamos as tecnologias que permitem a construção de sistemas ciber-físicos usando o paradigma AIoT, gerenciados por agentes cognitivos, capazes de perceber e atuar no mundo físico. Por fim, visamos demonstrar o potencial impacto positivo do uso de Sistemas Multi-agente (SMA) na prevenção de incêndios.

2. Fundamentação Teórica

Um Sistema Multi-agente (SMA) é um grupo de entidades de software que apresentam um comportamento autônomo em um ambiente físico ou digital, baseado em suas percepções e em suas interações. Um *agente reativo* é aquele que apresenta um comportamento tipo estimulo-resposta, sem memória sobre interações ou percepções passadas, nem previsão sobre futuras. Por outro lado, um *agente cognitivo* é aquele que apresenta um comportamento baseado em organizações sociais (grupos, hierarquias, etc.), capazes de cooperar e competir com outros agentes do sistema [Sichman et al. 1992].

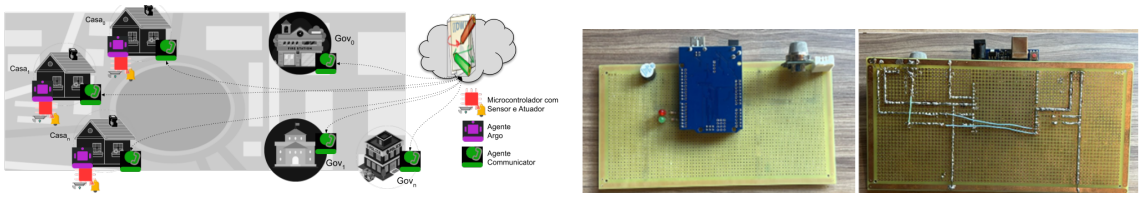
Um SMA Embarcado, por sua vez, é um sistema hospedado em um dispositivo IoT tal como um computador de placa única (Raspberry Pi, por exemplo) conectado diretamente a sensores e atuadores, permitindo que os agentes possam sentir e agir no mundo físico. Esses sistemas podem ser construídos seguindo uma arquitetura de quatro camadas, são elas: *hardware*, camada dos sensores e atuadores que efetivamente manifestam as ações SMA Embarcado no mundo físico; *firmware*, camada do microcontrolador que converte os dados brutos dos sensores, em crenças para os agentes; *interfaceamento*, camada que interliga o microcontrolador com o SMA hospedado no dispositivo IoT; e *raciocínio*, o habitat dos agentes cognitivos [Pantoja et al. 2016].

Um modelo útil na modelagem de agentes cognitivos é o *Belief-Desire-Intention* (BDI) [Bratman et al. 1988] que permite a utilização de *crença*, uma representação do conhecimento dos agentes sobre o ambiente; *desejo*, uma representação do estado do mundo que o agente quer atingir; e *intenção*, uma representação da sequência de ações que o agente se compromete a executar para atingir seus objeti-

vos [Hübner et al. 2004]. Para construir agentes cognitivos embarcados, baseados no modelo BDI, uma opção é utilizar *Jason*, uma linguagem de programação orientada a agentes [Bordini et al. 2005], adotada pelo *JasonEmbedded*, um framework específico para IoT e sistemas ciber-físicos [Pantoja et al. 2023], através do ambiente de desenvolvimento *chonIDE* [Souza de Jesus et al. 2023]. No desenvolvimento de AIoT com o *JasonEmbedded*, pode-se empregar agentes cognitivos com diferentes arquiteturas. Neste trabalho são utilizados os agentes BDI *Communicator* [de Jesus et al. 2019] e *Argo* [Pantoja et al. 2016]. O primeiro possuí a capacidade de comunicação com agentes externos ao SMA em que se encontra, por meio de uma rede IoT. O segundo, por sua vez, consegue manipular gerenciar microcontroladores ATMEGA.

3. Proposta

Propomos um sistema de monitoramento de incêndio baseado em agentes cognitivos que integram residências e órgãos governamentais, por meio de uma rede ContextNet [Endler et al. 2011]. Em cada residência um SMA Embarcado com: i) agente *Argo* capaz de perceber com sensores de temperatura e fumaça, e atuar com alarme sonoro; ii) agente *Communicator* capaz de emitir alerta para o SMA do corpo de bombeiros. Em cada órgão governamental um SMA com um agente *Communicator*, capaz de alertar residências vizinhas ou solicitar apoio de outros órgãos, auxiliando na rápida resposta em caso de incidentes. Uma visão geral da integração dos SMA's é apresentada na Figura 1a.



(a) Visão geral da integração. **(b) Vista Frontal e Traseira do módulo.**

Figura 1. Detecção de Incêndio apoiada por agentes BDI embarcados.

Buscando viabilizar a proposta, foi construído um SMA Embarcado para ser utilizado nas residências. Na camada de *hardware*, conforme a Figura 1b, foi utilizado: 01 sensor de gás inflamável e fumaça (*MQ-02*), 01 sensor de temperatura (*RHT-03*), 01 buzzer (*5V ativo*) e 02 diodos emissores de luz (*LED*), um verde e um vermelho. Na camada de *firmware*, por sua vez, foi utilizado um microcontrolador *Arduino Uno R3*. Por fim, na camada de *interfaceamento* foi utilizada a biblioteca Javino [Lazarin and Pantoja 2015].

```
#include <Javino.h>    /* https://github.com/chon-group/javino2arduino/releases/latest */
#include <DHT12.h>     /* https://www.arduino.cc/reference/en/libraries/dht12-sensor-library/ */
const int temp=4; const int fumaca=2; const int vermelho=5; const int verde = 3; const int buzzer=11; /* GPIO pin */
Javino javino; DHT12 dht12(temp, true);
void serialEvent(){ javino.readSerial(); }
void setup() {
javino.act["aviso"]  = funcaoAviso;
javino.act["standby"] = funcaoStandby;
javino.perceive(getExogenousPerceptions);
javino.start(9600);
pinMode(temp,INPUT);pinMode(fumaca,INPUT); pinMode(buzzer,OUTPUT);pinMode(vermelho,OUTPUT);pinMode(verde,OUTPUT); }
void loop() {  javino.run(); }
void getExogenousPerceptions(){
    if(digitalRead(fumaca) == 0){ javino.addPercept("fumaca(false)");} else{ javino.addPercept("fumaca(true)"); }
    float temp = dht12.readTemperature();  javino.addPercept("temperatura("+String(temp)+")");}
void funcaoAviso(){ digitalWrite(vermelho,HIGH); digitalWrite(verde,LOW);  tone(buzzer, 1000); }
void funcaoStandby(){ digitalWrite(vermelho,LOW); digitalWrite(verde,HIGH);  noTone(buzzer); }
```

Código-fonte 1. Módulo Sensor de Incêndio para SMA Embarcado.

Ao ligar o protótipo, o *LED* verde acenderá, indicando que os sensores estão em funcionamento. Simultaneamente, o microcontrolador começará a receber os dados do sensor de gás e fumaça e do sensor de temperatura. Se os valores recebidos ultrapassarem os limites estabelecidos no Código Fonte 1, o microcontrolador acionará o buzzer e o *LED* vermelho, sinalizando a ocorrência de um possível incêndio. O módulo construído informa duas crenças para o agente BDI: i) *fumaca(**b**)*, onde ***b*** é um termo $(true|false)$; ii) *temperatura(**N**)*, onde ***N*** é um decimal de dois dígitos. Além disso, o módulo aceita dois comandos de atuação no ambiente: i) *aviso*, neste caso um alerta luminoso e sonoro é ativado; ii) *standby*, neste caso o led verde é ativado e o alerta desativado.

4. Estudo de Caso

Diferente de [Mahzan et al. 2018] e [Yadav and Rani 2020], nesta abordagem o firmware do módulo sensor não possuí informações estáticas sobre os valores que definem quando há ou não um princípio de incêndio. Esses parâmetros são dinamicamente definidos pela autoridade responsável. Assim, o sistema é adaptável para qualquer região e ainda pode considerar mudanças de temperatura, nas diferentes estações do ano. Pois, ao iniciar a execução do SMA Embarcado, o agente comunicador busca os parâmetros atualizados.

Buscando avaliar a aplicação da proposta, consideramos um cenário de integração de uma casa inteligente e o corpo de bombeiros. A casa possuí um SMA Embarcado equipado com um módulo de monitoramento de incêndio e dois agentes cognitivos, um para perceber a atuar no ambiente e outro para comunicar-se com as autoridades. O corpo de bombeiros, por sua vez, possuí um SMA com um agente responsável pela orientação. Para a execução do cenário, utilizamos uma abordagem de simulação do ambiente exógeno do SMA Embarcado, conforme proposto por [Freitas et al. 2023]. Foram utilizas duas máquinas virtuais, uma para executar o SMA da casa e outra o dos bombeiros.

A camada de *raciocínio* do SMA Embarcado é apresentada, através dos Códigos 2 e 3. No primeiro, um agente *Argo* possuí apenas o objetivo inicial de executar o plano *perceber* que busca conectar-se ao microcontrolador e perceber o ambiente (*.percepts(open)*). Este plano é acionado depois de algum tempo, caso a porta serial não esteja disponível ou ele ainda não tenha recebido as configurações do corpo de bombeiros. No segundo, um agente *Communicator*, possuí um objetivo inicial de executar o plano *buscarConfiguracoes* que envia mensagens ao SMA dos bombeiros, solicitando os planos para o sistema. Este agente possuí ainda dois planos *novoPlano* e *novaCrenca* para retransmitir internamente as orientações dos bombeiros.

```
1  /* Crenca Inicial - use ttyACM0 ou ttyUSB0 para conectar a um Arduino real */
   serialPort(ttyEmulatedPort0).
3  /* Objetivo Inicial */
   !perceber.
5  /* Planos */
   +!perceber: serialPort(P) <- .abolish(port(P,_)); .port(P); .limit(750); .percepts(open).
7  +port(Port,Status): Status = off | Status = timeout <- .percepts(close); .wait(10000); !perceber.
   +port(Port,Status): Status = on & not configuracao(ok)<- .percepts(close); .wait(30000); !perceber.
```

Código-fonte 2. Raciocínio do agente *Argo* no SMA da casa inteligente.

```
   bombeirosUUID("bcb38a40-8605-4f07-a94a-d96678498165").  /* Crenca Inicial */
2  !buscarConfiguracoes.                                  /* Objetivo Inicial */
   /* Planos */
4  +!buscarConfiguracoes: bombeirosUUID(B)<- .connectCN("skynet.chon.group",5500,"efa435a4-42f5-4927-8d4b-2c77ce9e8eed");
       .sendOut(B,achieve,getConf(self,communicator));  .sendOut(B,achieve,getConf(agenteArgo,argo)).
6  +!novoPlano(Agente,Plano)[source(Externo)] <- .send(Agente,tellHow,Plano).
   +!novaCrenca(Agente,Crenca)[source(Externo)] <- .send(Agente,tell,Crenca).
```

Código-fonte 3. Raciocínio do agente *Communicator* no SMA da casa inteligente.

Na segunda máquina virtual executa o SMA do corpo de bombeiros. Neste, um agente *Communicator* possuí um objetivo inicial de executar um plano de conexão à rede IoT (*connect*). Além desse, ele possui um plano *getConf* com dois contextos distintos, para enviar ao solicitante os planos e crenças necessários, conforme o *Tipo* de arquitetura.

```
crencasPublicas([principioIncendio(49),standbyMSG("Online!"),incendioMSG("Alerta!"),bombeirosMSG("A caminho!")]).
!connect.                    /* Objetivo Inicial */
/* Planos */
+!connect <- .connectCN("skynet.chon.group",5500,"bcb38a40-8605-4f07-a94a-d96678498165").
+!getConf(Agt,Tipo)[source(S)]: Tipo=argo & crencasPublicas(Info)<-
  !sendBelief(S,Agt,Info);
  !sendPlan(S,Agt,"+!incidente <- .act(aviso); .broadcast(tell,incendio).");
  !sendPlan(S,Agt,"+!standby <- .act(standby); .broadcast(untell,incendio).");
  !sendPlan(S,Agt,"+temperatura(T): principioIncendio(N) & T > N <- !incidente.");
  !sendPlan(S,Agt,"+temperatura(T): principioIncendio(N) & T<N & not fumaca(true) <- .broadcast(untell,incendio).");
  !sendPlan(S,Agt,"+fumaca(false) <- .act(standby); .broadcast(untell,incendio).");
  !sendPlan(S,Agt,"+fumaca(true) <- !incidente.");
  !sendPlan(S,Agt,"+port(Port,Status): Status = on & configuracao(ok)<- !standby.");
  !sendBelief(S,Agt,configuracao(ok)).
+!getConf(Agt,Tipo)[source(S)]: Tipo=communicator & crencasPublicas(Info)<-
  !sendBelief(S,Agt,Info);
  !sendPlan(S,Agt,"+!acionarBombeiros: bombeirosUUID(B) <- .sendOut(B,achieve,incendio).");
  !sendPlan(S,Agt,"+incendio: incendioMSG(M) <- .print(M); !acionarBombeiros.").
+!sendBelief(Casa,Agente,Crenca)<- .sendOut(Casa,achieve,novaCrenca(Agente,Crenca)).
+!sendPlan(Casa,Agente,Plano)<- .sendOut(Casa,achieve,novoPlano(Agente,Plano))
+!incendio[source(Casa)] <- .sendOut(Casa,tell,bombeirosAcaminho).
```

Código-fonte 4. Raciocínio do agente *Communicator* no SMA dos bombeiros.

A execução do cenário ocorreu de forma adequada demonstrando a possibilidade da aplicação de SMA Embarcados no contexto de combate e prevenção de incêndios. Buscando garantir a reprodutibilidade, uma página[1] está disponível com os fontes das diferentes camadas e um vídeo demonstrando o estudo de caso.

Os agentes cognitivos melhoram a detecção de incêndios ao usar sensores avançados e algoritmos de inteligência artificial para monitorar e analisar dados em tempo real, tomar decisões autônomas e estabelecer comunicação com outros sistemas. Capazes de reduzir falsos positivos e coordenar ações de resposta rápidas, como alertar ocupantes e acionar sistemas de supressão de incêndio, assim, eles contribuem para aumentar a precisão e eficiência na resposta às emergências, garantindo uma abordagem integrada.

5. Conclusão

Este trabalho apresentou uma análise do uso das tecnologias de construção de agentes cognitivos embarcados no apoio a sistemas de detecção e prevenção de incêndios. Para tal, foi construído um módulo sensor de fumaça e temperatura, compatível com agentes BDI que utilizam a arquitetura *Argo*. Foi possível ainda, por meio de um estudo de caso, apresentar o potencial do uso de Sistemas Multi-agente no contexto de inteligência artificial para o bem-estar social. Trabalhos futuros podem explorar outros contextos, demonstrando a viabilidade da aplicação do modelo Belief-Desire-Intention, através do framework *JasonEmbedded* e do ambiente de desenvolvimento *ChonIDE*, no paradigma de Inteligência Artificial das Coisas, através da construção de dispositivos cognitivos.

Referências

Bordini, R. H., Hübner, J. F., and Vieira, R. (2005). *Jason and the Golden Fleece of Agent-Oriented Programming*, pages 3–37. Springer US, Boston, MA.

Bratman, M. E., Israel, D. J., and Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355.

[1] https://papers.chon.group/WESAAC/2024/deteccaoIncendio/

de Jesus, V. S., Manoel, F. C. P., and Pantoja, C. E. (2019). Protocolo de interação entre SMA embarcados bio-inspirado na relação de predatismo. In *Anais do WESAAC 2019*, pages 95–106, Florianópolis. UFSC.

Endler, M., Baptista, G., Silva, L. D., Vasconcelos, R., Malcher, M., Pantoja, V., Pinheiro, V., and Viterbo, J. (2011). ContextNet: context reasoning and sharing middleware for large-scale pervasive collaboration and social networking. In *Proceedings of Middleware '11*, Lisbon. ACM.

Fernandes, V. (2020). Os 15 maiores incêndios do Brasil | Tragédias que marcaram.

Folha Metropolitana (2023). Brasil tem mais de nove incêndios por hora.

Freitas, B. P. T., Lazarin, N. M., and Pantoja, C. E. (2023). Uma Proposta de Emulador de Portas Seriais para Sistemas Multiagentes Embarcados. In *Anais do WESAAC 2023*, pages 55–66, Pelotas. UFPel.

Gandra, A. (2021). Sobrecarga elétrica causa mais de 50% dos incêndios domésticos.

Hübner, J. F., Bordini, R. H., and Vieira, R. (2004). Introdução ao desenvolvimento de sistemas multiagentes com Jason. In *Anais da XII Escola de Informática da SBC*.

Lazarin, N. M. and Pantoja, C. E. (2015). A robotic-agent platform for embedding software agents using raspberry pi and arduino boards. In *Anais do WESAAC 2015*, pages 13–20, Niterói. UFF.

Mahzan, N. N., Enzai, N. I. M., Zin, N. M., and Noh, K. S. S. K. M. (2018). Design of an Arduino-based home fire alarm system with GSM module. *Journal of Physics: Conference Series*, 1019(1):012079.

Pantoja, C. E., Jesus, V. S. d., Lazarin, N. M., and Viterbo, J. (2023). A Spin-off Version of Jason for IoT and Embedded Multi-Agent Systems. In *Intelligent Systems. BRACIS 2023*, pages 382–396, Cham. Springer Nature Switzerland.

Pantoja, C. E., Stabile, M. F., Lazarin, N. M., and Sichman, J. S. (2016). ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In *Engineering Multi-Agent Systems. EMAS 2016*, pages 136–155, Cham. Springer.

Perilla, F. S., Villanueva, G. R., Cacanindin, N. M., and Palaoag, T. D. (2018). Fire Safety and Alert System Using Arduino Sensors with IoT Integration. In *Proceedings of ICSCA'18*, page 199–203, Kuantan Malaysia. ACM.

Sichman, J. S., Demazeau, Y., and Boissier, O. (1992). When can knowledge-based systems be called agents? In *Anais do SBIA 92*, pages 172–185, Rio de Janeiro. SBC.

Souza de Jesus, V., Mori Lazarin, N., Pantoja, C. E., Vaz Alves, G., Ramos Alves de Lima, G., and Viterbo, J. (2023). An IDE to Support the Development of Embedded Multi-Agent Systems. In *The PAAMS Collection*, pages 346–358, Cham. Springer.

Yadav, R. and Rani, P. (2020). Sensor Based Smart Fire Detection and Fire Alarm System. In *Proceedings of AdChE 2020*, UPES, Dehradun, India.

Zhang, J. and Tao, D. (2021). Empowering Things With Intelligence: A Survey of the Progress, Challenges, and Opportunities in Artificial Intelligence of Things. *IEEE Internet of Things Journal*, 8(10):7789–7817.

Solução Multiagente para Prescrição de Diálise Peritoneal

Augusto Vaccarelli Costa[1], Fernando Falquetto Coelho[1], Lucas Alexandre Tavares[1], Anarosa Alves Franco Brandão[1], Rogério da Hora Passos[2,3], Maristela Carvalho da Costa[4,5]

[1]Departamento de Engenharia de Computação e Sistemas Digitais –
Escola Politécnica – Universidade São Paulo (USP)
São Paulo – SP – Brazil

[2]Davita Tratamento Renal e [3]Hospital Israelita Albert Einstein
São Paulo – SP – Brazil

[4]Instituto do Coração - HCFMUSP e [5]Hospital Santa Catarina
São Paulo – SP – Brazil

{augustovcosta,fernandofalquetto,lucas.alex2000,
anarosa.brandao}@usp.br

Abstract. *Chronic kidney disease, in its advanced stages, often necessitates the implementation of renal replacement therapies (RRT) to overcome the compromised kidneys' excretory functions. Among RRT modalities, peritoneal dialysis (PD) stands out for its flexibility and lower cost. Its effectiveness crucially relies on tailoring the treatment to the patient's unique conditions. Historically, prescribing PD is a very iterative process, employing heuristic principles and standardized assessments. We propose the development and validation of a Multi-Agent Solution for prescribing PD regimens, in order to optimize time and resources, in addition to improving patients' quality of life.*

Resumo. *A doença renal crônica, em seus estágios avançados, frequentemente requer a implementação de terapias de substituição renal (TSR) para realizar as funções excretoras comprometidas dos rins. Entre as modalidades de TSR, a diálise peritoneal (DP) se destaca por sua flexibilidade e custo mais baixo. Sua eficácia depende crucialmente da adequação do tratamento às condições únicas do paciente. Historicamente, a prescrição de DP é um processo bastante iterativo, que emprega princípios heurísticos e avaliações padronizadas. Este Artigo propõe o desenvolvimento e validação de uma Solução Multiagente para prescrição de regimes de DP, a fim de otimizar tempo e recursos, além de melhorar a qualidade de vida dos pacientes.*

1. Introdução

Doenças renais são a terceira causa de morte que mais cresce ao redor do mundo, espera-se que esse tipo de enfermidade continue afetando cada vez mais pessoas no mundo todo ao decorrer das próximas décadas. Essa tendência de crescimento já afeta diretamente não só a saúde de 850 milhões de pessoas, mas também os cofres públicos de economias globais: países como os Estados Unidos gastam atualmente 130 bilhões de dólares no tratamento e mitigação dos danos causados por doenças renais. Tratamentos como a hemodiálise são um dos grandes responsáveis pelos altos gastos, afinal, ela precisa ser realizada de duas a três vezes por semana durante toda a vida dos pacientes ou até a realização de um transplante renal. Na Tailândia, por exemplo, os

gastos com hemodiálise são responsáveis por 3% dos custos de saúde do país inteiro [Editorial board Nature, 2024].

Em seus estágios avançados, a doença renal crônica (DRC) frequentemente requer a implementação de terapias de substituição renal (TSR) para realizar as funções excretoras comprometidas dos rins [Grassmann et al, 2005]. Existem três principais modalidades de TSR: hemodiálise, DP e TSR contínua. Entre essas modalidades, a DP se destaca por sua flexibilidade, custo menor e pela possibilidade de ser realizada pelo próprio paciente em domicílio. Para garantir um bom tratamento de DP, é necessário que seja adequadamente ajustado às necessidades individuais de cada paciente para garantir a estabilidade da condição renal. Assim, com uma melhora na prescrição de DP, será possível obter retornos sanitários e econômicos significativos.

Dadas as características dinâmicas do sistema excretor humano, a adoção de agentes inteligentes para modelar seu funcionamento pode ser uma opção à personalização de tratamento. De fato, Pereira et al, (2022), definiram uma abordagem multiagente que simula o sistema endócrino humano. Este artigo propõe um sistema multiagente para personalização da prescrição de DP, que adota arquitetura similar à proposta por Pereira et al (2022) e soluções algébricas adotadas pela comunidade médica em prescrições iniciais e dados históricos de prescrições de DP de pacientes com DRC para o aprendizado dos agentes.

Este artigo está estruturado em 5 seções: introdução, conceitos básicos sobre TSR, trabalhos relacionados, a proposta de sistema e por fim os resultados obtidos até o momento.

2. Terapia de Substituição Renal (TSR)

As terapias de substituição renal são um conjunto de tratamentos médicos projetados para replicar as funções dos rins quando estes estão incapacitados devido a doença renal aguda ou crônica. Na ausência de uma função renal eficaz, substâncias tóxicas e líquidos podem se acumular no corpo, levando a condições potencialmente fatais que necessitam intervenção através da TSR.

A hemodiálise utiliza um dialisador externo que filtra o sangue fora do corpo, removendo toxinas e excesso de líquidos através de uma membrana semipermeável. O processo requer que o sangue seja circulado fora do corpo, tratado e então retornado, geralmente sendo realizado de 3 a 4 vezes por semana em sessões de 3 a 4 horas. A TSR contínua é tipicamente reservada para pacientes que estão em condições críticas, como aqueles em unidades de terapia intensiva. Esta técnica é usada principalmente em pacientes instáveis que não tolerariam as mudanças rápidas de fluido e solutos associadas com a hemodiálise convencional [Jameson *et al, 2018].*

Entre as modalidades de TSR, a DP se destaca por sua flexibilidade, custo menor e pela possibilidade de ser realizada pelo próprio paciente em domicílio, utilizando a membrana peritoneal como um meio natural de filtração para remover toxinas e excesso de fluidos do corpo (Fig. 1). Contudo, a eficácia da DP depende crucialmente da adequação do regime de tratamento às condições únicas do paciente, o que inclui: taxa de filtração glomerular residual, características da membrana peritoneal, volume de urina, bem como aspectos demográficos e clínicos variados.

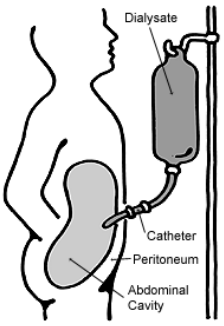


Figura 1: diagrama de DP [NKUDIC, NIDDKD e NIH, 2024]

Essa técnica é especialmente útil para pacientes com insuficiência renal crônica ou aguda que necessitam de tratamento contínuo para equilibrar os níveis de eletrólitos e remover toxinas do sangue. Na DP, uma solução dialítica estéril é infundida na cavidade peritoneal, a cavidade abdominal que envolve os órgãos internos. A solução de diálise contém um alto teor de glicose, que serve como um agente osmótico para puxar líquidos e solutos tóxicos do sangue através da membrana peritoneal. Com o tempo, a solução dialítica absorve os resíduos e é drenada do abdômen, sendo substituída por uma nova solução para continuar o processo de limpeza. Este ciclo de infusão e drenagem é repetido várias vezes ao dia ou é realizado continuamente, dependendo do método específico de DP utilizado.

Existem dois tipos principais de DP: a DP ambulatorial contínua (CAPD) e a DP automatizada (DPA). A CAPD é realizada manualmente pelo paciente ou cuidador, com trocas de fluido ocorrendo várias vezes ao longo do dia. Por outro lado, a DPA utiliza uma máquina chamada cicladora que automaticamente realiza as trocas de fluido durante a noite enquanto o paciente dorme, permitindo maior liberdade durante o dia.

A DP é frequentemente escolhida por oferecer maior conveniência e flexibilidade em comparação com a hemodiálise, que requer visitas frequentes a um centro de diálise. Além disso, como é geralmente administrada em casa, a DP permite que os pacientes mantenham um cronograma de trabalho ou atividades pessoais mais normal. Também é considerada mais suave em termos de equilíbrio de fluidos e estabilidade hemodinâmica, o que é particularmente vantajoso para pacientes idosos ou com condições cardíacas.

Apesar das vantagens, DP possui contraindicações e não pode ser prescrita em todos os casos de falência renal. A TSR com DP também exige que os pacientes mantenham uma rigorosa higiene para evitar infecções, especialmente peritonite, uma infecção potencialmente grave da cavidade abdominal. Os pacientes e cuidadores devem ser bem treinados no procedimento adequado de troca para minimizar o risco de contaminação.

3. Trabalhos Relacionados

A literatura sobre personalização da prescrição de diálise peritoneal tem como foco, principalmente, modelagem matemática [Galli et al., 2011] ou monitoramento remoto de pacientes [Mila Manani et al, 2018, John e Jha, 2019]. Mais recentemente surgiram soluções que fazem uso de técnicas de inteligência artificial para apoiar a personalização de tratamentos médicos. Chakraborty et al.(2024) apresentam um mapeamento sobre o

uso de técnicas de aprendizado de máquina neste assunto. Burlacu et al (2020) fizeram uma revisão da literatura sobre o uso de técnicas de inteligência artificial e aprendizado de máquina para diálises e transplantes de rim. No caso específico de DP, as soluções inteligentes se referiam à classificação de tratamentos por perfil de pacientes, à previsão de risco de infecções e a acidentes vasculares em pacientes.

Especificamente para personalização de prescrição de tratamentos, encontrou-se o trabalho de Pereira et al (2022). Nele foi desenvolvida uma solução multiagente para prescrição personalizada de dosagem de insulina em pacientes com diabetes tipo 1. O modelo personaliza prescrições para um intervalo de até 8 horas, impactando positivamente na vida de pacientes diabéticos. A solução usa um modelo matemático que simula o funcionamento do pâncreas para acelerar o aprendizado de agentes inteligentes alimentados com dados históricos de nível de glicose e outras informações sobre o paciente. Neste trabalho adotamos uma arquitetura similar, dado que ambos os sistemas a serem modelados são dinâmicos e de complexidade bastante próxima.

4. Proposta de Arquitetura

O desenvolvimento do modelo, que emprega agentes reativos, passa por três etapas diferentes no quesito de arquitetura: palpite inicial, treinamento com testes sintéticos e treinamento contínuo após disponibilização. Todas elas possuem dois elementos em comum que tipicamente constituem modelos de aprendizagem por reforço, que são os agentes aprendiz e recomendador.

O agente aprendiz é o agente responsável por gerar e atualizar o modelo de recomendação utilizado na geração das prescrições do tratamento de DP. Para o primeiro ciclo de treinamento, é necessário que ele possua não só os parâmetros de diversos pacientes em seu banco de dados, mas também um palpite inicial de prescrição, resultante da modelagem matemática da prescrição inicial [Galli et al., 2011]. Um diagrama da arquitetura proposta está apresentado na Fig. 2.

Com o modelo de recomendação ajustado pelo agente aprendiz, o agente recomendador gera os parâmetros necessários para a prescrição da DP, que incluem: tempo de duração, volume de líquido dialisante e concentração de solutos.

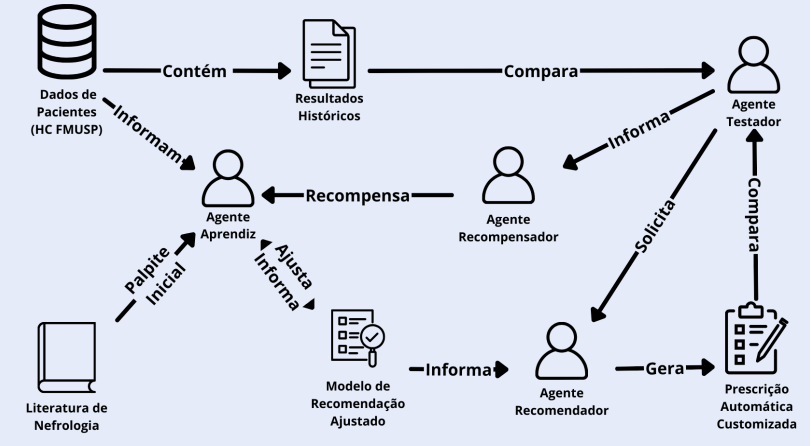


Figura 2. Arquitetura Inicial do Modelo. Fonte: Autoral

Após o palpite inicial e a prescrição do primeiro tratamento, um agente testador será inserido no sistema, com o objetivo de realizar o treinamento do modelo. Ele solicitará diagnósticos fornecendo dados de pacientes reais ao sistema e, após comparar as prescrições com resultados históricos de pacientes reais, fará ajustes no modelo de acordo com sua árvore de decisões.

Durante o treinamento, as prescrições geradas pelo sistema serão continuamente validadas por especialistas, a fim de refinar as recompensas do agente aprendiz. Quando o modelo for capaz de gerar prescrições com acurácia e precisão adequadas, ele poderá ser disponibilizado para uso e o testador sintético será substituído pelos médicos usuários do sistema, os quais serão responsáveis por solicitar os parâmetros do tratamento e fornecer feedbacks quanto às prescrições para que o aprendizado continue.

Após o período de treinamento, o projeto será voltado à disponibilização de uma plataforma virtual que encapsule o modelo e continue gerando prescrições que serão validadas por médicos voluntários e pacientes reais, enquanto isso o modelo continuaria treinando com pacientes reais e podendo manter uma base de dados que distribui prescrições personalíssimas com base no histórico individual dos pacientes.

5. Resultados Preliminares e Trabalhos Futuros

Neste estágio, a pesquisa concentrou-se na concepção teórica de um sistema multiagente para prescrição de DP. Embora ainda não seja possível realizar testes ou validações empíricas, foi desenvolvido um modelo conceitual com base na literatura clássica para definição dos problemas e em publicações recentes para buscar soluções que sejam capazes de ajustar os parâmetros do líquido dialisante da DP. Para o desenvolvimento e treinamento do modelo, espera-se utilizar o framework *SPADE* da linguagem *python*, ela permite e facilita a comunicação entre os agentes do sistema, os quais podem permanecer em execução através da nuvem.

Com a arquitetura concebida, foram iniciados a definição de casos de uso da plataforma, bem como a prototipação de telas para seu uso efetivo por médicos nefrologistas. Nela, o médico seria capaz de acompanhar seus pacientes, gerar prescrições e fornecer *feedback* à prescrição do modelo. Para gerar prescrições, bastaria inserir os dados básicos do paciente (aos que estão iniciando tratamento): altura, peso e idade, além de índices como a filtração glomerular residual e nitrogênio ureico no sangue.

6. Referências

Nature Editorial Board. (2024) “Time to sound the alarm about the hidden epidemic of kidney disease”. In: Springer Nature. v. 628, p. 7-8. 03/04/2024 https://doi.org/10.1038/d41586-024-00961-5

Grassmann A, Gioberge S, Moeller S, et al. (2005) “ESRD patients in 2004: global overview of patient numbers, treatment modalities and associated trends”. Nephrology Dialysis Transplantation, v. 20, i. 12, p. 2587–2593.

Galli, E.G., Taietti, C. e Borghi, M. (2011) “Personalization of Automated Peritoneal Dialysis Treatment Using a Computer Modeling System”, *Advances in Peritoneal*

Dialysis, Vol. 27, 90–96, 2011.

Milan Manani, S., Crepaldi, C., Giuliani, A., Virzì, G. M., Garzotto, F., Riello, C., ... & Ronco, C. (2018) "Remote monitoring of automated peritoneal dialysis improves personalization of dialytic prescription and patient's independence", In: Blood Purification, 46(2), 111-117.

John, O., & Jha, V. (2019) "Remote patient management in peritoneal dialysis: an answer to an unmet clinical need". Remote patient management in peritoneal dialysis, 197, 99-112.

American Journal of Kidney Diseases. (2006) "Clinical Practice Guidelines for Peritoneal Dialysis Adequacy",Volume 48, Suplemento 1, 98-129,

Lindholm, B.; Krediet, R. T. (2009) "Peritoneal Dialysis: From Basic Concepts to Clinical Excellence". In: Contributions to Nephrology. Basel: Karger, vol. 163.

Chakraborty, C.; Bhattacharya, M.; Pal, S.; Lee, S.-S. (2024) "From machine learning to deep learning: Advances of the recent data-driven paradigm shift in medicine and healthcare". Current Research in Biotechnology, v. 7. ISSN 2590-2628.

Aragão Pereira, J.P. Franco Brandão, A.A. Bevilacqua, J.d.S. Côrrea-Giannella, M.L.C. (2022) "A Multi-Agent Approach Used to Predict Long-Term Glucose Oscillation in Individuals with Type 1 Diabetes". Appl. Sci. 2022, 12, 9641. https://doi.org/10.3390/app12199641

M. Zhang, Y. Hu, and T. Wang. (2005) "Selection of peritoneal dialysis schemes based on multi-objective fuzzy pattern recognition". In: Sheng Wu Yi Xue Gong Cheng Xue Za Zhi, vol. 22, no. 2, pp. 335–338, 2005

Alexandru Burlacu A., Adrian Iftene, Daniel Jugrin, Iolanda Valentina Popa, Paula Madalina Lupu, Cristiana Vlad, Adrian Covic. (2020) "Using Artificial Intelligence Resources in Dialysis and Kidney Transplant Patients: A Literature Review". In: BioMed Research International, Volume 2020, Article ID 9867872, disponível em https://doi.org/10.1155/2020/9867872

NKUDIC, NIDDKD and NIH. (2024) "dialys.gif". In: National Kidney and Urologic Diseases Information Clearinghouse, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, USA - http://kidney.niddk.nih.gov/kudiseases/pubs/yourkidneys/images/dialys.gif, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1490875, 2024

Jameson, J.; Fauci, A. S.; Kasper, D. L.; Hauser, S. L.; Longo, D. L.; Loscalzo, J (2018) "Acute Kidney Injury". In: Jameson, J.; Fauci, A. S.; Kasper, D. L.; Hauser, S. L.; Longo, D. L.; Loscalzo, J. (Eds.). Harrison's Principles of Internal Medicine. 20th ed. New York: McGraw-Hill Education, 2018. p. 2099-2111.

Russell, J. S.; Norvig, P. (2010) "Artificial Intelligence A Modern Approach", Pearson Education, 3ª edição.

Python. (2024) Disponível em: https://www.python.org/

SPADE. Smart Python Agent Development Environment (2020) Disponível em: https://spade-mas.readthedocs.io/en/latest

Criação de Agentes BDI a partir de Modelos do UPPAAL*

João Vicente Markovicz[1], Gleifer Vaz Alves[1], André Pinz Borges[1]

[1]Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
Ponta Grossa – PR – Brasil

joao.080603@alunos.utfpr.edu.br gleifer@utfpr.edu.br

apborges@utfpr.edu.br

Abstract. *This paper presents the extension of ethical models as automata built using the UPPAAL tool. These models are translated into prototypes of agents using the BDI architecture. This translation aims to establish a direct relationship between formal modelling and agent implementation.*

Resumo. *Este artigo apresenta a extensão de modelos éticos em forma de autômatos construídos na ferramenta UPPAAL. Tais modelos são traduzidos em protótipos de agentes BDI. Esta tradução busca estabelecer uma relação direta entre a modelagem formal e a implementação dos agentes.*

1. Introdução

Sistemas inteligentes para automação de tarefas simples ou complexas (como o controle de um veículo autônomo) estão cada vez mais presentes no nosso dia-a-dia e demandam tomadas de decisão conforme princípios éticos [Alves et al. 2021]. Esta conformidade garante que tais sistemas operem de maneira responsável, seguindo regras éticas. Por exemplo, um VA não deve tomar decisões imprudentes perante as normas de trânsito, de modo a proteger as vidas dos seus ocupantes e demais envolvidos. A automação dos sistemas pode ser desenvolvida com uso de agentes inteligentes [Alves et al. 2021]. Os aspectos éticos destas soluções podem ser baseados em trabalhos como Bench-Capon [Bench-Capon 2020], que disserta sobre três abordagens éticas aplicadas em sistemas autônomos (Consequencialismo, Deontologia e Ética das Virtudes) e como elas poderiam ser usadas para tomar decisões. Os comportamentos dos agentes pode ser modelado como autômatos, o que leva ao trabalho de [Markovicz and Alves 2023], onde é apresentada uma modelagem formal desses três modelos éticos usando autômatos temporais com a ferramenta *UPPAAL* [Bengtsson et al. 1996]. O objetivo deste artigo é apresentar uma extensão dos modelos construídos em [Markovicz and Alves 2023], e a tradução dos modelos formais para geração de protótipos visando a programação de agentes seguindo o paradigma de agentes BDI.

2. Modelos Éticos e Agentes BDI

Bench-Capon descreve as abordagens éticas Consequencialista, Deontológica e Ética das Virtudes, e faz apontamentos sobre aplicar cada abordagem em agentes éticos. Ele utiliza

um cenário específico, a história "A Cigarra e a Formiga", onde a `Formiga` trabalha no verão e a `Cigarra` joga. No inverno, a `Formiga` tem comida suficiente para si e uma sobra que pode ser usada para uma festa no fim da estação. Já a `Cigarra` dependerá da ajuda da formiga para sobreviver. Este dilema ético que pode tomar diferentes caminhos dependendo da abordagem escolhida.

Em [Markovicz and Alves 2023] foram construídos autômatos para cada uma das abordagens utilizando o *UPPAAL* [Bengtsson et al. 1996], uma ferramenta que permite a modelagem e verificação formal de autômatos temporais de forma fácil e dinâmica, possibilitando até mesmo a simulação do autômato construído. Nesse trabalho, será usada uma nova versão dos modelos construídos, descritos na Seção 3.

Conforme declarado, esse trabalho busca estabelecer uma conexão entre os modelos formais previamente construídos e a sua implementação usando agentes BDI [Bratman 1987], que é um tipo de arquitetura de agentes baseada na estrutura: *Beliefs* (Crenças), *Desires* (Objetivos ou Desejos) e *Intentions* (Intenções ou Planos). As crenças representam o conhecimento que o agente possui sobre si ou o ambiente que está inserido. Os objetivos estabelecem aquilo que o agente pretende alcançar. Os planos organizam as ações que o agente deve executar para chegar aos seus objetivos, esses planos tem o seguinte formato: $TriggerEvent : \{Context\} \leftarrow Body$, onde ***Trigger Event*** é definido pela adição de numa nova crença ou objetivo, o *Context* é determinado por um conjunto de crenças e o *Body* é uma sequência de ações a serem executadas pelo agente [Bordini et al. 2007].

3. Modelagem formal das abordagens éticas

Os modelos descritos em [Markovicz and Alves 2023] são distintos na etapa da decisão de cada um. O modelo Deontológico usa na decisão apenas a variável `played_times`, que conta a quantidade de vezes que o agente joga. Já o modelo Consequencialista usa a variável `balance`, que incrementa quando um agente trabalha e decrementa quando ele joga. Por fim, o modelo da Ética das Virtudes, faz o uso de ambas as variáveis para a decisão. Todos esses modelos possuem um agente único que pode ser instanciado várias vezes e se comportar como formiga ou cigarra.

Neste artigo será utilizado apenas o modelo da Ética das Virtudes, que pode ser visto como o mais robusto dentre os três. Aqui em uma nova versão onde os papéis (formiga e cigarra) foram separados gerando dois agentes distintos, um representando a `Formiga` e outro a `Cigarra`. É importante notar que foram usadas técnicas de abstração para desenvolver modelos que representem o que é necessário para o entendimento do comportamento de cada agente do sistema, tendo em vista o processo de tradução que será descrito na seção 4.

3.1. Autômato Formiga

No autômato da `Formiga` (Fig. 1), tem-se um agente que, saindo do estado inicial, pode ir para o estado **working** (a variável `food` aumenta 2), após isso, pode ir para **eating** (`food diminui 1`). Então, este agente pode ir para **feasting**, que seria uma festa no fim da estação (`food diminui 1`) ou pode receber um pedido por comida de um agente `Cigarra`, indo para **negotiating**. Neste estado, o agente ativa um outro autômato que acessa as informações da `Cigarra` e faz o processo de decisão, ao voltar para a

Formiga, esta envia uma mensagem à Cigarra aceitando (**give_food**) ou recusando o pedido (**refuse_food**).

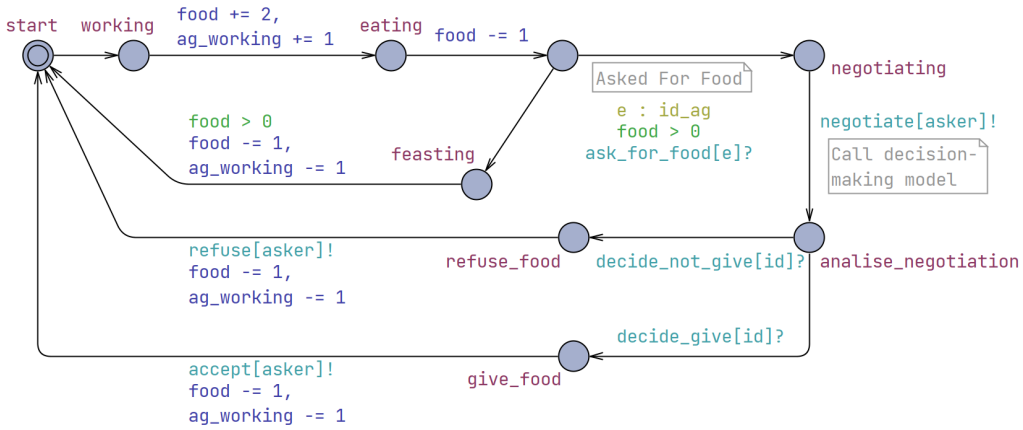


Figura 1. Autômato Formiga (UPPAAL)

3.2. Autômato Cigarra

Já o autômato da Cigarra (Fig. 2), saindo do estado inicial, possui uma escolha entre trabalhar ou jogar (respectivamente os estados **working** e **playing**, com um peso de 50% para cada). Além disso, são atualizadas as variáveis played_times (incrementa 1 caso o agente escolha jogar) e balance (decrementa 1 caso o agente jogue, incrementa 1 caso o agente trabalhe).

Caso o agente decidir trabalhar, segue para os estados **eating** e **feasting** similarmente ao que ocorre com a Formiga. Caso escolha jogar, vai para o estado **ask_for_food** e deverá recorrer a ajuda de algum agente Formiga utilizando troca de mensagens. A Cigarra espera até receber um sinal refuse (o pedido foi recusado e a Cigarra morre) ou o sinal accept (a Cigarra recebe a comida chegando a **received_food**, segue para **eat_given_food**, então usando a comida volta ao estado inicial).

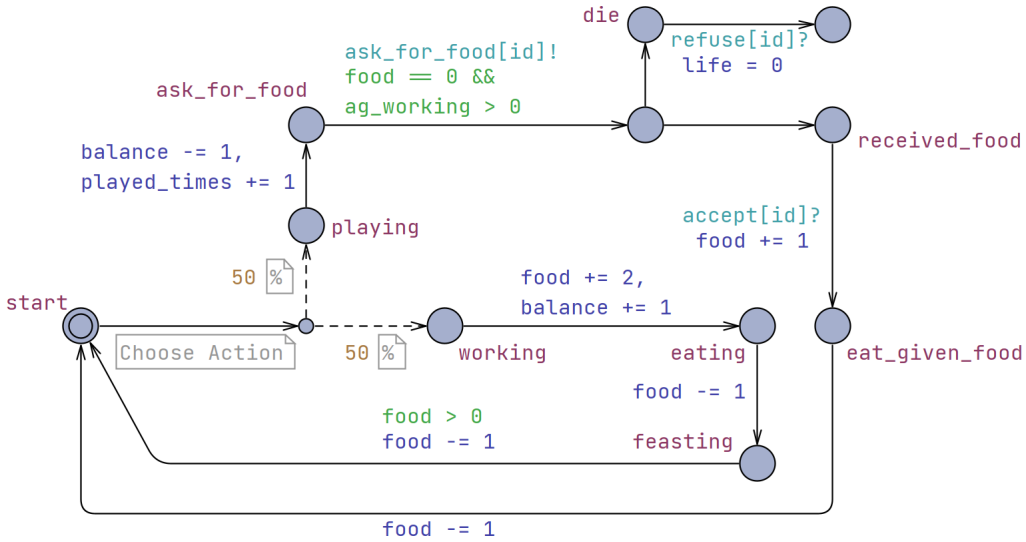


Figura 2. Autômato Cigarra (UPPAAL)

4. Tradução dos Modelos para Agentes BDI

Com os autômatos previamente apresentados, busca-se estabelecer uma conexão entre o modelo ético construído e a sua devida implementação como agentes BDI. Neste momento, é descrita uma tradução para um formato comumente utilizado por linguagens de agentes BDI, mas sem determinar uma linguagem alvo específica.

Nos protótipos 1 e 2 tem-se a definição das crenças e objetivos dos agentes `Formiga` (`Ant`) e `Cigarra` (`GH`) respectivamente. Nesse caso, as crenças foram mapeadas com base nas variáveis declaradas no UPPAAL e os objetivos são os estados do autômato. Cabe ressaltar que foram colocados apenas os objetivos iniciais, visto que outros deverão ser adicionados à medida que o agente executa seus planos.

A `Formiga` possui as crenças iniciais *life(1)* que controla a vida da formiga, e *food(0)* que controla a quantidade de comida que ela possui. Já como objetivo inicial tem-se apenas *start*.

```
1 Beliefs: life(1), food(0);
2 Goals: start;
```

Protótipo 1. Crenças e Objetivos - Agente Formiga

A `Cigarra`, possui as mesmas crenças *life(1)* e *food(0)*, além de *balance(0)* e *played_times(0)*, que eram as variáveis de controle presentes no autômato construído. Para objetivo inicial tem-se apenas *start*.

```
1 Beliefs: life(1), food(0), balance(0), played_times(0);
2 Goals: start;
```

Protótipo 2. Crenças e Objetivos - Agente Cigarra

Nos protótipos 3 e 4 há a definição dos planos para ambos os agentes. Para efetuar a tradução foi utilizado como base o mecanismo ilustrado na Figura 3, onde os estados e transição do autômato são usados para gerar o respectivo plano do agente. Especificamente, obtém-se o seguinte plano: $State1 : \{Conditions\} \leftarrow Commands,\ send(Agent, Communication),\ State2$; onde `State 1` determina o *Trigger Event* do plano, o *Context* do plano é obtido a partir das `conditions`, enquanto o *Body* é determinado pelos `commands`, `communication` e ainda o próprio `State 2`. Note que nem toda transição do autômato necessariamente tem os três elementos: `conditions`, `commands` ou `communication`.

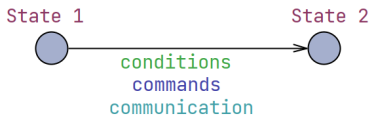


Figura 3. Visão geral do Autômato

No protótipo 3, há a definição dos planos do agente `Formiga`:

- ***start*** - O agente inicia, no contexto não possui pré-requisito, no corpo adiciona o objetivo *working*;

- ***working*** - O agente trabalha, no contexto seleciona a crença *food*, no corpo aumenta *food* em 2 e adiciona o objetivo *eating*;
- ***eating*** - O agente utiliza sua comida, no contexto seleciona a crença *food*, no corpo diminui 1 em *food*;
- ***negotiate(GH)*** - Recebe o pedido de negociação do agente `GH`, utiliza uma função do ambiente *analise_negotiation()*, que irá analisar o pedido com base nas informações de `GH`, assim como é realizado no autômato;
- ***give_food(GH)*** - Plano que dá comida ao agente `GH`, no contexto é necessário possuir *food(1)*, no corpo envia uma mensagem ao `GH` informando que o pedido foi aceito, então é adicionado o objetivo *start*;
- ***refuse_food(GH)*** - Plano que recusa comida ao agente `GH`, no corpo envia uma mensagem a ele recusando o pedido, e então adiciona o objetivo *feasting*;
- ***feasting*** - Realiza a festa, no contexto seleciona a crença *food* que precisa ser maior que 0, no corpo diminui 1 em *food* e adiciona o objetivo *start*.

```
start : {true} <- working;
working : {food(X)} <- +food(X+2), eating;
eating : {food(X)} <- food(X-1);
negotiate(GH) : {true} <- analise_negotiation(GH);
give_food(GH) : {food(1)} <- send(GH, accept), start;
refuse_food(GH) : {true} <- send(GH, refuse), feasting;
feasting : {food(X) and X>0} <- food(X-1), start;
```

Protótipo 3. Planos - Agente Formiga

No protótipo 4, há a definição dos planos do agente `Cigarra`:

- ***start*** - O agente inicia, no contexto não possui pré-requisito, no corpo adiciona o objetivo *choose_action*;
- ***choose_action*** - No contexto não possui pré-requisito, no corpo utiliza a função do ambiente *choose_action()* para decidir randomicamente se irá trabalhar ou jogar;
- ***working*** - O agente trabalha, no contexto seleciona as crenças *food* e *balance*, no corpo aumenta *food* em 2, *balance* em 1 e adiciona o objetivo *eating*;
- ***playing*** - O agente joga, no contexto seleciona as crenças *balance* e *played_times*, no corpo diminui 1 em *balance*, aumenta 1 em *played_times* e adiciona o objetivo *ask_for_food*;
- ***ask_for_food*** - O agente pede por comida, no contexto precisa ter a crença *food(0)* e a função do ambiente *ag_working()* precisa retornar *true*, no corpo envia uma mensagem ao `Ant` pedindo por uma negociação;
- ***received_food*** - O agente recebe comida, no contexto precisa da crença *accept* e seleciona a crença *food*, no corpo *food* aumenta em 1 e adiciona o objetivo *eat_given_food*;
- ***eat_given_food*** - O agente usa a comida que recebeu, no contexto seleciona a crença *food*, no corpo diminui 1 em *food* e adiciona o objetivo *start*;
- ***eating*** - O agente utiliza sua comida, no contexto seleciona a crença *food*, no corpo diminui 1 em *food* e adiciona o objetivo *feasting*;
- ***feasting*** - Realiza a festa, no contexto seleciona a crença *food* que precisa ser maior que 0, no corpo diminui 1 em *food* e adiciona o objetivo *start*;

- ***die*** - O agente morre, no contexto precisa da crença *refuse* indicando que o pedido por comida foi recusado, no corpo atualiza *life* para 0;

```
start : {true} <- choose_action;
choose_action : {true} <- choose_action();
working : {food(X), balance(Y)} <- food(X+2), balance(Y+1), eating ;
playing : {balance(X), played_times(Y)} <-
            balance(X-1), played_times(Y+1), ask_for_food;
ask_for_food : {food(0) and ag_working()} <- send(Ant, negotiate(GH));
received_food : {accept and food(X)} <- food(X+1), eat_given_food;
eat_given_food : {food(X)} <- food(X-1), start;
eating : {food(X)} <- food(X-1), feasting;
feasting : {food(X)} <- food(X-1), start;
die : {refuse} <- life(0);
```

Protótipo 4. Planos - Agente Cigarra

5. Considerações Finais

Nesse artigo os modelos éticos construídos em [Markovicz and Alves 2023] foram estendidos, de forma a favorecer a transição para um protótipo usando o paradigma de agentes BDI. Como trabalho futuro, pretende-se implementar o protótipo apresentado além de suas ações internas, em uma linguagem para programação de agentes, como JASON [Bordini et al. 2007] ou GWENDOLEN [Dennis and Müller 2008]. Além de realizar uma extensão para agentes que representem veículos autônomos em cenários específicos que necessitem tomada de decisão ética.

Referências

Alves, G., Dennis, L., and Fisher, M. (2021). An Agent-based architecture with support to Ethical Decisions on a Road Traffic Scenario. Publisher: Zenodo.

Bench-Capon, T. (2020). Ethical approaches and autonomous systems. *Artificial Intelligence*, 281:103239.

Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., and Yi, W. (1996). UPPAAL — a tool suite for automatic verification of real-time systems. In Alur, R., Henzinger, T. A., and Sontag, E. D., editors, *Hybrid Systems III*, number 1066 in Lecture Notes in Computer Science, pages 232–243. Springer Berlin Heidelberg.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Cambridge: Cambridge, MA: Harvard University Press.

Dennis, L. and Müller, B. (2008). Gwendolen: A bdi language for verifiable agents.

Markovicz, J. V. and Alves, G. V. (2023). Modelagem formal de abordagens éticas para comportamento de agentes. *In Anais do Workshop-Escola de Informática Teórica (WEIT)*, pages 134–138.

Implementação de módulos de kernel Linux para simulação e proveniência de Sistemas Multiagentes Embarcados

Bruno Policarpo Toledo Freitas[1], **Carlos Eduardo Pantoja**[1]

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
Av. Maracanã, 229 - Maracanã – Rio de Janeiro/RJ - CEP: 20271-110

Abstract. *A Multi-agent system (MAS) is a group of autonomous agents capable of deliberating and acting upon the world. A common model to implement those systems is the Belief-Desire-Intention (BDI), based upon human reasoning. Agents can also be used to implement cognition for embedded and cyber-physical systems. Given the abstract nature of agent models, various middleware have been proposed to ease the development of these systems. However, considering embedded systems, these abstractions can negatively impact the systems and miss opportunities of using operating systems features for embedded MAS development. This work proposes a different way of developing these systems by using Linux kernel modules. In this work, we show the current development of two modules: a serial channel emulator and a provenance module for real Embedded MAS devices. The serial channel emulator presented seamlessly connects an agent's reasoning to a simulator. The provenance module aims to transparently capture data exchanged between the agent and the embedded device.*

Resumo. *Sistemas Multiagentes (SMA) são um grupo de agentes capazes de deliberar e agir sobre um ambiente de acordo de forma autônoma. Um modelo bastante comum de implementar tais sistemas é o modelo Belief-Desire-Intention (BDI), baseado no raciocínio humano. Agentes também podem ser utilizados para o desenvolvimento de softwares embarcados e ciberfísicos a fim de prover uma camada cognitiva aos mesmos. Tendo em vista a natureza abstrata do modelo de agentes, diversos middlewares tem sido propostos para fornecer as abstrações necessárias para implementação desses sistemas. Todavia, no caso de sistemas embarcados, a utilização de abstrações de mais alto nível podem impactar negativamente o sistema e perde-se oportunidades de explorar características do sistema operacional no desenvolvimento desses sistemas. Nesse sentido, este artigo apresenta o andamento do desenvolvimento de dois módulos de kernel Linux para para SMAs embarcados: um para simulação de SMAs embarcados e outro para Proveniência. Para a simulação de SMAs embarcados, o módulo criado fornece um canal de comunicação virtual e genérico entre um SMA e simuladores, sendo mostrado como o canal pode ser utilizado para tal finalidade. Já o módulo de proveniência visa capturar os fluxos de dados de um canal de comunicação serial de um SMA embarcado executado no mundo real de forma transparente.*

1. Introdução

Um sistema multiagente (SMA) é um grupo de agentes autônomos capaz de receber percepções e agir em um ambiente virtual ou computacional [Michel et al. 2009]. Agentes diferem de softwares convencionais pois são independentes, adaptáveis, pró-ativos, com cognição, sendo capazes também de colaborar entre si [Hübner et al. 2004]. O modelo cognitivo de agentes *Belief-Desire-Intention* (BDI) [Bratman 1987], baseado no raciocínio humano, costuma ser utilizado para implementar a camada de raciocínio, permitindo que eles deliberem sobre quais objetivos atingir, baseados em crenças, desejos e intenções.

Os SMAs também têm sido utilizado para adicionar uma camada cognitiva em sistemas ciberfísicos, IoT, e computação na borda [Karaduman et al. 2023]. Também têm sido usados para o desenvolvimento de sistemas robóticos [Silvestre et al. 2023], criando novas camadas de abstrações ou middleware para fazer com que os agentes consigam controlar recursos de hardware a fim de agir no mundo real [Dal Moro et al. 2022b, Dal Moro et al. 2022a]. A criação de middleware para diversos domínios de problema é recorrente, e sua definição varia conforme a área mas, de forma geral, é a ponte entre uma aplicação e um sistema operacional, facilitando o seu desenvolvimento, aumentando a escalabilidade, e facilitando sua manutenção e automação [Gazis and Katsiri 2022]. Diversos middlewares tem sido propostos na literatura para facilitar o desenvolvimento de SMAs, tais como o SPADE3 [Palanca et al. 2020], assim como formas de adaptar middlewares robóticos, tais como o Robot Operating System (ROS) [Open Robotics 2023b], para adicionar uma camada de raciocínio ao sistema [Gavigan and Esfandiari 2021].

Nesses casos, tais middlewares são construídos em cima de camadas de software a fim de prover as abstrações pretendidas. Porém, ao se olhar nessas abstrações, perde-se a oportunidade de explorar características do hardware e do sistema operacional para o desenvolvimento de SMAs. Especificamente, este trabalho irá apresentar o andamento de dois trabalhos sobre SMAs embarcados baseados em sistemas operacionais: integração de simuladores e proveniência de SMAs embarcados. O primeiro trabalho irá apresentar o estado atual de integração de SMAs com simuladores, no caso, o simulador robótico WeBOTS [Michel 1998]. Já o segundo apresenta uma proposta de solução para captura de proveniência de SMAs sistemas embarcados, de forma a permitir que a comunicação entre um SMA com microcontroladores seja salva de forma invisível e sem alteração do SMA.

Este trabalho está organizado da seguinte forma: na Seção 2 mostra o andamento do trabalho sobre o uso de simuladores para SMAs embarcados utilizando a arquitetura ARGO. A Seção 3 mostra o andamento do trabalho acerca de um módulo de proveniência para SMAs embarcados. Finalmente, A Seção 4 faz uma discussão de trabalhos apresentados.

2. Simuladores

Esta seção irá apresentar o andamento de trabalhos de integração de simuladores para o desenvolvimento de SMAs embarcados. Especificamente, será abordado o andamento atual da integração com simuladores robóticos. Primeiro, será dado um breve panorama do uso de simuladores para o desenvolvimento desses sistemas, para então apresentar o andamento atual da integração com o simulador robótico WeBOTS.

Uma solução popular para o desenvolvimento de sistemas robóticos no geral é usar a plataforma *Robot Operating System* (ROS) [Open Robotics 2023b] para o desenvolvimento do sistema robótico em conjunto com o simulador Gazebo [Open Robotics 2023a] para realizar teste e avaliação da solução. Existem trabalhos que visam realizar o desenvolvimento do software robótico por meio de um SMA descrito em Jason, em que o ROS é utilizado como uma ponte entre o agente e uma camada de aplicação, onde esta última é a responsável pela comunicação com o ambiente de simulação ou o mundo real [Gavigan and Esfandiari 2021]. Porém, tal abordagem ainda precisa da construção dessa camada de aplicação específica para cada domínio de problema, além de ser computacionalmente mais custosa por possuir uma hierarquia com três camadas de abstração de software.

Em contrapartida, pode-se utilizar o próprio kernel do sistema operacional como uma camada de ligação do raciocínio do agente com um simulador. Essa configuração é mostrada na Figura 1. Nesse caso, o raciocínio do agente não precisa ser alterado e o simulador pode, a priori, ser qualquer um. O canal de comunicação serial emulado já existe e é funcional, assim como diversos simuladores já foram usados.

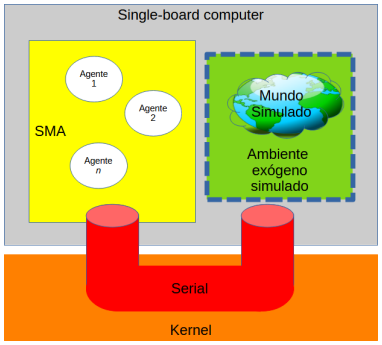


Figura 1. Canal de comunicação do agente com simulador via sistema operacional

As Listagens 1 e 2 ilustram como o canal é utilizado para conectar o raciocínio do agente ARGO com um simulador. Neste exemplo, foi utilizado o simulador WeBOTS para simular um protótipo físico 2WD. O agente ARGO precisa somente indicar como crença inicial a porta emulada de conexão, no caso *ttyEmulatedPort0*. Já no simulador é necessário mapear as ações correspondentes *goAhead*, *goBack*, *goLeft*, e *goRight* em comandos propícios do simulador e o comando *getPercepts* para retornar os valores dos sensores.

Listagem 1. ”Código do agente Jason”

```
/* Initial beliefs and rules */
serialPort(ttyACM0).

/* Initial goals */

!start.
/* Plans */
+!start: serialPort(Port) <- argo.port(Port); argo.
    percepts(open); argo.limit(750); !rollOut.

+!rollOut: dLeft(DL) & dRight(DR) & (DR >= 20) & (DL
    >= 20) & not wall
                          <- argo.act(goAhead);
    .wait(500); !rollOut.

+!rollOut: dLeft(DL) & dRight(DR) & ((DR < 20) | (DL
    <20)) & ((DR >= 10) | (DL >=10)) & not wall <-
    argo.act(stop); +wall; .wait(500); !rollOut.

+!rollOut: dLeft(DL) & dRight(DR) & ((DR < 10) | (DL
    <10)) & not wall
                          <- argo.act(goBack); +
    wall; .wait(500); !rollOut.

+!rollOut: dLeft(DL) & dRight(DR) & wall & ((DR < 30)
     | (DL <30) | (DR>100) | (DL>100))
            <- argo.act(goRight); .wait(500); !
    rollOut.

+!rollOut: dLeft(DL) & dRight(DR) & wall & ((DR >=
    30) | (DL >= 30))
                          <- -wall; !rollOut.

-!rollOut.

+dLeft(DL)  <- .print("Left Distance: ",DL).
+dRight(DR) <- .print("Right Distance: ",DR).
```

Listagem 2. ”Código do simulador 4WD”

```
if ( ! strcmp( javino_received_msg , "getPercepts"
    ) ){

  // left distance sensor value
  float d1 = wb_distance_sensor_get_value( ds[0]
      );

  // right distance sensor value
  float d2 = wb_distance_sensor_get_value( ds[1]
      );

  // Composing percepts message to send to Javino
  sprintf(percepts_msg,
    "dLeft(%.1f);dRight(%.1f);",
    d1, d2 );

  javino_send_msg( exogenous_port,
     percepts_msg);

  free( javino_received_msg );

} else if ( ! strcmp( javino_received_msg , "
    goAhead" ) ){

  left_speed = 1.0;
  right_speed = 1.0;

} else if ( ! strcmp( javino_received_msg , "
    goRight" ) ){

  left_speed = 1.0;
  right_speed = 0.0;

} else if ( ! strcmp( javino_received_msg , "
    goBack" ) ){

  left_speed = -1.0;
  right_speed = -1.0;

}
```

3. Proveniência de SMAs embarcados

Proveniência de um produto de dados informa sobre os processos e dados usados para gerá-lo [Freire et al. 2008], dessa maneira possibilitando reproduzir e validar experimentos científicos. Diferentemente do problema de simulação apresentado na Seção 2, um experimento no mundo real com SMAs embarcados gera um fluxo de dados oriundos dos sensores e as ações enviadas pelo SMA ao hardware. Nesse caso, os dados gerados são salvos apenas se houver uma instrumentação específica dentro do SMA.

De acordo com a arquitetura ARGO de SMAs embarcados, existe um ciclo bem definido de fluxos de dados onde, primeiro, o SMA recebe dados dos sensores por meio de um *getPercepts* e, de acordo com as crenças atualizadas pelos sensores, envia ações por meio de diretivas *act* do Jason. Ou seja, em termos de proveniência, o importante é capturar os dados trocados nesse ciclo. Dessa maneira, esses fluxos de dados podem ser usados posteriormente para depuração ou até mesmo para reproduzir o experimento com a ajuda de um simulador.

Para isso, está sendo desenvolvido um módulo de kernel Linux responsável por ”espionar”periodicamente o canal de comunicação serial para capturar os dados trafegando nele. Tal abordagem torna transparente para o SMA o processo de captura de proveniência - ou seja, o SMA não precisa realizar logging nem precisa ser alterado para fazer isso. Essa arquitetura é mostrada na Figura 2. O módulo captura os dados trafegando no canal por meio de interrupções periódicas, armazenando-os temporariamente em seus buffers internos. Posteriormente, um ”Processo Coletor”no espaço de usuário coleta tais dados, também de forma periódica, salvando-os então no meio de armazenamento. Por

estar dentro do kernel, o módulo pode obter dados com uma frequência superior ao do Processo Coletor e a um custo de processamento menor pois assim evitam-se o custos de trocas de contexto dos processos de usuários para o kernel e das chamadas do sistema do sistema operacional para leitura e escrita de dados.

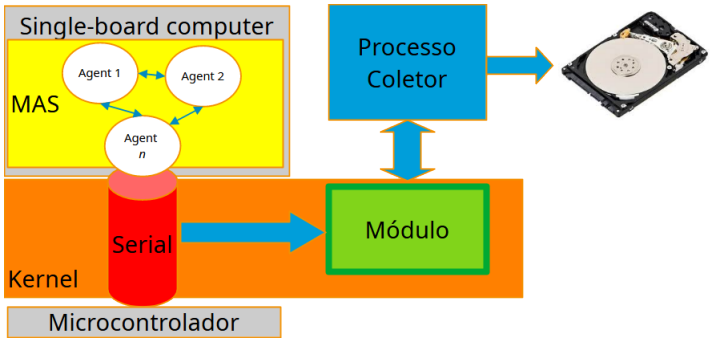


Figura 2. Módulo de proveniência proposto para SMAs embarcados

Com relação ao andamento atual deste trabalho, o módulo já consegue acessar e capturar dados do canal, e já existe uma interface kernel-espaço de usuário para que o Processo Coletor possa fazer a coleta dos dados. No estado atual, o módulo captura dados a uma frequência de 10 HZ, e o Processo Coletor os coleta com um período de 2s, mas isso é facilmente modificável. O principal desafio no momento é traduzir os dados coletados do canal nas ações Jason e dados dos sensores, devido ao fato deles estarem sendo capturados de forma bruta. Também, é ainda uma questão em aberto se a abordagem por amostragem pode resultar em perdas de dados significativas entre dois eventos de captura.

4. Discussão

A utilização de módulos de kernel é algo pouco explorado na literatura sobre SMAs. Tradicionalmente, o desenvolvimento de SMAs é feito com a ajuda de middlewares em alto nível. As vantagens de se utilizar tal abordagem é algo já bem estabelecido na literatura, e já existem diversos middlewares para o desenvolvimento de SMAs.

Este trabalho propõe fazer o movimento inverso, mostrando o andamento de diversos trabalhos de SMAs embarcados utilizando o sistema operacional como base. Foram apresentados trabalhos utilizando o sistema operacional como ponte entre um SMA e simulador e como provedor de proveniência para SMAs embarcados.

Referências

Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Cambridge: Cambridge, MA: Harvard University Press.

Dal Moro, D., Robol, M., Roveri, M., and Giorgini, P. (2022a). A demonstration of bdi-based robotic systems with ros2. In Dignum, F., Mathieu, P., Corchado, J. M., and De La Prieta, F., editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection*, pages 473–479, Cham. Springer International Publishing.

Dal Moro, D., Robol, M., Roveri, M., and Giorgini, P. (2022b). Developing bdi-based robotic systems with ros2. In Dignum, F., Mathieu, P., Corchado, J. M., and De La Prieta, F., editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection*, pages 100–111, Cham. Springer International Publishing.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21.

Gavigan, P. and Esfandiari, B. (2021). Agent in a box: A framework for autonomous mobile robots with beliefs, desires, and intentions. *Electronics*, 10(17).

Gazis, A. and Katsiri, E. (2022). Middleware 101. *Commun. ACM*, 65(9):38–42.

Hübner, J. F., Bordini, R. H., and Vieira, R. (2004). Introdução ao desenvolvimento de sistemas multiagentes com jason. *XII Escola de Informática da SBC*, 2:51–89.

Karaduman, B., Tezel, B. T., and Challenger, M. (2023). Rational software agents with the BDI reasoning model for Cyber–Physical Systems. *Engineering Applications of Artificial Intelligence*, 123:106478.

Michel, F., Ferber, J., and Drogoul, A. (2009). Multi-Agent Systems and Simulation: A Survey from the Agent Community's Perspective. In *Multi-Agent systems: Simulation and applications*. CRC Press. `https://doi.org/10.1201/9781420070248-10`.

Michel, O. (1998). Webots: Symbiosis between virtual and real mobile robots. In Heudin, J.-C., editor, *Virtual Worlds*, pages 254–263, Berlin, Heidelberg. Springer Berlin Heidelberg.

Open Robotics (2023a). Gazebo. `https://gazebosim.org/home/`.

Open Robotics (2023b). Robot operating system. `https://www.ros.org/`.

Palanca, J., Terrasa, A., Julian, V., and Carrascosa, C. (2020). Spade 3: Supporting the new generation of multi-agent systems. *IEEE Access*, 8:182537–182549.

Silvestre, I., de Lima, B., Dias, P. H., Buss Becker, L., Hübner, J. F., and de Brito, M. (2023). Uav swarm control and coordination using jason bdi agents on top of ros. In Mathieu, P., Dignum, F., Novais, P., and De la Prieta, F., editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*, pages 225–236, Cham. Springer Nature Switzerland.