Implementation of Communication between Autonomous Agents for Learning Collaborative Behaviors

Renato A. Nobre, Guilherme N. Ramos, Department of Computer Science, University of Brasília

***Abstract*—Systems that rely on multiple robots are capable of achieving numerous tasks in a more reliable and trustful way that systems with a single robot. The communication between these robots can multiply their capacities and effectively. This report discusses communication usage in the reduction of undesirable effects in stochastic multi-agent systems, with multiple robots learning in parallel while interacting with each other. Two huge issues, noise and locality, are addressed using direct communication methods in a "blackboard" approach. The implemented methodology is tested in a behavior selection simulator in multiples autonomous agents with reinforcement learning in a classical predator-prey environment. The communication is used to share, between the predators, the probability map of its prey position, with the attempt to reduce the knowledge's noise. Also, the learning values are shared between the agents, trying to diminish the locality issue. The obtained results demonstrate that the error from the noise decreases with the sharing of the probability maps. Also, the learning behavior's communication can increase aggressive politics, showing the impact in the selected behavior in the experiments, where the agents prefer to use the same policy, instead of complementing each other. However, the policies learned by the agent with communication does not generate collaborative behaviors in all of the experiments made.**

Keywords*—shared learning, probability maps, communication, predator-prey, multi-agent systems, artificial intelligence, Bayesian programming, Q-Learning, reinforcement learning.

I. Introduction

Uncertainties are inherent in real-world applications, where information cannot be measured directly, and measurements are often inaccurate[2]. For example, the robotics, a widely studied area in the field of distributed artificial intelligence [18], is inherently stochastic. In other words, robotics is full of uncertainties and random factors. Their mathematical models describe reality in a simplified way, where sensors are mostly imprecise and unpredictable, and the state's representation (knowledge about the environment) is limited to computational boundaries [12].

Although there is no consensus in the literature, it is necessary to select a definition of what is an agent, to limit the scope of the work. Thus, we will consider an agent as an entity that perceives the environment through sensors and acts in this environment, autonomously, through actuators [7]; and rational agents as those who seek actions that will maximize a certain measure of performance, in which case is the function of the state of the environment [13]. When multiple agents coexist and interact in a dynamic environment, that being cooperatively or competitively, the system is said to be multi-agent [14].

Cooperative agents combine their efforts to achieve a common goal, allowing the purposes of multiple agents to be performed simultaneously [13]. A scenario commonly applied in multi-agent systems is the predator-prey, where the prey must avoid being captured, while predators aim to capture it [1]. An example of this type of scenario is the case of multiple autonomous drones hunting a fugitive, where the fugitive is the prey in motion, its location is known, and when captured, it will imply a rewarding assignment for all drones. Despite not having a single format, the predator-prey problem traditionally has a discrete environment, a single prey and four predators [3]. As the predators' objectives are the same, this scenario is characterized as a cooperative multi-agent system [13]. In these environments, a behavior that is considered good at one moment may cease to be at another, so a rational agent needs mechanisms to adapt. One of these mechanisms is learning, in this work, understood as the agent's ability to improve his performance using experience acquired over time [7].

Reinforcement learning is one of the main approaches, in which the agent finds out what actions should be performed to maximize the numerical rewards received from the environment [13]. In this scenario, the agents do not have information on the best actions to be performed, discovering on their own when interacting with the environment [10].

In robotics, simulating a real environment is frequent, since computer simulations are usually faster and safer than physical tests [21]. Digital games provide an ideal simulation environment for the experimentation and study of artificial intelligence [16]. A classic game that studies the predator-prey problem in multi-agent systems is the *Pac-Man*, where the player, represented by the character *Pac-Man*, has to eat all the food available in the environment and avoid being captured by ghosts [1].

This work uses an environment platform for selecting behaviors in systems with the previously described characteristics; however, in previous works, the rational agents don't act in groups to capture the game in the experiments carried out on the platform. Therefore, it is interesting to analyze whether the application of communication methods in the simulator can generate cooperative policies in all simulations. It is possible that, when inserting forms of communication between agents, they explore cooperation more frequently and achieve better results.

Two methods of communication are aimed to be imple-

mented in this work. The first method focuses on reducing the precision error of the sensors, the system noise. The noise problem increases the system's uncertainties and comes from the lack of reliable sensors, one of the most significant learning limitations in robotic systems [15]. The second method aims to solve the locality of the distributed system, that is, the fact that the agents decide to accomplish the objective by themselves, not working as a group. This issue is a restriction to achieve group-level cooperation between agents [8].

This report is organized as follows. Section II develops important theoretical concepts for understanding the selected platform, and this work as a whole. Section III introduces the concept of multi-agent systems and the platform used. Section IV develops the concepts and methods of communication implemented. Section V presents the hypotheses of the experimental script and its results. Lastly, Section VI finalizes the report, summarizing the results obtained and suggesting future work.

II. Reinforcement Learning and Bayesian Programming

In stochastic environments, random environments where uncertainties are present, the agent has no way to rely on previous examples and starts the simulation without any information about the inserted place, and no reward function [10]. However, if the agent tries random movements, it can eventually build a predictive model of the environment. It will receive a reward r, and learn from previous experiences which actions can improve the future reward [7].

In artificial intelligence, reinforcement learning represents a category of computational techniques for agents to improve their performance over time. From their iterations with the environment, they are inserted [10]. So the task of reinforcement learning is to use such rewards r to learn an optimal behavioral function (a function that can achieve the goal with the greatest reward in the shortest period). The agent is assigned an action-value function, based on the expected reward of taking action in a given state of the environment. This is denominated *Q-learning* [7]

A. Q-learning

The Q-learning algorithm was developed as an agent control algorithm with the simplicity and possibility of transitioning a function V that estimates the value of a current state s_t to an optimized control policy, which calculates action-state values $Q(s_t, a_t)$ [10]. Thus, an agent does not store the information of how much reward can be received from a state, but how much can be obtained if, in a given state s_t, the agent performs the action a_t [1].

The Q-learning algorithm uses a discount factor γ, limited to the interval $0 \leq \gamma \leq 1$, which allows providing greater relevance for immediate or future rewards [2]. When γ approaches 0 the agent seeks to maximize short-term rewards, on the other hand, when approaching 1 long-term rewards will be maximized [1]. Another factor used by Q-learning is the learning rate α contained in $0 < \alpha < 1$ which has the role of limiting interaction's learning, that is, the maximum variation of the value $Q(s_{\text{t-1}}, a_{\text{t-1}})$ [2].

The Eq. (1) presents the Q-learning algortihm with a single step:

$$Q(s_{\text{t-1}}, a_{\text{t-1}}) = Q(s_{\text{t-1}}, a_{\text{t-1}}) + \alpha\varphi \tag{1}$$

Where φ is defined as:

$$\varphi = [r_{\text{t}} + \gamma \max{}_{\text{a}} Q(s_{\text{t}}, a) - Q(s_{\text{t-1}}, a_{\text{t-1}})] \tag{2}$$

As an example, consider a predator-prey situation presented in the *Pac-Man* game. If the *Pac-Man* is in a state s that is a position away from a ghost, and performs an action a' that causes him to be captured, the value of $Q(s, a')$ will be updated with a negative weight. Therefore, when *Pac-Man* sees himself in that situation again, considering that there are better actions, Q-learning will choose the action with the highest Q value, thus avoiding the action that leads to him being captured.

However, Eq. (1) and Eq. (2) are computationally infeasible in continuous environments, due to the infinity of state-action pairs [2]. An alternative is to apply function approximation methods to estimate $Q(s_t, a_t)$.

B. Q-learning with Function Approximation

It may become unfeasible in stochastic systems to execute the algorithm represented in the Eq. (1). This unfeasibility is mainly due to time and hardware limitations, e. g., memory size and processing power [1]. This issue might be fixed using function approximation to estimate the $Q(s_t, a_t)$ value. Doing so, instead of keeping all the possible parameters for each state-action pair, only a vector of parameters θ is kept and used to calculate the estimation of the $Q(s_t, a_t)$ value [1]. The learning is then performed by updating the components θ_i in the vector with the agent's rewards.

Among the function approximation methods, a common choice is the linear approximation [10], where a vector of characteristics $\phi(s)$, of the same size as θ, is defined from the current state of the environment. Thus, the parameter θ_i defines the weight that each characteristic $\phi(s)$ $0 \leq \phi_i(s) \leq 1$, will have in the calculation of the approximation of $Q(s, a)$ [1].

For example, in the *Pac-Man* scenario, the features vector can be composed of two distinct features, the first assuming the value 0 or 1, indicating whether the ghost is a cell away from the *Pac-Man* and the second represents the probability of the ghost not capturing *Pac-Man*. A vector of parameters θ with a negative value in the first parameter $\phi_1(s)$, and positive in the second, $\phi_2(s)$, indicates that $\phi_1(s)$ negatively affects the value $Q(s_t, a_t)$ of the action-state pair, and $\phi_2(s)$ affects positively.

Therefore, from the vectors θ and ϕ, the estimated value of the action-state pair is calculated. In the linear approximation approach, learning occurs by updating the vector θ according to the reward received [2].

C. Bayesian Programming

Reinforcement learning algorithms select the most appropriate behavior for the current state of the system according to its objective. However, such algorithms require the state to be a known variable, which does not happen in stochastic environments [2]. Decision-making processes that consider uncertainties in the environment have been increasingly studied by artificial intelligence researchers [12].

Bayesian programming is a *framework* for developing intelligent systems, assisting in decision-making processes that consider the uncertainties of a stochastic environment. Its mathematical formalism is generic enough to reinterpret classic probabilistic models [19]. Its fundamental element is the logical proposition, a hypothesis h that can be either true or false, where all propositions are dependent on prior knowledge of the system, represented by π. So logical propositions are always conditioned on π, that is, $P(h|\pi)$ [20]. Another fundamental concept in Bayesian Programming is a discrete variable: a set X of exhaustive and mutually exclusive propositions.

Any Bayesian program contains two parts: a *description*, a set of probabilistic distributions of all relevant variables, using previous knowledge π and experimental data δ; and a *question*, which is the probabilistic distribution calculated using the *description* [20].

Therefore, the *description* is divided into three sets of variables: the variables S on which the probabilities are calculated, the variables K that can be observed, and the unknown variables U [19]. For example, consider as the *description* the probability sets of, "*Pac-Man* being close to the ghost", "*Pac-Man* next to the pill" and "ghost next to the pill". The *question* is then defined by the probability of the *Pac-Man* being close to the ghost, given the probability of the *Pac-Man* and the ghost being close to the pill. Later, this estimation can be provided in other parts of the intelligent system, such as in the action's selection [2].

III. Reinforcement Learning Simulation Platform for Multi-agent Systems

The implementation of communication between autonomous agents is based on a system developed in the work of *Selection of behaviors in multiple autonomous agents with reinforcement learning in stochastic environments* [1].

The work proposes an algorithm for stochastic multi-agent systems, using Bayesian programming for state estimation, and Q-learning with function approximation to provide agents with the ability to learn and select more appropriate behaviors. For the work mentioned above, the behavior was defined as the set of actions performed by an agent according to the environment's state. Such a definition will also be used for this work.

A. Multi-agents System

The sub-field of artificial intelligence that provides principles and mechanisms for building complex systems and coordinating agents' behavior is that of Multi-agent Systems [3]. Although there is no general concept defined for an agent in artificial intelligence [7], for this research, we will consider an agent as an entity inserted in a particular environment, with well-defined objectives and actions [4].

B. The Location and Noise Problems

A key challenge for distributed multi-agent systems is to achieve group-level cooperation between agents [8]. Fully distributed systems in which each agent learns independently introduce difficulties such as non-stationary environments. As the agents learn, their behavior changes, resulting in inconsistencies; and credit assignment problems, which cause agents to achieve goals on their own. In this problem, as each agent receives their rewards individually, if one of the agents learns to accomplish the goal independently and achieves satisfactory results without the need of the others in the group, he may always tend to achieve the objectives, and the others will not collaborate [14].

Such behavior is called *locality* of the distributed system, usually this behavior is not desired, considering that in problems of multiple agents with the same objective, better results can be obtained with the cooperation [8].

Another widespread problem in stochastic environments and robotic systems is the presence of noise in which it interferes with sensor measurements in numerous ways. Consequently, it limits the information that can be extracted by measurements [12]. The lack of accurate and reliable sensors is arguably the biggest problem for agent control and learning researchers, being one of the most significant limiting factors for real-time learning [15].

C. Pac-Man Simulator

A simulator of the *Pac-Man* game was used for this work. This simulator was developed for artificial intelligence subjects by professors John DeNero and Dan Klein of the University of California, Berkeley, called *The Pac-Man Projects* [1].

An essential distinction of the simulator is the existence of a scoring system that can be used to attribute the agents' rewards and, thus, used in reinforcement learning [1]. Such a scoring system present in the simulator is unique, this means that the lower the system's score, the better the ghosts' performance, and the higher the score, the better the performance of *Pac-Man*.

D. System Specifications

The system structure was developed to avoid coupling between the learning algorithm and the simulator, using two modules, the controller and the adapter [1]. Thus, it is possible to perform tests on robotic systems or other simulators with little code change.

The controller receives measurement and reward information provided by the adapter, passes that data on to the learning module, executes the behaviors and sends the actions back. The adapter system is responsible for reconciling the environment to respect the interfaces provided by the controller [1].

[1] Available at http://ai.berkeley.edu/project_overview.html

The communication system used to transfer messages between the communication module and adapter was implemented using the library *ZeroMQ3* [2]. Its choice was made due to the ease of implementing several communication architectures with few modifications and the abstraction of the sending and receiving *TCP/IP* protocols, allowing the modules to be executed in simple ways on different machines [1].

Finally, the system used presents problems typical of distributed multi-agent systems, such as those described in Section III-B. To deal with these behaviors, communication methods have been implemented.

IV. Communication Methods' Implementation

Communication between agents can multiply their capacities and effectiveness [4]. Any observed behavior, as well as its consequences, can be interpreted as a form of communication.

The system developed in [1], when used with rational ghosts, for this research defined as ghosts that have gone through a learning process and a *Pac-Man* that has not performed such a process, can be classified as a *homogeneous multi-agent system without communication* [3]. In such scenarios, all agents have the same internal structure, including the objective. However, when applying communication, we can transform it into a *homogeneous multi-agent system with communication*, and possibly, agents will be able to coordinate their objectives more effectively than before.

To develop the concept of the implementation of communication, it considered its forms and definitions. The *indirect communication* is based on the observation of the effect of other agents' behavior in the environment. In other words, it is the noticeable communication of the modification of an environment [8]. In contrast, we have *direct communication*, an act of pure communication, to transmit information to a recipient agent. For this research, direct communication was implemented to reduce noise and location problems. The choice of direct communication was favored because indirect communication methods are not efficient in situations in which communication does not generate changes in the environment. To implement them, agents would need a way to change the environment [4] .

The transmission of messages between agents was implemented in a "blackboard" approach. All agents send their information to a module that will process the data and redistribute it back to the agents [13]. This approach was favored over the "pear to pear" transmission of messages from one agent to another specific agent [3]. The reason for such is due to the system already having a centralized controller module that works ideally for the approach.

To carry out the message transmission between the controller module and the agents, two types of message protocols were defined in the communication module: *Request* and *Assignment*. The protocols of the *Request* kind, ask the agent for the information they need and respond when the information is received. On the other hand, the *Attribution* protocols properly handle data from *Request* methods and distribute the treated data to agents.

[2]Available at http://zeromq.org

Two types of direct communication have been implemented: communication of the estimated position of *Pac-Man* and communication of the states, behavior and reward parameters between ghosts.

A. Probability Map Communication Implementation

In multi-agent systems, the ability to correctly detect and distinguish agents between groups, obstacles and other diverse characteristics of the environment is crucial for most of the tasks and objectives [8].

On the platform used, the agents' ability to detect their positions and detect each other in the environment is performed with probability maps. A probability map is a probabilistic model of the environment in which each position in a matrix $M_{i,j}$ will have a value x contained in $0 < x < 1$, and the sum of all its positions is always equal to 1, represented by the Eq. (3). Therefore, the highest x value of $m_{k,l}$ positions is where the agent is most likely to be located.

$$\sum_{k=1}^{i}\sum_{l=1}^{j} m_{k,l} = 1 \tag{3}$$

Each agent in the simulation has a probability maps for all of the simulation's entities. An agent has a unique probability map for its position and for the place of the other $n-1$ agents, where n is the number of agents inserted in the environment. For example, suppose that a simulation is performed with three ghosts and a *Pac-Man*, each ghost will have a map of probabilities of its position, one for each of its allies and one for the position of *Pac- Man*, that is, each ghost will have four distinct probability maps, one for each entity.

The presence of noise from sensors is one of the biggest challenges for learning in the system, making it challenging to detect agents. To simulate this problem from the sensors, the simulator provides a noise implementation system in the probability maps, which may increase the agents' estimation error.

Therefore, the probability map's communication was implemented in the ghosts to decrease the estimation error from noise in the detection of the *Pac-Man*. That is, each ghost communicates only the probability map of the *Pac-Man* position. This communication occurs in a way that the controller module calls the *Request* protocol, requiring each agent to communicate its estimate of the position of the *Pac-Man*. When all ghosts have communicated their assessment, the controller will merge the maps.

The union of the maps is done so that each position $m_{i,j}$ of an agent's map is multiplied by the same position $m_{i,j}$ of the map of the other agents. Thus a new map M' will be generated by multiplying each position $m_{i,j}$ of each map of the agents. However, when multiplying probabilities, we need to normalize them, that is, we need to divide each value of the map by the sum of all values, so that the sum of all positions $m_{i,j}$ is always 1.

The multiplication of two tiny probabilities can be rounded to zero in computational environments due to the approximation error. To avoid this unwanted behavior, multiplication

has been replaced by an equivalent method, using the sum of natural logarithms base e. At the end of the calculations, the M' map is distributed to the agents using the *Attribution* protocol.

To test whether the approach of the union of probability maps was capable of reducing the error generated by noise, methods were implemented to calculate the mean square error of the simulations.

The calculation of the errors ξ for each simulation when there is a communication of the probability map was performed as follows. In each instance of the simulation, the squared difference between the higher probability of the map and the real position of the *Pac-Man* was calculated. Therefore, at the end of the game, the sum of errors for each instance t is divided by the number of instances described in Eq. (4), was obtained. This procedure is repeated for k games, and at the end of the simulation, the error is equivalent to the sum of the games' errors xi_j divided by the quantity k of games, thus represented by the Eq. (5).

$$\xi_j = \frac{\sum_{t=1}^{n} \xi_t}{n} \tag{4}$$

$$\xi = \frac{\sum_{j=1}^{k} \xi_j}{k} \tag{5}$$

Similarly, in cases where there is no communication, the game error is calculated as the sum of the most significant mistakes between agents. The error of each agent is calculated by the difference in estimate in a similar way to communication. At the end of the simulation, the error is equivalent to the sum of the games error divided by the number of games.

B. Learning Communication Implementation

The communication of learning aims to reduce locality and resolve the credits attribution issue so that agents perform actions as a group based on cooperative policies.

We will consider cooperative policies as complementary policies to achieve the objective. The *framework* of *Pac-Man*, defines three types of behavior in which the ghost can perform, *Ambush*, in which the ghost moves to reach the future position of the *Pac Man*; *Chase*, of predatory character, the agent moves in a straight line with the prey; and finally the *Flee* where the agent tries to get away from the predator. Therefore, we will define cooperative policies, policies in which agents choose complementary behaviors as a priority, that is, *Chase* and *Ambush*.

To address credits' attribution and locality, communication protocols have been developed to transfer the learning information. The controller module, when executing a *Request* protocol, requires each ghost agent to report its state s_t, the previous state s_{t-1}, the executed behavior c, and the reward r received for behavior. When the controller receives the learning information from all ghosts f, it performs the learning update by calling the protocol *Assignment*, so each ghost learns n times per iteration.

The learning update occurs when the *Assignment* protocol receives the variables stored by the *Request* protocol and calls a method from the learning module. This method takes such variables as a parameter and updates the weight φ, represented by Eq. (2). The procedure is called $f-1$ times for each ghost, each time revising the learning with another allied agent's parameters.

V. Performance Evaluation

Based on the problems mentioned in Section III-B, and with the methods implemented in the system described in Section IV, the following questions were elaborated:

1) Is the probability maps communication able to reduce the error from the noise?
2) Is the probability maps communication able to reduce the number of instances in noisy simulations?
3) Can the shared reward decrease the system locality?
4) Is it possible with the learning communication that the ghosts' final score increases significantly against an efficient *Pac-man*?
5) When both types of communication are used (probability map and learning), is it possible for agents to generate cooperative policies in all executions when pitted against a random *Pac-Man* or an efficient *Pac-Man*?

To answer the questions presented, experiments were developed. We will consider an experiment as the set of 30 simulations performed with the same simulation parameters. The value of 30 was defined by Eq. (6), which estimates a sample size given that we are not aware of the variance or population proportion, with $\delta = 0.95$ and $\epsilon \approx 0.179$. Where δ is the value of the confidence interval, z_δ is the value of the normal distribution for a confidence interval δ, and ϵ is the maximum sample error [11] .

$$n \approx \frac{z_\delta^2}{4\epsilon^2} \tag{6}$$

Regardless of the experiment performed, all simulations were performed with 100 learning games, 15 test games, and when there is a n noise, it is contained between the $-3 < n < 3$ range, as well as defined in [1]. The simulations were done with a reduced simulation environment, compared to the classic map of *Pac-Man*. The choice of the reduced map is due to the time reduction for the experiments' execution.

The *Pac-Man* agents used for the experiments were developed and described in [17]. The random *Pac-Man* used, called in the simulator as *RandomPacmanAgentTwo*, is a pseudo-random agent that chooses an action and performs it until the action is no longer available when it finds a wall blocking it, or have more than three possible actions, in which case, there is a probability to select one of these actions. The random *Pac-Man* was avoided in the described simulations since the agent frequently stays in the same region or map position. Another *Pac-Man* used is the *BFSPacmanAgent*, which performs a search procedure by width, on the map foods. And finally, the *Pac-Man* used considered efficient, is called *NimblePacmanAgent*, which feeds when it is safe and runs away when it is in danger, analyzing its position with that of

ghosts. Its efficiency was analyzed compared to the other *Pac-Man* agents available in the simulator, presenting satisfactorily better results regarding its score obtained in the game.

A. Probability Map Communication

To answer Question 1, two experiments were conducted. Both experiments with four ghost agents, for being the classic quantity in a predator-prey problem [3], and with the presence of noise, the only difference between the two is the presence of communication of the maps.

In the experiments that communicated the probability map, the average quadratic error was $5,772$, and in cases without the communication, this error increases to $7,183$ reducing the noise error by approximately 19.64%.

Answering Question 2, experiments were carried out with four ghost agents. Two experiments were performed for the *BFSPacmanAgent*, and the other two for the *NimblePacmanAgent*, for both, the only difference between the two experiments was the presence, or not, of the probability map communication.

Even with the mean square error decreasing, the games' number of instances does not appear to change effectively. The average number of instances of experiments performed with *BFSPacmanAgent*, with the presence of communication, is approximately $37,311$, still very close to when there is no communication, which is on average $38,318$. In the case of *NimblePacmanAgent*, the number of instances is slightly greater when there is communication compared to when there is not, standing at 43.578 and 43.225 respectively.

B. Learning Communication

With the communication of the shared reward, the aim is to solve credit allocation and reduce the locality of the system. To answer Question 3, experiments were performed with four ghosts agents and without noise, again using only *BFSPacmanAgent* and *NimblePacmanAgent*. For each *Pac-Man* agent, two experiments were carried out, one with learning communication, and the other not.

In these experiments, for each simulation, the probability of each agent selecting a specific behavior was analyzed. If the agent's likelihood to choose a behavior was greater than 50%, concerning the others analyzed, the behavior was considered the agent's primary in the simulation. The value was chosen to avoid uncertainties in the definition of the agent's behavior since there are three behaviors; the majority would be greater than 33%. However, it is analyzed when the probability of a behavior is necessarily greater than the sum of the other two. This way, there is greater precision in the choice of the agent's behavior. Therefore, if none of the agent's behaviors had a probability greater than 50%, his final behavior was indeterminate. Then, the average of the behaviors selected by the agents in the experiment was performed. Table I summarizes the relationship between the experiments and the average of each behavior.

Table I. LEARNING COMMUNICATION EXPERIMENTS RESULTS

	BFSPacmanAgent		*NimblePacmanAgent*	
Communication	Absent	Present	Absent	Present
Chase	1,400	1,233	0,967	1,500
Ambush	1,267	1,266	1,267	1,400
Flee	1,233	1,366	1,433	0,900
Indeterminate	0,100	0,133	0,300	0,200

A more detailed analysis shows that for the cases of experiments with the *NimblePacmanAgent*, the average of an agent selecting an escape behavior decreases from approximately $1,433$ to $0,900$. While search and chase behaviors increase from $1,267$ to $1,400$ and $0,967$ to $1,500$ respectively. To sum up, escape behavior decreases by 37.19%, search behavior increases by 10.50% and chase behavior by 55.11%. Another analysis carried out under the experiment was regarding the number of simulations in which the agents chose cooperative policies. It was noted that in the experiment without the presence of communication the agents chose 11 times cooperative policies, and 3 times equal policies, in which all agents chose the same policy at the end of the simulation, in one of those 3 times all policies were *Flee*. On the other hand, the agents chose fewer complementary policies with the communication, totaling 9 times, but the number of equal policies increases to 14 times, being 3 times with the escape policy. Those results indicate that agents choose the same policies more often when there is the presence of communication.

For experiments with the *BFSPacmanAgent*, the average of an agent selecting an escape behavior with the communication, increases from $1,233$ to $1,367$, the chase behavior selection decreases from $1,400$ to $1,233$ and the ambush remains at $1,670$ for both cases. Again, the policy choice analysis was performed. In this case, when there is no communication, the agents never selected the same policies and selected 20 times complementary policies. On the other hand, when there is communication, the number of complementary policies decreases slightly, being chosen 16 times, while simulations with equal policies occur 11 times, being 6 with the Flee policy. The same behavior was observed in *NimblePacmanAgent* in which equal policies are chosen primarily over complementary policies, which may indicate that the selection of equal aggressive policies has better results.

Answering Question 4, the final score of the games was analyzed in the presence of learning communication. The game score with *BFSPacmanAgent*, decreases from $59,346$ when there is communication to $45,358$ when there is not. This means that the presence of communication decreased the ghost's score. And the same type of behavior occurs for the *NimblePacmanAgent*, which, with the shared reward, has a score of 463.207, which decreases to 444.806 in the experiment without communication.

C. Joint Communication

Finally, joint communication was analyzed to answer Question 5. Eight experiments were carried out with the presence of noise, four using two ghosts and four using three. This reduced number of ghosts was defined since only two cooperative policies exist, so for two ghosts, the ideal would be for each ghost to choose one policy complementary to the other.

The first four experiments were carried out with two ghosts and the *RandomPacmanAgentTwo* and *NimblePacmanAgent* agents. Again, for comparison, two of the experiments were carried out without any communication. Table II summarizes the results of these experiments.

For the experiments with the random *Pac-Man*, the presence of the escape policy was still observed. The policy selection average remains almost the same when communication is applied; the average decreases from 0.667 to 0.600. The search policy selection also decreases, and the pursuit policy increases. For the efficient *Pac-Man*, the choice of the escape policy is also observed, and remains at 0.700 regardless of the presence of communication, as the ambush and chase policies work in the opposite way to the previous case, ambush increases and chase decreases.

Table II. RESULTADO DOS EXPERIMENTOS COM DOIS AGENTES FANTASMAS

	RandomPacmanAgentTwo		*NimblePacmanAgent*	
Communication	Absent	Present	Absent	Present
Chase	0,533	0,733	0,667	0,533
Ambush	0,800	0,667	0,667	0,767
Flee	0,667	0,600	0,700	0,700
Indeterminate	0,000	0,000	0,033	0,000

Os outros quatro experimentos seguem o mesmo padrão dos experimentos anteriores, diferenciando apenas na quantidade de fantasmas, agora com três agentes. A Tabela III mostra o gráfico destes experimentos.

The other four experiments follow the same pattern as the previous experiments, differing only in the number of ghosts, now with three agents. Table III shows the graph of these experiments.

Table III. RESULT OF EXPERIMENTS WITH THREE GHOSTS AGENTS

	RandomPacmanAgentTwo		*NimblePacmanAgent*	
Communication	Absent	Present	Absent	Present
Chase	0,833	0,933	1,200	1,100
Ambush	1,267	1,067	0,733	1,167
Flee	0,900	1,000	1,067	0,633
Indeterminate	0,000	0,000	0,000	0,100

Again there is still the presence of the Flee behavior for both *Pac-Man* agents. The amount increases when there is communication for the random agent, from 0.900 to $1,000$. However, for the efficient agent the Flee decreases from $1,067$ to $0,633$. For the Chase and ambush, the results are similar to the experiments carried out with just two ghosts, with the efficient *Pac-Man*, the search increases again, and the Chase decreases, with the random *Pac-Man*, the search decreases and the ambush increases.

VI. CONCLUSION

Multi-agent distributed systems can multiply efficiency compared to a single agent system. However, a key challenge for this type of system is to achieve group-level cooperation between agents. The implementation of communication methods may be able to address such a challenge.

This work implements two types of direct communication with a "blackboard" approach, in multiple autonomous agents. The primary purpose of the communication is to analyze its impact on the learning of collaborative behaviors, addressing two problems of the distributed multi-agent systems, the location and the noise. The locality problem was addressed using the communication of the agents' learning parameters. Meanwhile, the noise issue was approached with the communication of the *Pac-Man*'s position estimate generated by the ghosts, via probability maps, was implemented.

The implementations were tested with several experiments, each with thirty simulations. Their results were analyzed:

- The error from the noise decreases by 19.64% with the communication of the probability maps; however, the decrease in noise does not significantly change the simulations' number of instances.
- The learning communication, when the opponent is efficient, increases the occurrences of the *Chase* and *Ambush* policies, decreasing occurrences of *Fleeing*, however, such behavior does not occur when the opponent is not efficient. Such communication also shows an impact on the selection of behaviors in the experiments, where agents prefer to use equal policies instead of complementary policies.
- The phantom score does not increase with learning communication.
- The simultaneous communication of both methods did not generate collaborative behavior in all experiments.

The policies learned by the agents with communication did not generate cooperative behavior in all the experiments carried out. Therefore, it is interesting to analyze whether other types of communication are capable of obtaining better results. It may be possible that when making the probability maps union in relation to the position of the ghosts, the agents will have better knowledge regarding their position and that of their allies, being able to improve their score. Another learning method that can also be implemented is sharing the entire policy, at the end of the game, of the agent who captured the prey. Finally, future work can analyze the communication performance in a convolution scenario, where the prey and the predators would evolve simultaneously.

ACKNOWLEDGEMENT

The author would like to thank Professor Guilherme Novaes Ramos for the vote of confidence in providing me with the opportunity to conduct this research and his excellent guidance and academic support. I also would like to thank my family for their support in difficult times and my friends Khalil Carsten and Marcelo Araujo to find time for several readings and provide suggestions for my work. Finally, I thank Matheus Vieira Portela for developing the excellent system used in the project.

REFERENCES

[1] Portela, Matheus V. "Seleção de comportamentos em múltiplos agentes autônomos com aprendizagem por reforço em ambientes estocásticos." (2015). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

[2] Portela, Matheus V. and Ramos, Guilherme N. “State estimation and reinforcement learning for behavior selection in stochastic multiagent systems.” SBC – Proceedings of SBGames 2015 | ISSN: 2179-2259

[3] Stone, Peter and Veloso, Manuela. “Multiagent systems: A survey from a machine learning perspective.” Autonomous Robots 8.3 (2000): 345-383.

[4] Balch, Tucker and Arkin, Ronald C. “Communication in reactive multiagent robotic systems.” Autonomous robots 1.1 (1994): 27-52.

[5] Stone, Peter and Veloso, Manuela. “Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork.” Artificial Intelligence 110.2 (1999): 241-273.

[6] Lebeltel, Olivier and Bessière, Pierre and Diard, Julien and Mazer, Emmanuel. “Bayesian robot programming” Autonomous Robots 16.1 (2004): 49-79.

[7] Russell, Stuart and Norvig, Peter “Artificial Intelligence: A modern approach.” Artificial Intelligence. Prentice-Hall, Englewood Cliffs 25 (1995): 27.

[8] Matarić, Maja J. “Using communication to reduce locality in distributed multiagent learning.” Journal of experimental & theoretical artificial intelligence 10.3 (1998): 357-369.

[9] Kaiser, M. and Dillman, R. and Rogalla, O. “Communication as the basis for learning in multi-agent systems.” ECAI’96 Workshop on Learning in Distributed AI Systems. 1996.

[10] Sutton, Richard S. and Andrew G. Barto. “Reinforcement learning: An introduction.” Vol. 1. No. 1. Cambridge: MIT press, 1998.

[11] Bussab, Wilton de O. and Pedro A. Morettin. “Estatística básica.” Saraiva, 2010.

[12] Thrun, Sebastian and Burgard, Wolfram and Fox, Dieter. “Probabilistic Robotics.” [S.l.]: MIT press, 2005.

[13] Weiss, Gerhard. “Multiagent systems: a modern approach to distributed artificial intelligence.” MIT press, 1999.

[14] Panait, Liviu, and Luke, Sean. “Cooperative multi-agent learning: The state of the art.” Autonomous agents and multi-agent systems 11.3 (2005): 387-434.

[15] Matarić, Maja J. “Reinforcement learning in the multi-robot domain.” Autonomous Robots 4.1 (1997): 73-83.

[16] Lucas, Simon M. “Computational intelligence and games: Challenges and opportunities.” International Journal of Automation and Computing 5.1 (2008): 45-57.

[17] Cesarino, Pedro S. “Implementação de Agente Racional Autônomo para Aprendizagem de Comportamentos Colaborativos.”
Available: https://github.com/PedroSaman/Multiagent-RL

[18] Arai, Tamio, and Pagello, Enrico, and Parker, Lynne E. “Editorial: Advances in multi-robot systems.” IEEE Transactions on robotics and automation 18.5 (2002): 655-661.

[19] Koike, Carla M. C. E. C. “Bayesian Approach to Action Selection and Attention Focusing. An Application in Autonomous Robot Programming.” Diss. Institut National Polytechnique de Grenoble-INPG, 2005.

[20] Lebeltel, Olivier, and Bessière, Pierre, and Diard, Julien, and Mazer, Emmanuel. “Bayesian robot programming.” Autonomous Robots 16.1 (2004): 49-79.

[21] Tikhanoff, Vadim, and Cangelosi, Angelo, and Fitzpatrick, Paul, and Metta, Giorgio, and Natale, Lorenzo, and Nori, Francesco. “An open-source simulator for cognitive robotics research: the prototype of the iCub humanoid robot simulator.” Proceedings of the 8th workshop on performance metrics for intelligent systems. ACM, 2008.