Ministry of Higher Education and Scientific Research

Abbes Laghrour University of Khenchela

Faculty of Science and Technology

Mathematics and Computer Science Department

Thesis

Submitted to obtain a Master's degree in Computer Science (L.M.D)

By:

BADIS Sami

TABTI Mouhamed Amine

Domain: Web Security and Technology (WST)

The Proposition of a New Multi-Agent Access Mechanism Based on the Fairness Principle

Examined on June 27th, 2022 by the jury:

- ***Dr. LEDMI Abdeldjalil (President)***
- ***Dr. SOUIDI Mohammed El Habib (Supervisor)***
- ***Dr. MEHALAINE Ridha (Examiner)***

University year: *2021/2022*

Abstract

A multi-agent system can be defined as a set of entities able to interact with each other and the environment to execute complex tasks. Usually, this kind of system is composed of reactive agents playing different roles. The intelligence of the system emerges from the behaviors undertaken by the agents. These behaviors can be categorized into two types, collaborative and competitive behaviors.

Multi-agent Systems (MAS) often must pursue a common goal to realize that they have to form an efficient coalition. The Pursuit-Evasion game (PE) is a MAS np-hard problem in which the pursuers' agents must block the evaders' movement while considering the temporal constraint. On the one hand, this problem can be processed via the introduction of a coalition formation algorithm allowing the pursuers to be grouped in different pursuit groups according to the detected evaders in the environment. On the other hand, it can be treated via path planning approaches that permit the displacements of the pursuers.

An approach for coalition formation of multi-agent fairness-based and Agent Group Role Membership Function-based model is proposed, thus making a unique algorithm Fairness-based AGRMF known as FBAGRMF. To achieve this, we have completely changed the organization method and created a group access mechanism that creates balanced pursuit groups and compared it to the existing approach of coalition formation in an existing PE. The simulation results reflect this new algorithm's advantages and collaboration between agents in MAS.

Keywords: Multi-agent systems; Organization; Coalition formation; Pursuit-Evasion; Markov Decision Process.

Résumé

Un système multi-agents peut être défini comme un ensemble d'entités capables d'interagir entre elles et avec l'environnement pour exécuter des tâches complexes. Habituellement, ce type de système est composé d'agents réactifs jouant des rôles différents. L'intelligence du système émerge des comportements adoptés par les agents. Ces comportements peuvent être classés en deux types, les comportements collaboratifs et compétitifs.

Les systèmes multi-agents (SMA) doivent souvent poursuivre un objectif commun pour se rendre compte qu'ils doivent former une coalition efficace. Le jeu Poursuite-Evasion (PE) est un problème SMA np-difficile dans lequel les agents des poursuivants doivent bloquer le mouvement des évadés en tenant compte de la contrainte temporelle. D'une part, ce problème peut être traité via l'introduction d'un algorithme de formation de coalition permettant de regrouper les poursuivants dans différents groupes de poursuite en fonction des évadés détectés dans l'environnement. En revanche, elle peut être traitée par des approches de planification de trajectoire permettant les déplacements des poursuivants.

Une approche pour la formation de coalition d'un modèle basé sur l'équitabilité multi-agents et sur la fonction d'appartenance au groupe d'agents est proposée, créant ainsi un algorithme unique AGRMF basé sur l'équitabilté connu sous le nom de FBAGRMF. Pour y parvenir, nous avons complètement changé la méthode d'organisation et créé un mécanisme d'accès de groupe qui crée des groupes de poursuite équilibrés et l'avons comparé à l'approche existante de formation de coalition dans un PE existant. Les résultats de la simulation reflètent les avantages de ce nouvel algorithme et la collaboration entre les agents dans MAS.

Mots clés : Systèmes multi-agents ; Organisation ; Formation de coalitions ; Poursuite-évasion ; Processus de décision de Markov.

ملخص

يمكن تعريف الأنظمة متعددة العملاء على أنها مجموعة من الكيانات القادرة على التفاعل مع بعضـــها البعض ومع البيئة لتنفيذ المهام المعقدة. عادة، يتكون هذا النوع من الأنظمة من عوامل تفاعلية تلعب أدوارًا مختلفة. ينبثق ذكاء النظام من الســـلوكيات التي يقوم بها العملاء. يمكن تصنيف هذه السلوكيات إلى نوعين، سلوكيات تعاونية وتنافسية.

الأنظمة متعددة العملاء غالبًا إلى تحقيق هدف مشـــترك لتدرك أنه يتعين عليها تشـكيل تحالف فعال. لعبة المطاردة والهروب هي مشـكلة ذات صـعوبة np، حيث يجب على العملاء المطـاردين منع حركـة المتهربين بـأخذ القيد الزمني في عين الاعتبـار. من ناحية أخرى، يمكن معالجة هذه المشـــكلة عن طريق إدخال خوارزمية تشـــكيل تحالفات تســـمح للمطاردين بالتجميع في مجموعات متابعة مختلفة وفقًا للهاربين المكتشـــفين في البيئة. من ناحية أخرى، يمكن معالجتها من خلال مناهج تخطيط المســـار التي تســـمح بتحرك المطاردين.

تم اقتراح طريقة لتشـكيل التحالف متعدد العملاء بنظام قائم على العدل لنموج مبني على خوارزمية العميل المجموعة الدور وظيفة الانتماء (AGRMF)، وإنتاج خوارزمية فريدة قائمة على العدل والمعروفة باســم FBAGRMF. لتحقيق ذلك، قمنا بتغيير طريقة التنظيم تمامًا وأنشأنا آلية دخول إلى المجموعة تنشئ مجموعات متابعة متوازنة ومقارنتها بالنهج الحالي لتشـــكيل التحالف في لعبة المطاردة والهروب الحالي. تعكس نتائج المحاكاة مزايا هذه الخوارزمية الجديدة والتعاون بين الوكلاء في الأنظمة متعددة العملاء.

الكلمات المفتاحية: أنظمة متعددة العملاء؛ التنظيم؛ تشـكيل التحالفات؛ الطاردة والهروب؛ عملية اتخاذ القرار ماركوف.

Acknowledgment

The realization of this thesis was made with the help of several individuals that we will not be able to never express our gratitude to them.

We would like to especially thank our thesis supervisor Dr. Souidi Mohammed Elhabib for his patience, presence, and, his availability during the months of work as well as for all the times that he guided us.

We would also like to thank the teachers of Computer Science at the University of Khenchela who provided the tools necessary for success during the university years.

Many thanks to our families, friends, and, colleagues for their unwavering support.

Dedication

I dedicate this work to my entire family, especially my parents and my brother.

I also dedicate it to all my friends and colleagues that I've been with my entire academic journey.

Without forgetting everyone who had helped me reach this point.

Sami

Dedication

I dedicate this work to my beloved deceased mother, my dear father, my siblings and their lovely children.

I dedicate it to my close friends, my distant relatives, and everyone that helped me.

I also want to dedicate it to my teacher and everyone that helped reach this point.

Amine

Table of Contents

Chapter 3: Simulations & Results

List of Figures

List of Tables

General Introduction

Multi-Agent Systems are computerized environments where several agents can interact with each other as well as elements in the environments, these agents may have different objectives that may or may not be conflicting with each other, but they can also have a common objective where they either cooperate or even compete to achieve said objective, this study focuses on cooperative agents with a common goal, where agents form groups in which the work together to complete the required tasks allowing the entire group to succeed in their goals as a team.

The forming of coalitions is a prolific topic when it comes to multi-agent systems in the current landscape as it is a cornerstone when it comes to the cooperation between agents, the coalitions are generally formed to solve problems that an agent cannot achieve by itself, and there many papers that seek to study different ways of forming coalitions in different scenarios to enhance the task completion through a better organization.

Pursuit-Evasion (PE) games are a distinctive area of distributed artificial intelligence dealing with the problem of cooperative decision-making in multi-agent systems. It focuses primarily on cooperative approaches used to connect many autonomous agents acting as pursuers to capture mobile evaders. The pursuit problem requires organizing the pursuers in a manner that makes the capture easier for the coalition members finishing the task quicker.

In this work, we propose a pursuit of coalition formation based on a new extension of the AGRMF (Agent Role Group Membership Function) organizational model. Specifically, we will process the problem linked to the negative externality where a pursuer might be forced into a group where it performs below its potential due to the forming of the chase group of the first evader in the set with the pursuers that suit the most even if it contains pursuers better suited elsewhere, creating a bias towards evaders higher ranked in the set.

Thusly this report is divided into the following manner:

In the first chapter, we will be taking a general view on multi-agent systems and some of their concepts in-line with our topic, such as the organization-centered systems where the focus lies in the coalition rather than individual agents, we'll also be taking a brief look at Markov Decision Process (MDP), as well as viewing some organizational models that have been aimed at the pursuit-evasion problem like the Agent Role Group (AGR) model, the Yet

Another Multi-Agent Model (YAMAM), and mainly AGRMF; we'll be extending. We will also be mentioning some previous papers that relate to our topic.

The second chapter will focus on the AGRMF model and its algorithm so that we can compare it to the extension we're proposing in the same chapter, focusing on the similarities and especially the differences that are present between the two, as we try to improve upon it with a different group access mechanism optimizing pursuers' performances.

Finally, the last chapter will contain a thorough comparison between the previously mentioned algorithms in different scenarios with multiple metrics about the group performances by measuring their capture time and individual pursuers' performance through their reward development and gains throughout the capturing process.

Chapter 1:

Multi-Agent Systems

I.1. Introduction

The notion of Multi-Agent Systems (MAS) is relatively new in the field of Distributed Artificial Intelligence (DAI), it has proven to be quite promising over the years it's been in existence, and this notion also become very popular in mainstream computer science.

Multi-agent systems (MAS) are a core area of research in contemporary artificial intelligence. A multi-agent system consists of multiple decision-making agents which interact in a shared environment to achieve common or conflicting goals. MAS research spans a range of technical problems, such as how to design MAS to incentivize certain behaviors in agents, how to design algorithms enabling one or more agents to achieve specified goals in a MAS, how information is communicated and propagated among agents, and how norms, conventions, and roles may emerge in MAS. A vast array of applications can be addressed using MAS methodologies, including autonomous driving, multi-robot factories, automated trading, commercial games, automated tutoring, etc.

In this first chapter, we shall be exploring Multi-Agent Systems, along with agents' organizing, as well as the Agent Group Role Membership Function (AGRMF) model which is the center of our study.

I.2. Multi-Agent Systems (MAS)

I.2.1. Definition

Multi-Agent Systems (MAS) are computer-based environments comprised of multiple intelligent agents that interact with one another. MASs are best used to solve problems that are difficult (or impossible) for a single agent to solve. There is no categorical definition of MAS, as there is for agents, so let us focus on one that is relatively consensual.

MAS is defined by Stone and Veloso [1] as “a loosely coupled network of problem-solving entities (agents) that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity (agent)”. [2]

The study, behavior, and construction of a collection of potentially pre-existing autonomous agents that interact with each other and their environments is the focus of MASs research [3]. The study of such systems extends beyond the study of individual intelligence to include problem-solving with social components. An MAS is a loosely coupled network of problem solvers that interact to solve problems that are beyond each problem solver's capabilities or knowledge. These problem-solvers, known as agents, are self-contained and can be heterogeneous.

I.2.2. Multi-Agent learning

Much of the literature on multi-agent learning has emerged from historically distinct communities, most notably reinforcement learning and dynamic programming, robotics, evolutionary computation, and complex systems. Existing surveys of the work have also tended to define multi-agent learning in ways that are unique to these communities. Instead, we'll start with a broad definition of multi-agent learning: it's the application of machine learning to problems involving multiple agents.

There are two aspects of multi-agent [4] learning that merit study as a separate field from traditional machine learning. First, because multi-agent learning deals with problem domains involving multiple agents, the search space involved can be unusually large; and because those agents interact, small changes in learned behaviors can frequently result in unpredictable changes in the resulting macro-level ("emergent") properties of the multiagent group as a whole. Second, multi-agent learning may involve multiple learners, each learning

and adapting in the context of others; this introduces unresolved game-theoretic issues into the learning process.

I.2.3. Agents

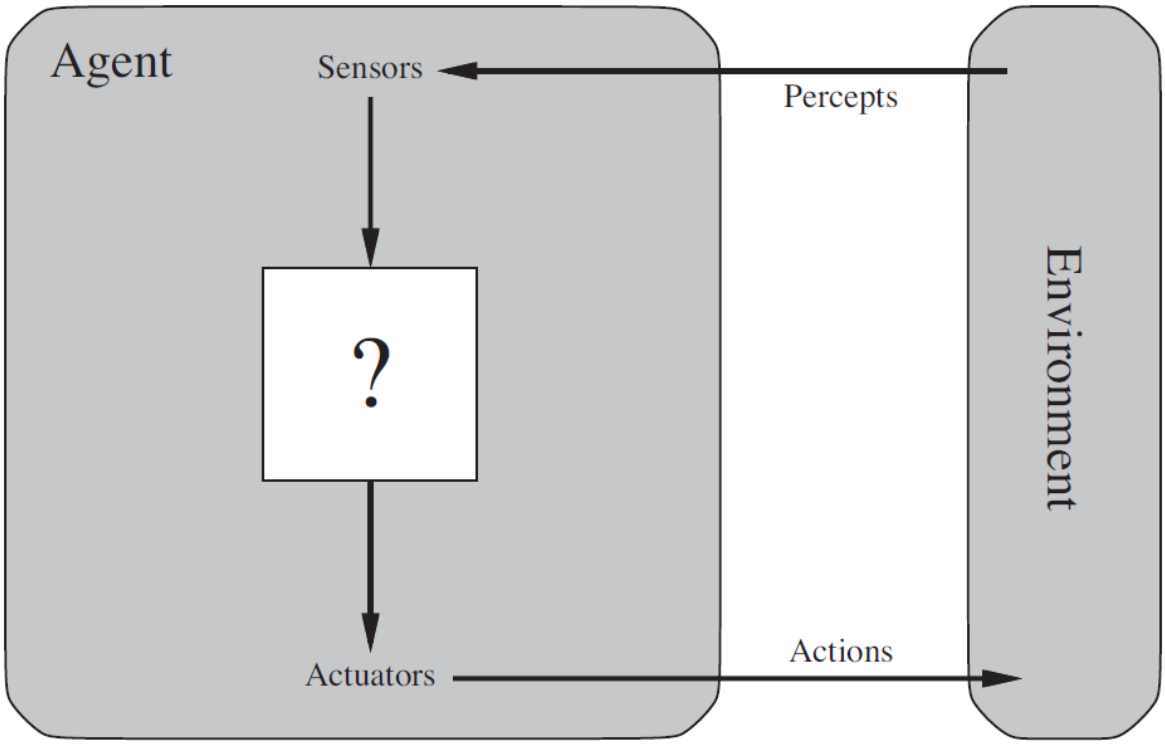

Figure 1:Agents interact with environments through sensors and actuators. [5]

"Agents" [6] are autonomous, computational entities that perceive their surroundings through sensors and act on their surroundings through effectors. To say that agents are computational entities simply means that they exist physically as programs that run on computing devices. They are autonomous in the sense that they have some control over their behavior and can act without the intervention of humans or other systems. Agents pursue goals or perform tasks to achieve their design objectives, and these goals and tasks can be both supplementary and conflicting.

"Intelligent" [6] indicates that the agents pursue their goals and execute their tasks such that they optimize some given performance measures. To say that agents are intelligent does not mean that they are omniscient or omnipotent, nor does it mean that they never fail. Rather, it means that they operate flexibly and rationally in a variety of environmental circumstances, given the information they have and their perceptual and effectual capabilities. A major focus of DAI, therefore, is on processes such as problem-solving,

planning, search, decision making, and learning that make it possible for agents to show flexibility and rationality in their behavior, and on the realization of such processes in multiagent scenarios.

"Interacting" [6] indicates that the agents may be affected by other agents or perhaps by humans in pursuing their goals and executing their tasks. Interaction can take place indirectly through the environment in which they are embedded (e.g., by observing one another or by carrying out an action that modifies the environmental state) or directly through a shared language (e.g., by providing information in which other agents are interested or which confuses other agents). DAI primarily focuses on coordination as a form of interaction that is particularly important to goal attainment and task completion. The purpose of coordination is to achieve or avoid states of affairs that are considered desirable or undesirable by one or several agents. To coordinate their goals and tasks, agents have to explicitly take dependencies among their activities into consideration. Two basic contrasting patterns of coordination are cooperation and competition. In the case of cooperation, several agents work together and draw on the broad collection of their knowledge and capabilities to achieve a common goal. Against that, in the case of competition, several agents work against each other because their goals are conflicting. Cooperating agents try to accomplish as a team what the individuals cannot, and so fail or succeed together. Competitive agents try to maximize their benefit at the expense of others, and so the success of one implies the failure of others.

I.2.4. Environment types

I.2.4.1. Fully observable vs. partially observable

A task environment [5] is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action; relevance, in turn, depends on the performance measure. Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world. An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data, if the agent has no sensors at all then the environment is unobservable.

I.3. Markov Decision Process

A Markov Decision Process (MDP) is a stochastic sequential decision-making method. [7] Sequential decision-making is applicable any time there is a dynamic system that is controlled by a decision-maker where decisions are made sequentially over time. MDPs can be used to determine what action the decision-maker should make given the current state of the system and its environment. This decision-making process takes into account information from the environment, actions performed by the agent, and rewards to decide the optimal next action. MDPs can be characterized as both finite or infinite and continuous or discrete depending on the set of actions and states available and the decision-making frequency. [7] The name Markov refers to the Russian mathematician Andrey Markov since the MDP is based on the Markov Property. In the past, MDPs have been used to solve problems like inventory control, queuing optimization, and routing problems. [8] Today, MDPs are often used as a method for decision-making in reinforcement learning applications, serving as the framework guiding the machine to make decisions and "learn" how to behave to achieve its goal.

It is a stochastic process $\{X_t, t \in T\}$ is a set of random variables indexed by a parameter t (discrete-time) and defined on the same probability space. The variable X_t represents the state of the process at time t. Moreover, the set of all possible values for this variable is called the state space of the process and will be denoted S.

A discrete-time Markov chain is a discrete-time stochastic process $\{X_t, t = 0, 1 \ldots\}$, defined on a finite space of states S and satisfying the following Markov property:

$$\forall\, i \in S\,, \forall\, t > 1\, [\, X_t = i \mid X_0, \ldots, X_{t-1}] = P[\, X_t = i \mid X_{t=i} \mid X_{t-1}] \qquad (01)$$

In such a process, known as a memoryless process, predicting the future from the present does not require knowledge of the past. Markov chains are used to model stochastic systems but do not allow an agent to intervene and act on the evolution of the system.

Markov decision processes are an extension of Markov chains to model the dynamics of a system under the control of an agent through actions and assign rewards to transitions between states

The Markov Decision Process (MDP) provides a model formalism and solves planning and learning problems under uncertainty. Suppose we have, on the one hand, a system that changes over time like a probabilistic automaton and on the other hand, a kind

of external controller that examines the current state of the system. This controller can use several actions for a purpose. His goal is to build an optimal plan to maximize the reward he can get. The evolution of the system is considered a Markov process, i.e. the evolution of a temporal sequence of distinct states according to transition probabilities, based solely on the phenomena of the previous state. Puterman [9] describes these models as located at the confluence of decision and probability theories. PDMs enable sequential decisions to be made under uncertainty.

The MDP is composed of two distinct elements: on the one hand, a Markov process, and on the other a controller element. At each step, the system controller observes the process and can act on it, making a possibly null action.

Reasoning based on the MDP can lead to the following discourse: "I know that I am in such a situation and if I do such an action, there is so much chance of being in this new situation by obtaining such a reward". The MDP is based on four components, namely states, actions, transitions, and rewards.

I.4. Organizational Models in Multi-Agent Systems

An organizational model is a task coordination mechanism for multi-agent systems. This model allows agents to cooperate and share tasks to be able to solve complex problems. All this is done within the framework of a metamodel which makes it possible to define the relations between the agents as well as the role of each one. [10]

I.4.1. Organizations in MASs

This viewpoint of MAS engineering is what is called organization-centered MAS (OCMAS) in which the structure of the system is given bigger attention through the explicit abstraction of agent organization. With that approach, the designer designs the entire organization and coordination patterns on the one hand, and the agents' local behaviors on the other hand. It is considered a top-down approach because the organization abstraction imposes some rules or norms used by agents to coordinate their local behaviors and interactions with other agents.

The OCMAS viewpoint has been promoted by many pioneers in MAS research. For instance, Jennings and Wooldridge [11] stated that MAS contributes to the software engineering (SE) discipline as a way to simplify the design of complex software systems but

considering MAS with no real structure isn't suitable for handling current software systems complexity, and higher-order abstractions should be used and some way of structuring the society is typically needed to reduce system complexity, to increase system efficiency, and to more accurately model the problem being tackled. Odell et al. [12] stated that the current practice of MAS design tends to be limited to individual agents and small face-to-face groups of agents that operate as closed systems which is not adequate to model and design complex adaptive systems. Also, Gutknecht and Ferber [13] argued that taking organizational concepts, such as groups, roles, structures, dependencies, etc, as first-class citizens, and relating them to the behavior of agents is a key issue for building large-scale and complex systems. In another article, Ferber [10] also stated that a MAS as an organization consists of roles enacted by agents arranged (statically or dynamically) to form groups of agents, and can handle many drawbacks such as system complexity, uncertainty, and system dynamism.

Gasser [14] stated that we simply have hardly any real experience building truly heterogeneous realistically coordinated multi-agent systems that work together and almost no basis for systematic reflection and analysis of that experience. Further, Horling et al. [15] stated that our real-world getting more complex and highly distributed and that should be reflected in new software engineering paradigms such as MAS. Therefore, the adoption of higher-order abstract concepts like organizations, societies, communities, and groups of agents can reduce systems complexity, increase efficiency, and improve system scalability.

Establishing an organizational structure that specifies how agents in a system should work together helps the achievement of effective coordination in MAS [16]. Broek [17] stated that the complexity of real-world applications needs to be tackled from higher abstraction order such as organizations that can be used to limit the scope of interactions, provide strength in numbers, reduce or manage uncertainty, and formalize high-level goals which no single agent may be aware of. Further, Hübner [18] confirmed that organizations provide a framework for structuring and managing agents' interactions and serve as a kind of tuning of the agents' autonomy level. Furthermore, Burns et al. [19] stated that in organization theory [20] [21], it is commonly accepted that different types of organizational structures are suitable for particular environmental conditions and one of the main reasons for creating organizations is to provide stable means for coordination that enable the achievement of global goals.

Moreover, Corkill et al. [22] stated that as agent-based systems become more widespread and complex, the designed organization will become an important aspect of

effective system performance, and they suggested the possible situations where organization design will be very important such as a large number of agents, long duration of agent activities, more repetitive activities, more activities require shared resources, more collaborative the activities, more specialized agents, less capable agents, and less slack resources are available. Also, they emphasized that no one organization is right for every situation.

In nutshell, proposing a way for statically or dynamically organizing MAS, has been given great attention by MAS researchers, as a promising approach for handling the challenging issue of engineering complex and large-scale software systems. The adoption of the OCMAS viewpoint mainly depends on the nature of the application domain and the degree of system complexity. The developers interested in top-down system reconfiguration will prefer the OCMAS approach.

The OCMAS viewpoint is more adequate for engineering complex adaptive multi-agent systems, which are expected to be, soon, the mainstream approach for engineering large-scale and even ultra-large-scale application domains especially with the evolving topic of the Internet of Things (IoT) [23], which concerns devices capable to communicate via the Internet and manipulate an enormous amount of data. Examples of such application domains are CPS (Cyber-Physical Systems) [24], Smart Grids [25], global SCADA (Supervisor Control and Data Acquisition) [26], Pervasive Computing [27], Ubiquitous Computing [28], etc.

I.4.2. Organization types

In [29], [30], and [31] several types of organizations are distinguished:

- Group: several types of groups exist:

● Simple group: as soon as a group exists, we can have cooperative coordination to achieve a common and shared goal.

● Team: A collection of individuals who have been brought together to work together. In an organization, several teams are formed for communication reasons. In this definition, individuals belonging to a team must necessarily communicate with each other, which leads to the introduction of the notion of environment (framework in which agents exist and evolve).

- Interest group: each member has the same interests; they share information and cooperate to achieve a common goal.
- A community of practice: is formed when professionals come together and organize themselves to share information and experiences relating to their activities. The members of these communities can thus exchange and cooperate to jointly solve the problems with which they may be confronted, thus learning from each other and building common knowledge and practices together.

- Hierarchy where we distinguish:

- Simple hierarchy: based on a master/slave relationship, this type of organization is no longer used.
- Multi-level hierarchy: authority links form a tree. Control in this type of organization is very complex, for example, the problem of resource allocation or planning must be taken into account.

- Decentralized organization: it is a multi-division hierarchy where each top of a branch is an organization in its own right. The main difficulty in this type of organization is the integration of the different results from the different divisions.

- Market: this type of organization is based on the consumer/producer relationship. A special instance of the market is the Contract Net Protocol which is a protocol allowing the development and execution of a contract between a manager agent and a contracting agent. It involves agents interacting with each other during the development and execution of the contract employing performatives.

- Coalitions: A coalition is a short-term organization based on specific and contextual commitments allowing agents to benefit from their respective skills opportunistically.

I.4.3. Agent Group Role (AGR)

The AGR (also known as AALAADIN) model is based on three primitive concepts, Agent, Group, and Role that are structurally connected and cannot be defined by other primitives. They satisfy a set of axioms that unite these concepts. [10]

I.4.3.1. Agent

An agent is an active, communicating entity playing roles within groups. [10] An agent may hold multiple roles and can be a member of several groups. An important characteristic of the AGR model is that no constraints are placed upon the architecture of an agent or on its mental capabilities. Thus, an agent may be as reactive as an ant, or as clever as a human.

I.4.3.2. Group

A group is a set of agents sharing some common characteristic. [10] A group is used as a context for a pattern of activities and is used for partitioning organizations. Two agents may communicate if and only if they belong to the same group, but an agent may belong to several groups. This feature will allow the definition of organizational structures.

I.4.3.3 Role

The role is the abstract representation of a functional position of an agent in a group. [10] An agent must play a role in a group, but an agent may play several roles. Roles are local to groups, and a role must be requested by an agent. A role may be played by several agents.

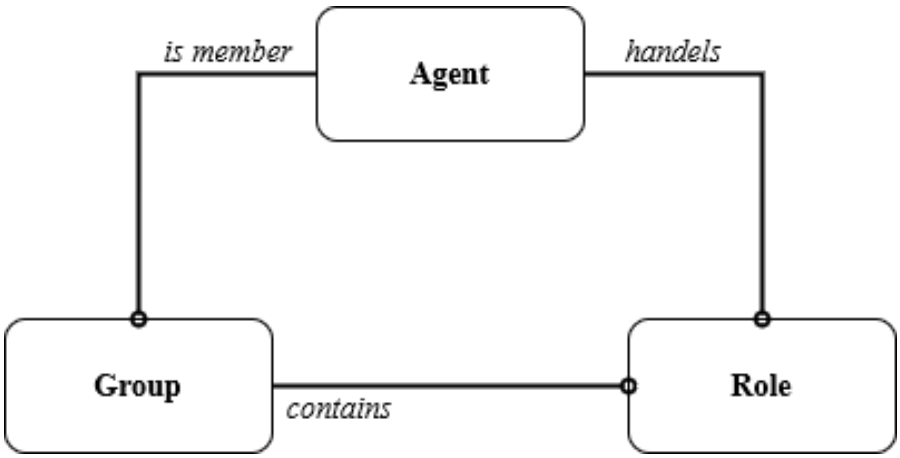

Figure 2: AGR core model

I.4.4. Agent Group Role Membership Function (AGRMF)

AGRMF [33] is a fully observable, multiagent, stochastic, episodic, static, continuous in a known environment flexible organizational model extension of the AALAADIN model by the introduction of fuzzy logic and Markov decision process ‘MDP’

principles that aim at the Pursuit problem, this model allows the interaction, organization, and re-organization of pursuers.

I.4.5. The YAMAM model (Yet Another Multi-Agent Model)

The YAMAM organizational model [34] is an alternative to other agent-based models such as AGR [43] and AGRMF [33]. The main advantages of this model come down to its good modularity and its possible scalability. Also, it can be interfaced with task scheduling tools, as will be used in this document. YAMAM is based on four different concepts: agent, skill, role, and task. In this model, the organization is described by its inherent structure. Therefore, agent relationships are paramount concerning agents and their behavior.

I.4.5.1. Agent

In this organizational structure, an agent is defined as an autonomous and communicating entity located in an explicit or implicit environment. The agent is built on reactive properties, which implies that skills cannot be added dynamically. However, it is possible to instantiate a cognitive agent capable of using reactive and cognitive skills to evaluate over time.

I.4.5.2. Role

The role reflects a service or agent identification form. The agent can manage one or more roles regarding a specific environment. Moreover, the role is based on a set of tasks to be performed in different ways. It is assumed that an agent can play a role only if it can perform the tasks involved, and therefore if it has the required skills. Typically, the role requires the ability to perform multiple tasks. Therefore, an agent must be characterized by the skills related to the tasks concerned. In this article, this concept is used to assign the role of "pursuer" which differs from one prosecution group to another. In addition, we note that each agent cannot play the role of "pursuer" in more than one group of pursuits.

I.4.5.3. Task

To be carried out, the task can be considered as the operation of skill or as an action requiring one or more skills. In the chase-escape game, the different chases represent the tasks to be performed. In addition, each escapee requires the capture of a specific number

and type of pursuers (skills). The following figure details the different relationships between the concepts that make up the YAMAM organizational model. However, unlike the AGR organizational model, the concept group is not physically implemented in this model. In other words, agents have no view of existing groups and are completely unaware of their membership information. In our proposal, the groups will be represented by the various coalition formations formed to capture detected escapees.

I.5. The pursuit-evasion problem

The pursuit-evasion has always been a popular topic in AI research, as many combinations of different scenarios and situations could be explored, it can be generally defined as the problem of capturing mobile target(s) with one or more pursuers, with the possibility of having obstacles in the environment using different strategies. There are four main types of problems that we'll be exploring below.

I.5.1. The single pursuer, single evader

This type of problem contains a single pursuer chasing after a single mobile target throughout the environment, here the pursuer and the evader can move at the same speed, like in Yamashita et al. [35] where the evader has unbound speed or Alonso et al [36] Lion ad Man problem where the pursuer possesses a higher speed.

I.5.2. Multi-pursuers, single evader

Here there are multiple pursuers attempting to catch a single mobile evader, in his type, the pursuer can be competitive where the first pursuer to capture the target is the winner or in a cooperative environment like in Wan et al. [37] where the pursuers form flocks that coordinate between its members to corner and apprehend the evader.

I.5.3. The single pursuer, multi-evader

In this type, a single pursuer tries to catch multiple evaders, the evaders can move independently from one another, or they can cooperate such as in the case of Salmon et al. [38].

I.5.4. Multi-pursuer, multi-evader

This type has many different scenarios and might require one or more pursuers to capture each evader, where every evader requires a different number of pursuers to be captured resulting in a cooperative environment where pursuers form coalitions and coordinate to optimize the capture time which is the case in the AGRMF [33] model.

I.6. Related works

The Organization concept is often used in MAS in different ways such as in Holonic multiagent manufacturing systems [39], in which a two-dimensional self-organization mechanism was conceived taking into account structural and behavioral vectors to achieve truly evolutionary and flexible systems. Moreover, other works use an underlying organization to lead the coalition formation of the agents [40], [41].

Application of the pursuit coalition formation process based on MAS organizational models is a recent research activity regarding distributed artificial intelligence. In [42], the authors focused on Agent Group Role (AGR) organizational model [43] to extract a coalition formation algorithm allowing the alliances of the pursuers in different pursuit groups. Also, they demonstrated the effects of the pursuit groups' stability on the capturing time as well as on the pursuers' development during the execution of the tasks. In the same context, they proposed a flexible organizational model extended from AGR through the application of fuzzy logic principles to equip each pursuit group with a membership function used to determine the membership degree of each pursuer. This model is known as Agent Group Role Membership Function (AGRMF) [44], [45], [46]. In addition, they showcased the impact of this fuzzy access mechanism on the roles' attribution and on the dynamism of coalition formation, additionally, the authors of [47] also used the YAMAM (Yet Another Multi-Agent Model) model based on the AGR meta-model to propose a coalition formation algorithm based on organizational principles and applied to the pursuit-evasion problem that allows the alliances of the pursuers in different pursuit groups,

Concerning game theory, a pursuit coalition formation algorithm based on the iterated elimination of dominated strategy (IEDS) model was proposed to provide a pursuit task coordination mechanism [48]. This coalition formation algorithm is based on the iterated elimination of the dominated pursuit groups, causing a certain equilibrium between the pursuit groups selected and excluding any problem related to the negative externalities.

Furthermore, this approach showcases an interesting decentralized calculation of the possible coalition formations.

Application of coalition formation in PE games is very common, such as in [49] where the authors proposed a pursuers' coalition formation algorithm to improve the real-time computation of the optimal coalition. Knowing that this computation exponentially increases the number of the pretending pursuers. Specifically, this algorithm is based on greedy optimal gains, allowing the assignment of the pursuers in different coalitions.

I.7. Conclusion

In the chapter above, we have presented a general overview of multi-agent systems explaining what an agent is with some context on the MAS environments, although we've come across several definitions, we included the ones that bit better within our context to help us move forward in the next segments of this thesis.

The second section contained Markov Decision Process which gives decisions numerical values to help agents take better actions, this process is the one utilized in planning the trajectory of pursuer agents in AGRMF.

The next section discussed the organization in multi-agent systems with some previous organizational models that were utilized for the pursuit-evasion problem which is further explained properly in the next section where it's defined comprehensively along with the different types associated with it as well as mentioning some prior examples.

The final section goes through several previous papers that share some relevance to our topic that helped in both better understanding our topic and provided insight into how to solve our problem most efficiently.

Using the knowledge gained in this chapter, we'll try to introduce a new access mechanism that changes how coalitions are organized to AGRMF in hopes of yielding better capture times.

Chapter 2:

The Proposed Organization Mechanism

II.1. Introduction

In this chapter, we'll be proposing a new group access mechanism for the AGRMF algorithm based on a fair approach to coalition formation. The main purpose of this approach is to form balanced coalitions where pursuers can chase after evaders that may be closer it rendering the capture time faster and more balanced, to do that we must eliminate the issue where the earliest evader in the set monopolizes the closest pursuers giving it fast capture time while the remaining evaders can take much longer to be captured due to their optimal pursuers joining the first evader's coalition.

The main goal is to solve the issue of negative externality that plagues the AGRMF model. A negative externality is where an agent joins a coalition where its possible performance decreases instead of joining a coalition where it can reach or at least gets close to its full potential.

While the mechanism that will propose may decrease the performance of certain groups, it will allow the coalitions to be more balanced leading to all pursuits finishing at approximately the same time instead of having one pursuit take much longer than others.

To achieve that task, we must first understand the pursuit-evasion scenario that we are facing, as well as a proper understanding of AGRMF and its components.

II.2. Agent Group Role Membership Function (AGRMF):

The authors of [33] created an extension of the AGR model that changed the groups' vision, providing methodological guidance which enables their flexibility within a multi-agent system. For that purpose, it introduced some principles of fuzzy logic. The groups in this model are equipped with a membership function calculating the membership degree of each agent concerning each group. Moreover, they have extracted a dynamic pursuit coalition formation algorithm from this model, in which the membership function dynamically determines the access to the role pursuer proposed by the pursuit group concerned.

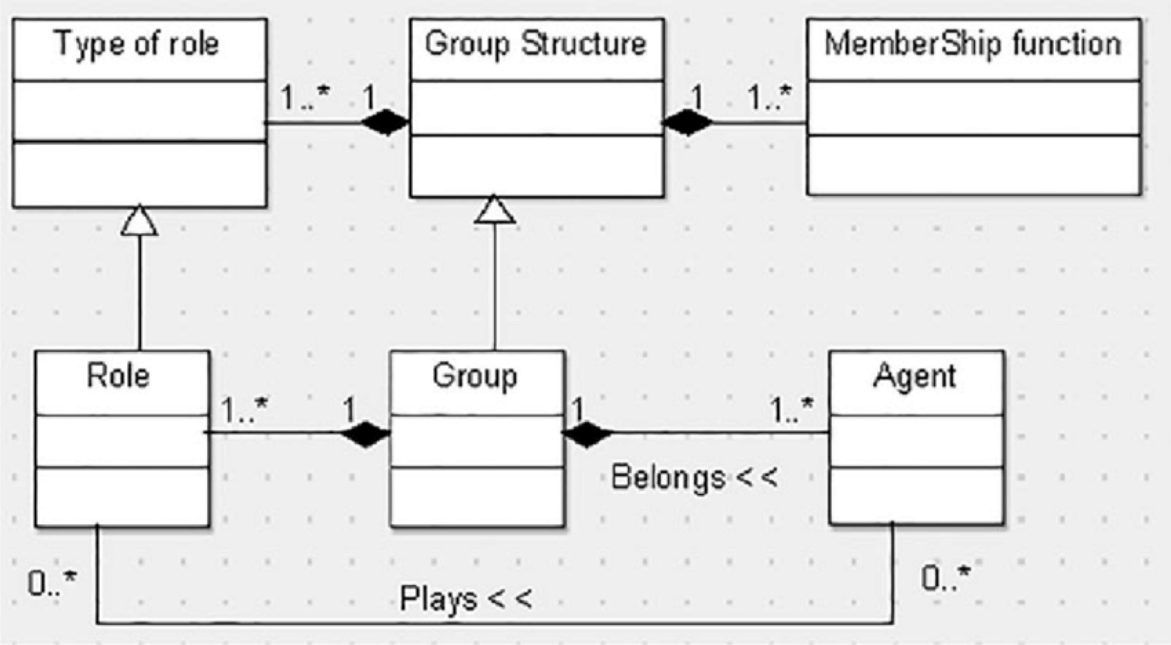

Figure 3: AGRMF meta-model

II.3. The pursuit problem

In this type of problem [33], the agents are split into two different roles, the evaders, which are agents moving freely and completely independently around the environment, and the pursuers, which form coalitions to cooperate in catching the evaders, these two roles, as well as the method of forming coalitions will be explained more thoroughly later on.

II.3.1. The pursuer

The set of pursuers that is responsible for the capturing is denoted by $P = \{P_1; P_2; P_3; \dots; P_n\}$ where each pursuer possesses its capacity parameters explained below.

II.3.1.1. Self-confidence degree

A metric that represents the agent's competence, calculated using the number of tasks that the agent has successfully finished C_s, and C_t; the number of tasks in which the agent has participated, this parameter is denoted as follows:

$$\forall Conf \in (0.1; 1): Conf = max\left(0.1, \left(\frac{C_s}{C_t}\right)\right) \quad (02)$$

II.3.1.2 Credit

Each agent receives credit after tasks, the credit is affected negatively if the agent cannot complete a task, the credit is denoted bellow with C_b being the number of the abandoned tasks by the agent:

$$\forall Credit \in (0; 1): Credit = min\left(1, 1 - \left(\frac{C_b}{C_t - C_s}\right)\right) \quad (03)$$

II.3.1.3 The distance

The position of the agent in the environment is a very important criterion for the pursuit sequences, because if the pursuer is closer to the evader, then the capture will be easier. The position Pos is computed as follows:

$$Pos = \min\left(Dist\left(S_p; S_E\right)\right) \quad (04)$$

where $Dist$ is the distance between the pursuer and the evader, S_p is the state (cell) of the pursuer, S_E is the state (cell) of the evader.

$$Dist\left(S_p; S_E\right) = \sqrt{\left(Co_{p_i} - Co_{E_i}\right)^2 + \left(Co_{p_j} - Co_{E_j}\right)^2} \quad (05)$$

where $\left(Co_{p_i}, Co_{p_j}\right)$ is the Cartesian coordinates of the pursuer, and $\left(Co_{E_i}, Co_{E_j}\right)$ the Cartesian coordinates of the evader.

II.3.2. The evader

The set of evaders which are mobile agents that move in random directions is denoted by $E = \{E_1; E_2; E_3; \dots; E_n\}$.

II.3.3. The Membership function

Using fuzzy logic, the degrees of membership are admitted to a given set. These values represent how well an agent fits in a certain role:

$$\mu_{Group}(Agent) = \frac{Coef_1 \times Pos + Coef_2 \times Conf + Coef_3 \times Credit}{\sum_{i=1}^{3} Coef_i} \quad (06)$$

II.3.4. AGRMF Algorithm

---

(01) Create-MemFu();

(02) Broadcast (Pos);

Waiting-Response;

(03) SendResponse (Pos, $Conf$, $Credit$);

(04) $\mu_{Group}(P_i)$

(05) $List1$: contains the membership degree of each agent pursuer.

$List2$: empty list.

Nbr: number of agent pursuers.

$\lambda \leftarrow 0$;

Repeat

If $\left(P_j \notin List2\right)$ && $\left(\mu\left(P_j\right)\right) = Max(List1)$ **then**

Add $(P_j, List2(E_i))$;

Delete $(P_j, List1)$;

$\lambda \leftarrow \lambda + 1$;

End if;

$i \leftarrow i + 1$

Until $\lambda = x$

Else

Waiting

End if;

(06) **If** $P_i \in List2$ **then**

Join group;

(07) Launch of the chase:

$C_{life} = \begin{cases} e^{-klt^2} & t \leq T_{max} \\ 0 & t > T_{max} \end{cases}$;

(08) $C_{life} = 0$

If $(capture == true)$ **then**

Go to step (10);

Else

Go to step (02);

End if;

(10) For each pursuer

$r(S_t) \leftarrow r(S_n)$;

$C_t \leftarrow C_t + 1$;

$C_s \leftarrow C_s + 1$;

The algorithm above's steps are explained in the following manner: the organizer evaluates the parameters of each evader agent and cerates n groups of the alliance. Also, the organizer integrates a membership function for each group of the alliance created (01). Each group represents the coalition to capture one evader. The organizer broadcasts n messages addressed to all the agents who play the role of pursuer. Each message will contain the position Pos and Re of one evader agent. Furthermore, the organizer waits for a specific moment until the reception of all the pursuers' responses (02). Each pursuer must respond to the offer made by the organizer and sends a response message (03). The organizer

calculates the membership degree of each pursuer in each group (04). The organizer enrolls x pursuer agents for each group where $x = Re$, and where Re is the number of pursuers that the evader needs to be captured. As a consequence, the organizer must choose x pursuer agents which have the highest membership degree (05). The organizer sends a response to all the pursuers who responded about its first broadcast. Each pursuer will handle the response of the organizer (06). At the launch of the chase, the organizer must follow the development of the pursuit. This chase will not last forever; the alliance has a life value defined by C_{life} (07). When $C_{life} = 0$ and the evader is captured, the organizer updates the ability of each P pursuer who participated in the chase (10). If the evader is not captured then the organizer reassigns the groups again after each episode.

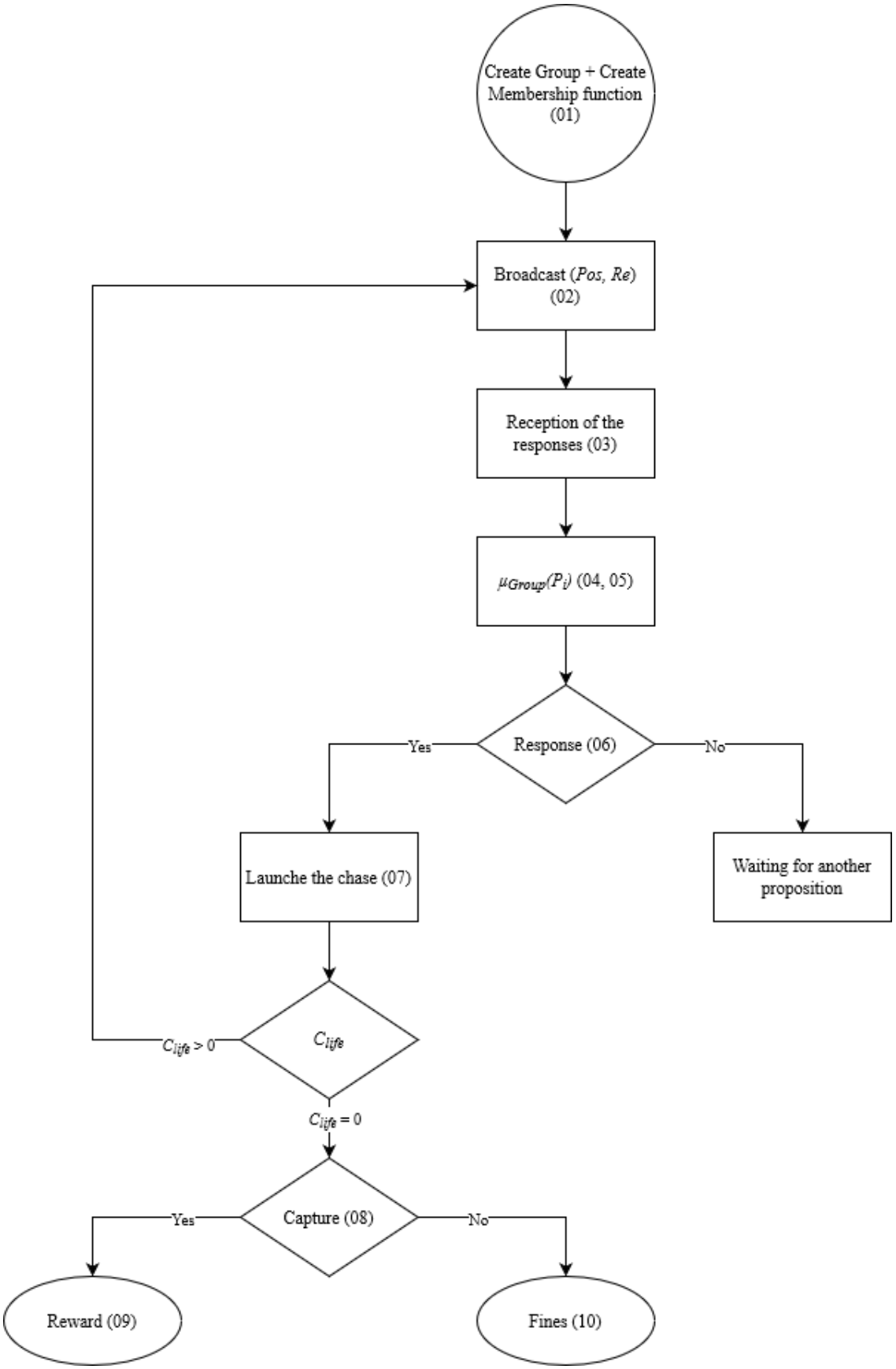

Figure 4: AGRMF flowchart

II.4. The proposed pursuit coalition formation algorithm

In our proposition, we are looking to further optimize the Organizer, to minimize the capture time, through a more even distribution of pursuers. The Organizer in the AGRMF model allows the first in the set of evaders to be pursued by the pursuers in the most optimal position (closest distance) completely disregarding the remaining evaders, as opposed to our proposed method of allowing all evaders in the set to add a single pursuer to their respective groups at a time resulting in a fairer pursuit.

II.4.1. The Organizer

The Organizer is the entity responsible for admitting pursuers into groups, in our proposition, the organizer goes through the evaders one by one, choosing the closest pursuer and admitting it to the chase group of the current evader, this process is repeated as many times as needed to fill the chase groups.

The main difference between our proposition and AGRMF is that in AGRMF the evaders receive higher priority based on their placement in the set, allowing the first evader to take up the most optimal pursuers leaving the remaining evaders with scraps, in our proposition, however, each evader has a chance of having closer pursuers due to adding a single pursuer to each group at a time while repeating this until all groups are filled, resulting in an even distribution of pursuers.

II.4.2. Proposing the algorithm

---

(01) Create-MemFu();

(02) Broadcast (Pos);

Waiting-Response;

(03) SendResponse (Pos, $Conf$, $Credit$);

(04) $\mu_{Group}(P_i)$

(05) $List$: contains the membership degree of each agent pursuer.

$x = 0$;

Repeat

For $i = 0$ **to** n **do**

Select $Max(P_j)$ from $List$;

Add P_j to $Group(E_i)$;

Delete P_j from $List$;

$i \leftarrow i + 1$;

End for

$x \leftarrow x + 1$;

Until $x = Re$.

(06) Launch of the chase:

$$C_{life} = \begin{cases} e^{-klt^2} & t \leq T_{max} \\ 0 & t > T_{max} \end{cases};$$

(07) **If** $(capture == true)$ **then**

End;

Else

Go to step 02;

End if;

The algorithm steps shown in the flowchart below are explained as follows; the organizer integrates the membership function into the alliances needed after evaluating the parameters of the evader agents. The organizer sends as many messages as there are groups to the pursuer agents containing the Pos of each evader, after that it waits for a certain amount of time for the response from the pursuers (02). The pursuers now must respond to the organizer with a message containing the Pos, $Conf$, and $Credit$ (03). The organizer now uses the membership function to compute the degree of membership of each pursuer (04). The organizer goes through the set of evaders one by one, adding a single pursuer to the current evader's chase group, this pursuer has the highest membership degree of the available pursuers. When a pursuer gets added to a group chase, it is automatically removed from $List$ which contains all the available pursuers. This process is repeated Re times, where Re is the number of pursuers that the evader needs to be captured (05). The organizer must keep track of the chase's progress from the start. This hunt will not continue

indefinitely; the alliance has a C_{life} value (06). When C_{life}= 0 and the evader is captured, and the organizer ends the chase (07). If the evader is not captured; the organizer must recreate new groups based on the current situation restarting the chase again.

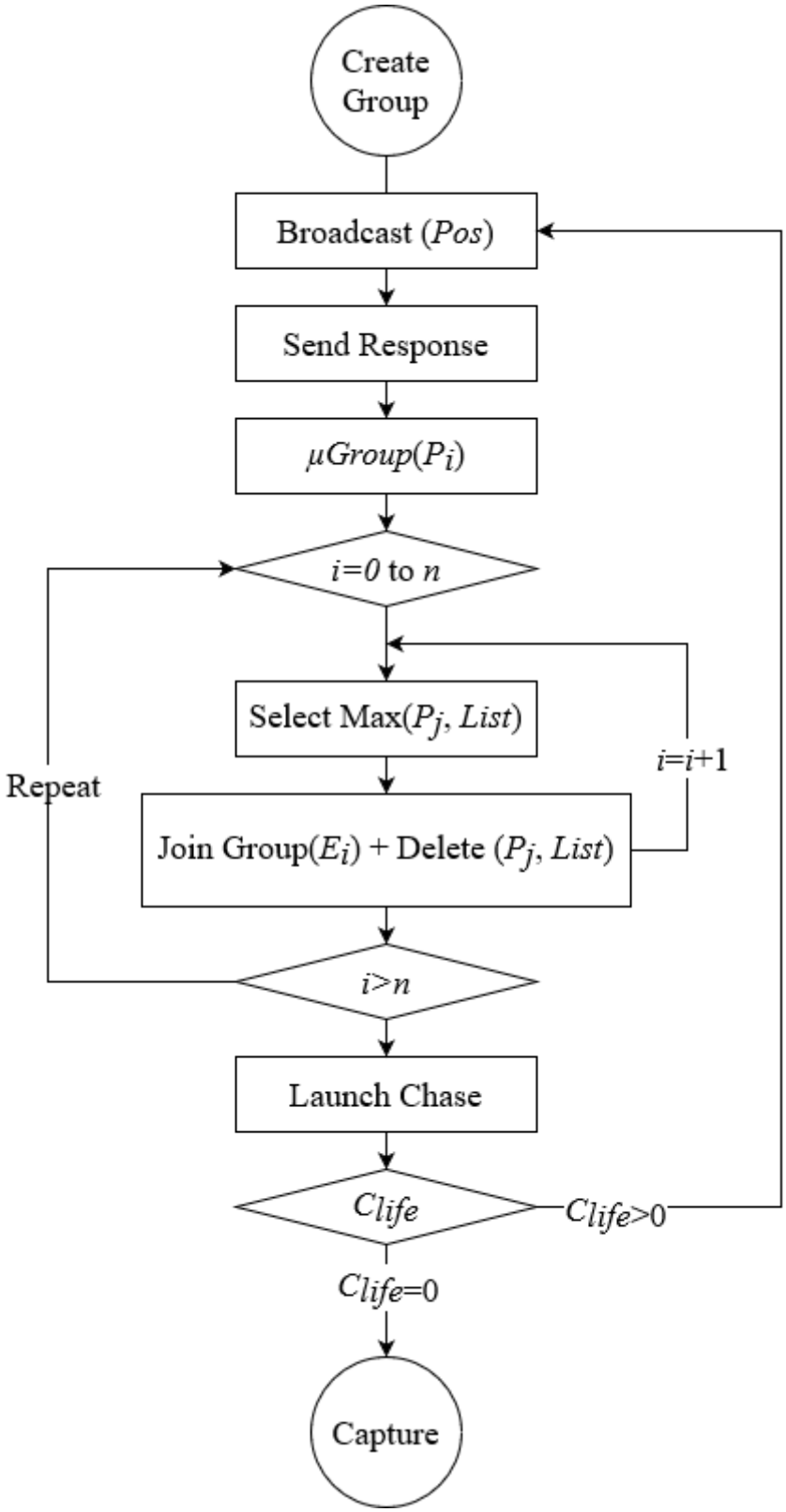

Figure 5: Fairness-based AGRMF flowchart

The main difference between the two algorithms is how they distribute the pursuers; our proposition seeks to increase the efficiency by allowing the pursuers to join coalitions where they are better suited, rather than joining a coalition that only suits the first evaders in the set, this is demonstrated by better through the following example.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
$Group1$	23%	50%	45%	80%	61%	20%	30%	55%
$Group2$	28%	44%	54%	80%	62%	23%	17%	90%

Table 1: Role attribution

In this example, the coalitions formed with AGRMF are $Group1: (P_4, P_5, P_8, P_2) = 61.5\%$, $Group2: (P_3, P_1, P_6, P_7) = 30.5\%$, while the Fairness-based algorithm forms the coalitions like so $Group1: (P_4, P_5, P_2, P_7) = 55.3\%$, $Group2: (P_8, P_3, P_1, P_6) = 48.8\%$. even though the fair distribution caused $Group1$'s attribution to drop by 5%, it more than made up for it by increasing $Group2$'s attribution by over 18%.

II.5. Pursuers path planning

In this section, we will explain how the pursuers are moving in the environment according to MDP principles where agents try to end up the maximum reward possible at the end of the chase through the quadruplet (S, A, T, R) where S is the space of states [42], A is the space of actions, T represents a transition function where $T(s, a, s')$ is the probability of reaching a state $s' \in S$ from an initial state $s \in S$ after executing an action $a \in A$. And the reward function R where $R(s, a) \in R$ is the reward of executing action $a \in A$ from a state $s \in S$. A policy is a function $\pi: S \to A$ that associates each state S with a corresponding action A as so:

$$\pi(s_t) = a_t \tag{07}$$

The value function of policy π links each state with the accumulated rewards gained from the state through the function $V^{\pi}: S \to R$.

$$V^{\pi}(s) = E\left[\sum_{k=0}^{\infty} \gamma^k R\left(s_k, \pi_{s_k}\right) | s_0 = s\right] \tag{08}$$

The discount facture $\gamma \in [0,1]$ reduces the emphasis on distant rewards. A fundamental property of policy π is that it checks a recursive equation of Bellman:

$$V^{\pi}(s) = R(s, \pi(s), s') + \gamma \cdot \sum_{s'} T(s, \pi(s), s') \cdot V(s') \quad (09)$$

The primary goal of MDP is to establish the best policy. We take notice of V^*, the optimal value function, which associates the best possible reward with each state.

$$V^*(s) = max_{\pi} V^{\pi}(s) \quad (10)$$

We can easily determine the optimal policy π by selecting the greedy policy among the values of V^* if the optimal value function is known:

$$\pi^*(s) = arg \max_{a \epsilon A} R(s,a) + \gamma \cdot \sum_{s'} T(s,a,s') \cdot V^*(s') \quad (11)$$

Our pursuers' movement strategy is entirely centered on the best policy π. Given $V^*(s)$, an agent can maximize its reward by employing the "greedy" strategy of choosing the action with the highest reward. This technique is greedy because it uses $V^*(s)$ as a substitute for immediate reward before maximizing its immediate profit. It is ideal since the function V^* is a precise summary of future rewards.

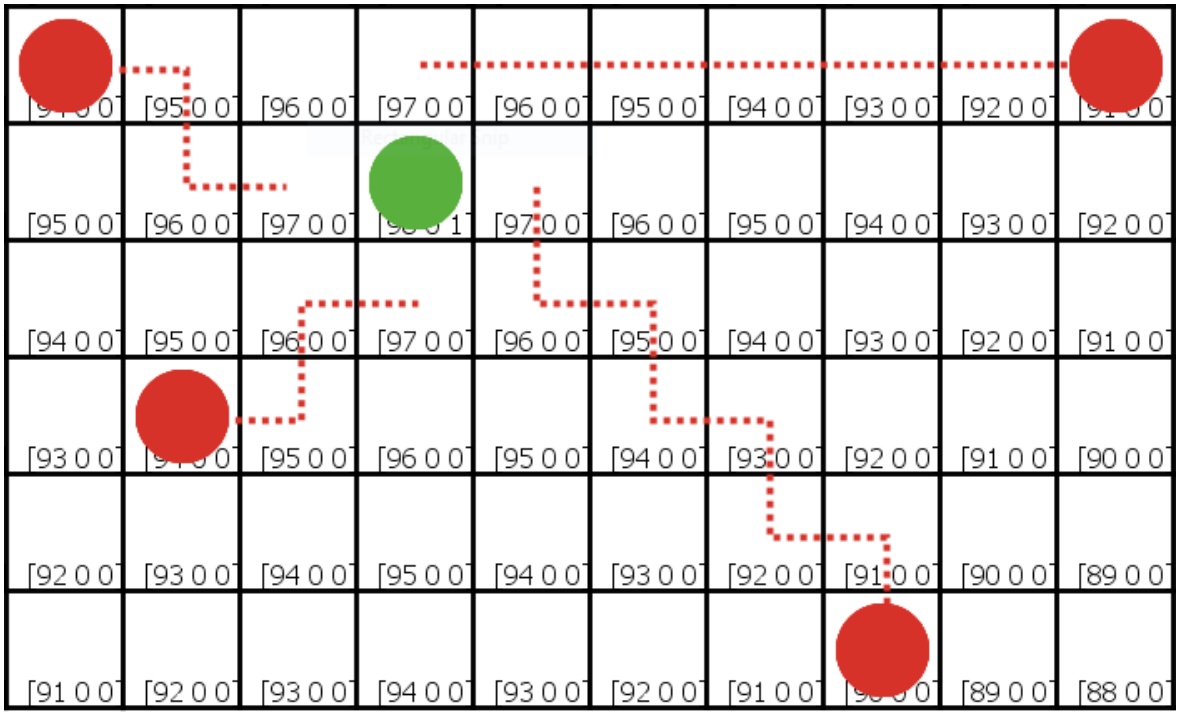

Figure 6: Pursuers' motion strategy. Red agents: pursuer. Green agent: evader

II.6. Conclusion

This chapter goes through a comprehensive overview of AGRMF and the pursuit problem in the first and second sections respectively, the first section went through the definition of AGRMF and its main components. The second section explains the pursuit problem and its components that are utilized in AGRMF while providing the algorithm used for the Pursuit-Evasion problem in AGRMF as well as an explanation of the mechanics of the algorithm through both text and a flowchart

The third part is where we presented our solution while thoroughly explaining how our proposed algorithm works as well as providing an example of how it solves the negative externality using the fair distribution of pursuers to create more balanced coalitions, even if this solution may not be perfect, it makes the role attribution of the created groups a lot more balanced making the individual pursuits finish within a relative time frame. The final quarter of this chapter explains the use of MDP in planning the trajectory of the pursuers.

Using what we created in this chapter, we'll simulate both algorithms to find whether what was proposed was able to provide it with tangible improvements or not using the NetLogo platform and run simulations in three different scenarios to verify the improvements' consistency.

Chapter 3:

Simulations &

Results

III.1. Introduction

In this chapter, we'll introduce the concept of Agent-Oriented Programming highlighting the differences between it and classical programming paradigms. we'll also be taking a look at the multi-agent platforms that use said paradigm which we chose for this study; NetLogo denoting its history and presenting both its components and its features.

We shall also be discussing the results of the comparison between the Fairness-based AGRMF that was discussed in the previous chapter, and AGRMF as well as mentioning the testing methodologies for the different scenarios, the scenario with a dynamic evader where both pursuers and evader move at the same speed throughout the entire chase. In the second scenario, evaders can make a single move at the beginning of the chase and then remain static for the remaining time. The final scenario will have an evader that's dynamic and moves twice as fast as its pursuers increasing the difficulty of the chase.

The results will be displayed visually through charts that'll help us makes an objective comparison between the two approaches, the results will also be summarized in a table to highlight the differences that we'll be finding.

III.2. Agent-oriented programming

Agent-oriented programming was proposed by Yoav Shoham in 1993 [51] as a new programming paradigm, which can be seen as a specialization of object-oriented programming. In this approach, agents are the central elements of the language, in the same way, that objects are central for object-oriented languages. The perspective on agents is cognitive: agents are characterized by mental notions such as their beliefs, decisions, and obligations. In addition, each agent is associated with a set of skills that represent what the agent can do. At the same time, agent-oriented programming assumes that we will develop programs in which several agents interact, which emphasizes the social dimension of agents. The programming language proposed by Shoham as a demonstration of this new paradigm is called AGENT0.

The main difference between such a language and a classical programming language that one could use to develop agents, comes from the fact that the mental notions which characterize the agents appear in the language itself, and that the semantics of the language is intimately linked to the semantics of these mental notions. Agent-oriented programming can be seen as a specialization of object-oriented programming because program modules are now agents, i.e., objects with a state that defines the associated mental notions, and the messages between objects are replaced by messages between agents. How do messages between agents differ from messages between objects? Firstly, these messages are modeled based on speech act theory, which is concerned with communicative actions such as informing, asking, offering, accepting, rejecting, and more. Second, since agents are autonomous and mentally capable, they have the freedom to decide whether or not to perform the action specified in the message. In contrast, an object receiving a message will always perform the action specified in the message.

III.2.1. The NetLogo Platform

Simulation platforms that allow rapid prototyping and testing of initial design ideas and alternatives can significantly improve the process of implementing complex agent and multi-agent systems (MAS). Such a platform should ideally feature a short learning curve, simple implementation, and display of the MAS under development while taking into account agent-oriented programming features to transfer design alternatives to a full agent

development environment without any hassle. However, these specific requirements make this simulation platform a perfect learning tool.

NetLogo [52] was created by Dr. Uri Wilensky at the Center for Connected Learning and Computer-Based Modeling (CCL), located at Tufts University, Medford, MA. In 2000, CCL was transferred to Northwestern University in Evanston, Illinois, where NetLogo processing has continued to this day.

NetLogo is a simple and primitive multi-agent programming language based on the Logo programming language. This platform is distributed in open-source in its current version. The design and development of NetLogo were carried out with the aim of modeling and simulating complex systems changing over time. NetLogo has been used to design and implement several complex systems in different fields such as economics, biology, physics, chemistry, psychology, system dynamics, and many other natural and social phenomena.

Both mobile and immobile agents can be modeled by NetLogo. It deals with multiple agents belonging to the same physical space and can also include thousands of agents in the same simulation. In addition, there are other capabilities including agent and network binding capabilities, two-dimensional and three-dimensional (2D and 3D) options regarding display modes, as well as easy switching between one-step execution and continuous simulations. The features distinguishing NetLogo could be summarized as follows: [53]

- Clear and complete documentation, containing a large and rich set of examples (example models) concerning different fields to guide and reinforce the objectives of the users.
- NetLogo offers an easy-to-understand Graphical User Interface (GUI), through which a set of simple tools is available to produce a model, monitor it, and control its behavior.
- This platform offers an easy-to-learn, powerful, and flexible modeling language for creating models.
- A HubNet participatory simulation tool allowing multiple users running separate client programs to interact through a NetLogo model. Also, it contains several tools that make it easier to run multiple programs in the background.
- Behavioral space is a tool that facilitates running a model repeatedly through the use of different inputs to study the impacts of alternative scenarios.
- Icon editors and possibilities for importing links and agents.

- A separate environment with appropriate tools for graphically developing system dynamics models rather than agent-based models.

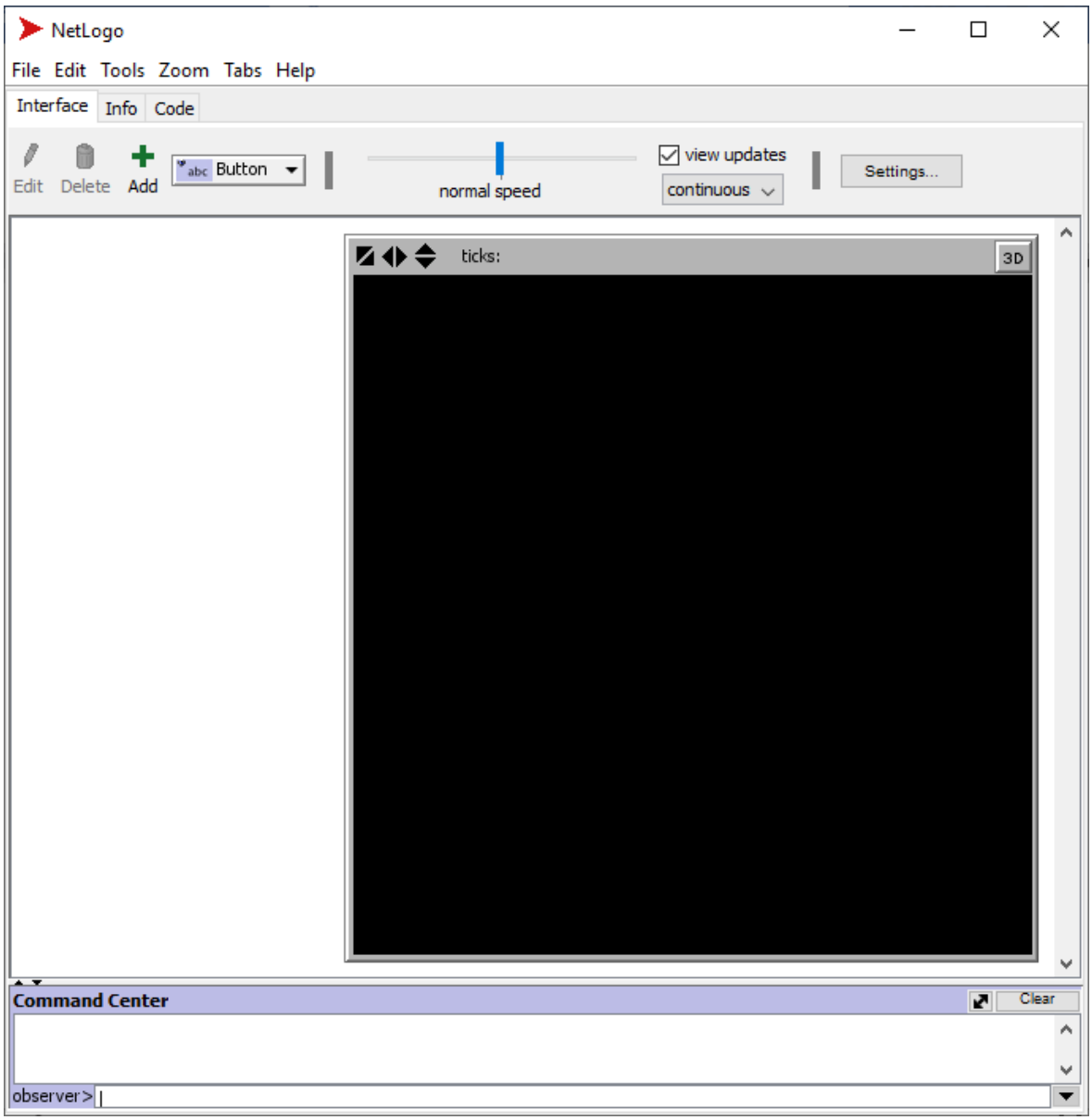

Figure 7: NetLogo's graphical interface

To study the application of NetLogo regarding modeling and simulation of mobile ad hoc networks simulating in particular higher phenomena of mobile ad hoc networks, the authors in [54] analyzed the programming language NetLogo and determined several characteristics and superiorities of the latter compared to conventional network simulators. They designed and implemented a realistic mobile ad hoc network model in NetLogo and then demonstrated its applicability to mobile ad hoc network modeling and simulation.

Also, through research, they concluded in [55] that NetLogo was able to effectively simulate the behaviors of the farmer. There is evidence in one case that when financial subsidies greatly increase farmers' income, the proportion of those violating the policies will decrease significantly, and in the other case, when financial subsidies greatly reduce farmers' income, the proportion of those who follow the policies will drastically decrease.

In recent research activities [56], NetLogo has been used to study the impact of ant colony optimization on the traffic problem, the traffic control problem is modeled as a multi-agent-multi- uses (MAMP).

During our work, we found the usefulness of using NetLogo to describe the pursuit-escape environment as well as the process of pursuing agent coalitions until the capture of the detected escapees. This platform offers the possibility of creating different types of agents which could be distinguished by several characteristics such as the degree of mobility defining the speed of movement and the paths to follow available, the shape, and the color defining the role awarded during the prosecution process.

A NetLogo program can be composed of different types of agents:

- Turtles: they are our agents, they have characteristics, they move around the world, they are born and die; they can be of several types (Breeds, consumers, firms)
- Patches: these are the spatial components of the "world" that is modeled; agents move over patches and patches can store variables. In other words, each patch characterized by Cartesian coordinates is considered a cell that can be occupied by an agent or an obstacle. The colors of the patches are modifiable allowing us to define free cells, obstacles, as well as the limits of the environment.
- Links: a particular type of agent which connects two agents and is represented as a line drawn between these agents. This link may or may not be oriented
- World: it corresponds to the spatial representation (in 2d or 3D) of the modeled environment.

NetLogo is the next generation in a series of multi-agent modeling languages, including StarLogo and StarLogoT. NetLogo runs on the Java Virtual Machine and therefore runs on all major platforms (Mac, Windows, Linux, etc.). It is run as a desktop application. Command-line operation is also supported.

III.3. Running the simulations

III.3.1. Simulation Environment

The environment for our tests was done in a 100x100 cell grid, containing ten Pursuer agents, and two Evader agents, the agents can move by a distance of one cell per episode in one of the four cardinal directions; north, south, east, and west, each evader needs a group of four Pursuer to be captured.

The simulations were run a hundred times to find these results.

III.3.3. Dynamic Simulations

The following chart depicts the result of running a 100 simulation or both the normal AGRMF and the fairness-based AGRMF.

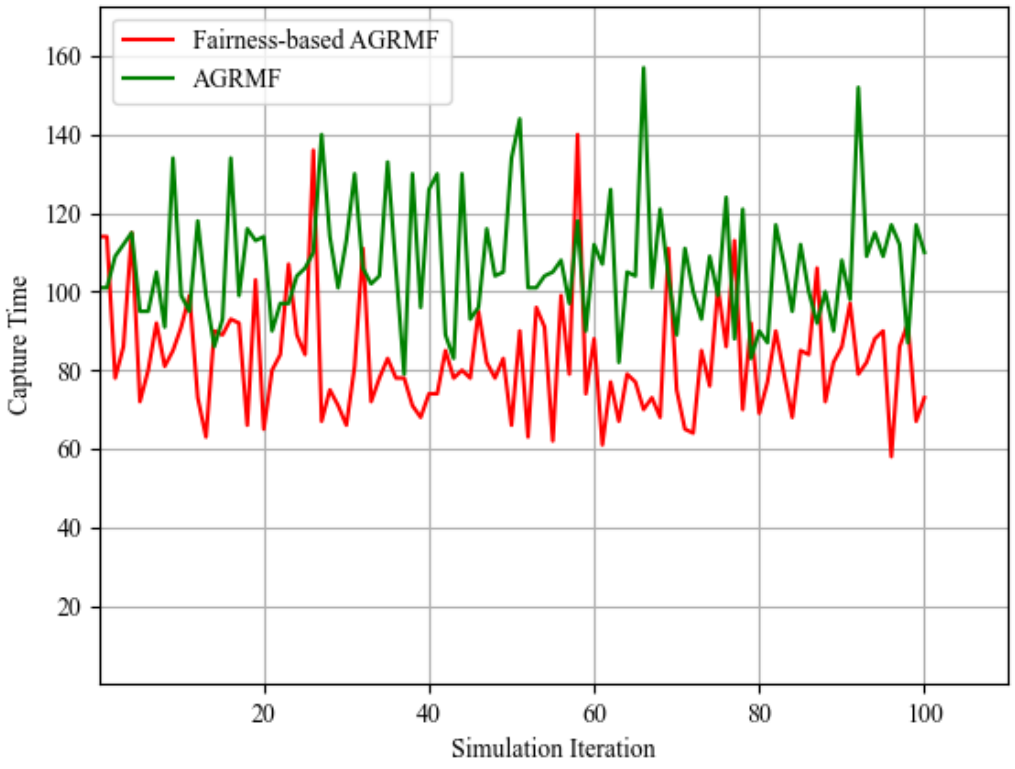

Figure 8: Capture time per run (Dynamic evader)

In the superseding chart, we see that the fairness-based algorithm performs consistently better than the normal AGRMF, as it has a lower capture time aside from a few

edge cases here and there, with an average of 83 for the fairness-based AGRMF, compared to the 107 of the normal AGRMF showing a 22.5% improvement overall, while the highest capture time for the fairness-based AGRMF is 140, it's still 17 episodes lower than the normal AGRMF which peaked at 157, and there's an even bigger difference of 21 episodes when it comes to their fastest times of 58 for the fairness-based AGRMF, and, 79 for the normal AGRMF.

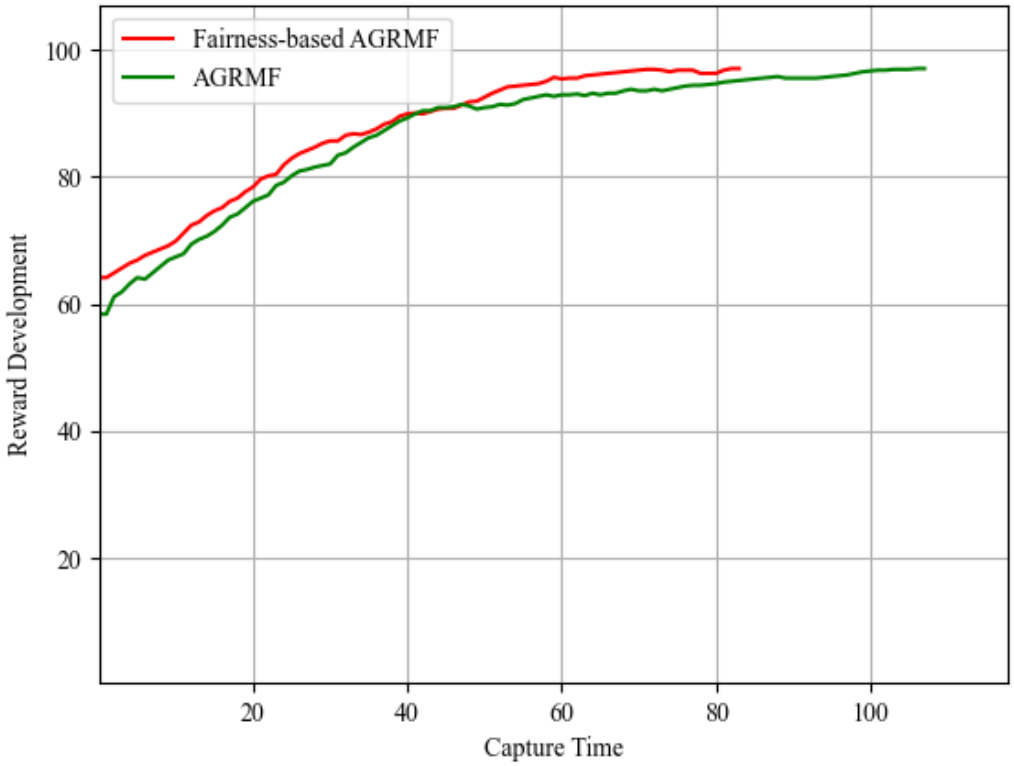

Figure 9: Average reward development (Dynamic)

The chart above depicts the reward development of a run with the same length as the average of each algorithm, we can see that even from the starting position, the fairness-based AGRMF has the advantage here, which it maintains until around the 40s when it crosses paths with the normal AGRMF even dipping below I occasionally, before it picks itself back up right around the ending of the 40s, until it completes the objective, while the normal AGRMF consistently remains below until the end.

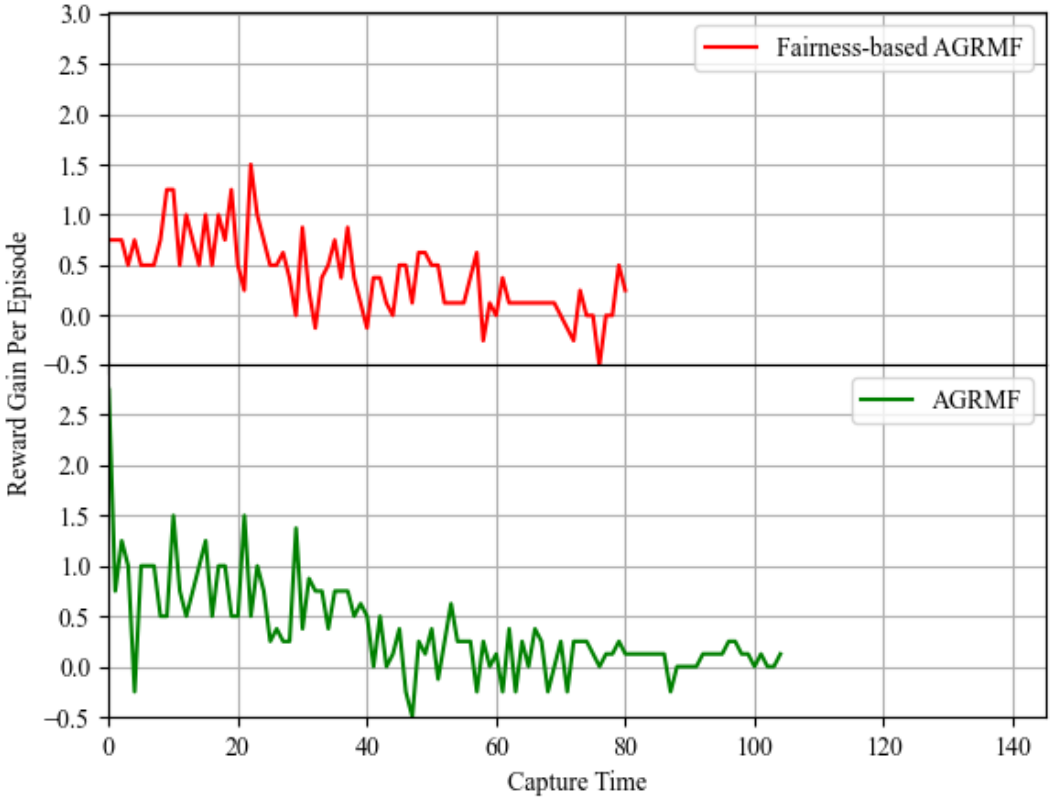

Figure 10: Average reward gain (Dynamic)

These results are extracted from the same run as the previous graph, the Fairness-based AGRMF suffers a loss in reward in 5 instances, about 6% of the time, while the old implementation loses rewards about 8.4% of the time, or in 9 instances. From this we can deduce that pursuers are more effective when they are distributed in a balanced manner where they can be a lot more effective.

III.3.4. Static Simulation

The static simulation included evaders that move by a single cell in a random direction at the beginning of the chase and remain static for the rest of it.

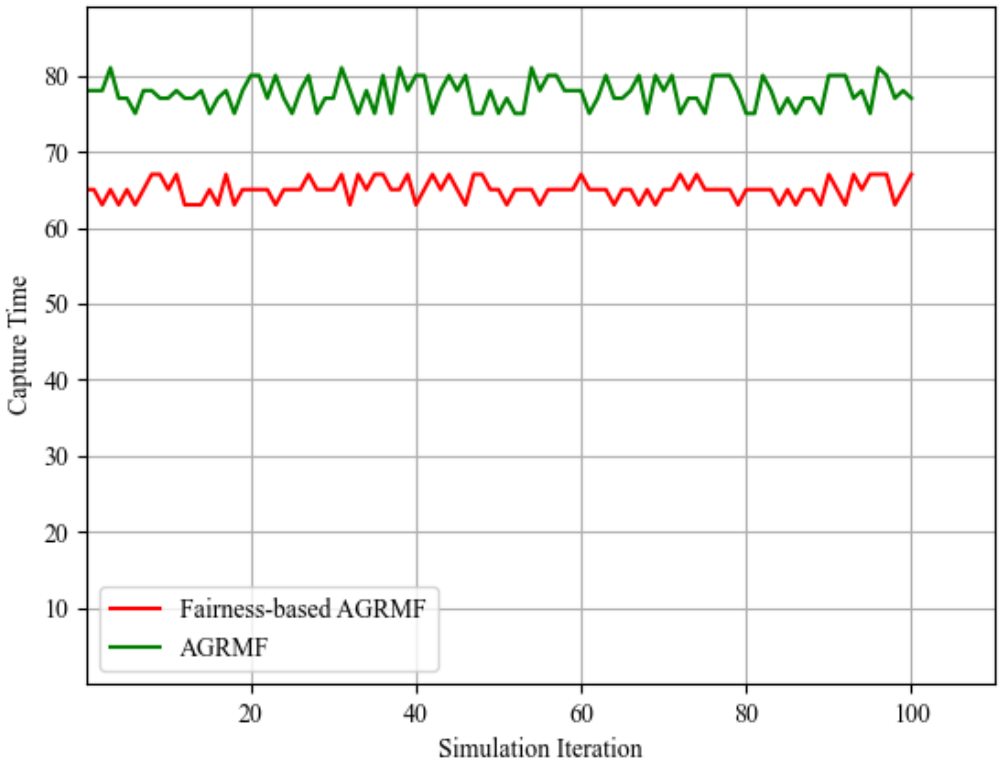

Figure 11: Capture time per run (Static evader)

The advantage of the fair distribution is a lot more apparent in this case, with an average of 65 episodes for the fairness-based AGRMF, as opposed to 77 episodes when it came to the normal AGRMF, netting it a 15.4% less capture, even if the decrease in capture time on average is less than the situation where the evader is dynamic, it is a lot more consistent as the fairness-based algorithm is always faster.

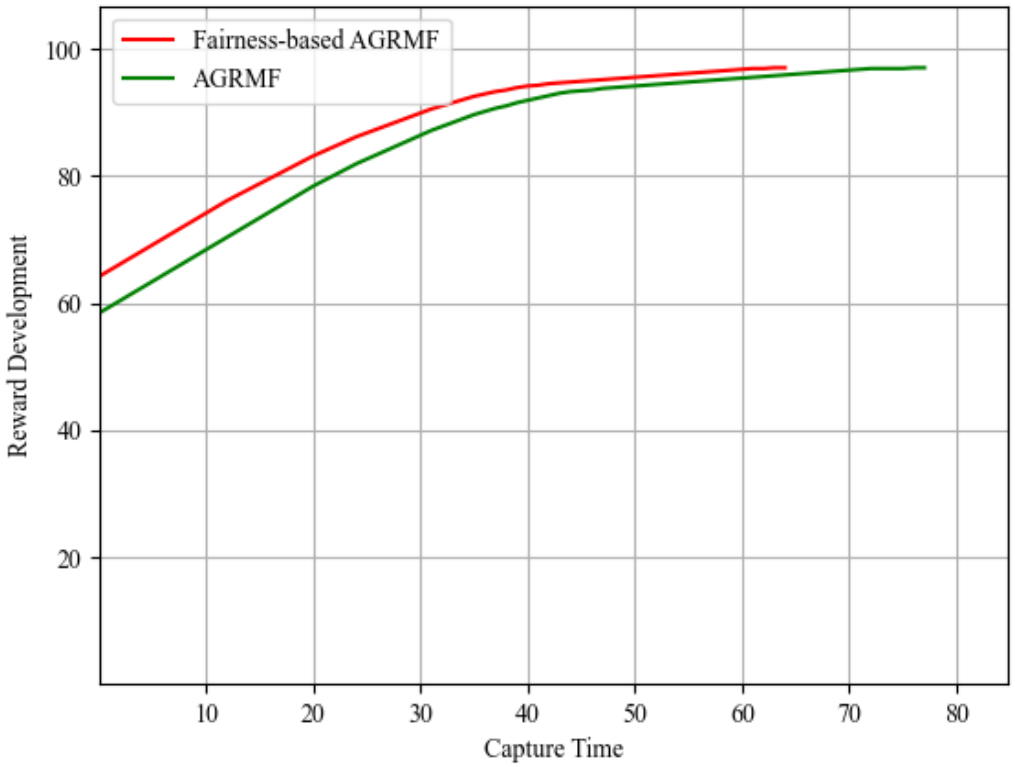

Figure 12: Average reward development (Static evader)

This chart tells the same story as its dynamic counterpart, with a consistent rise for both algorithms with a maintained advantage for the fairness-based AGRMF of the normal AGRMF from beginning to end.

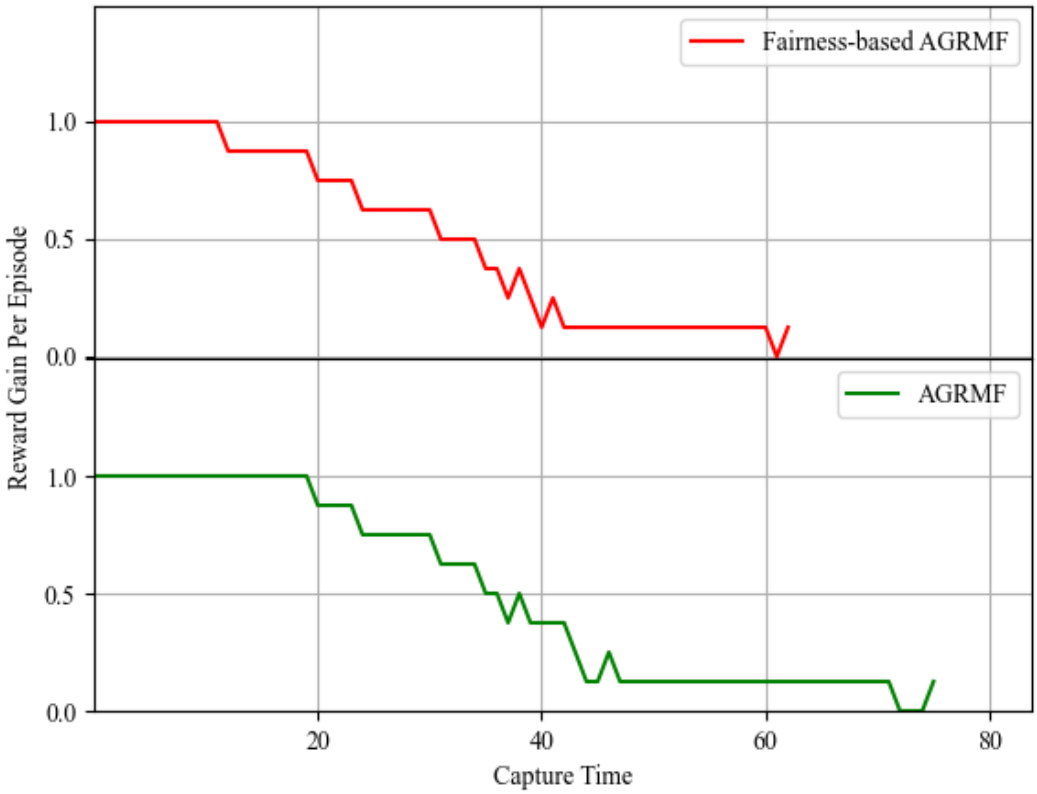

Figure 13: Average reward gain (Static evader)

Here we see that both instances maintain the same gain but for nearly twice the number of episodes for the normal AGRMF before a steep curve down in contrast to the much gentler one for the fairness-based AGRMF right before flattening around the later third for both of them until the end.

III.3.5. Fast Evader Simulations

The following graph depicts the result of running a 100 simulation or both the normal AGRMF and the fairness-based AGRMF where the evader is allowed to move twice each episode making it faster than the pursuers.

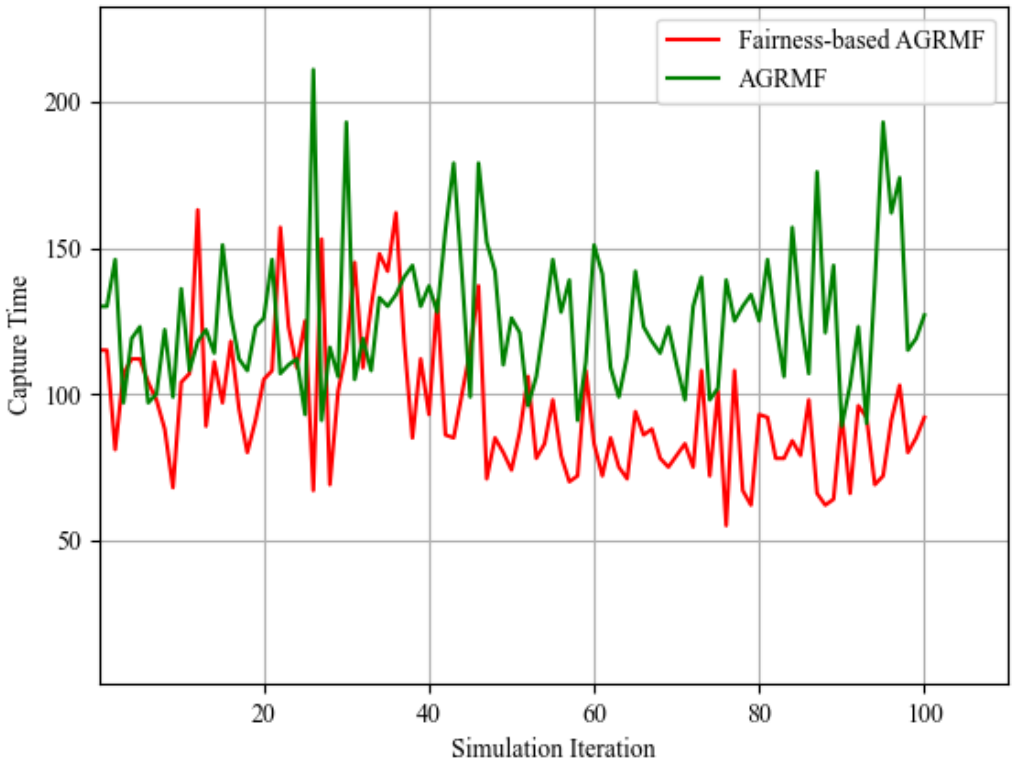

Figure 14: Capture time per run (Fast evader)

The previous chart shows the result that further solidifies our findings above, with a substantial difference of 31 episodes on average between our implementation of 95, and AGRMF with an average of 126, an improvement of around 24.6%. the difference is more apparent in their fastest runs with our implementation and AGRMF having 55 and 89 respectively, the same narrative is repeated in their slowest runs with the fairness-based AGRMF scoring 163 as opposed to the 211 episodes that took AGRMF to complete its slowest run, further highlighting the improving our implementation provided.

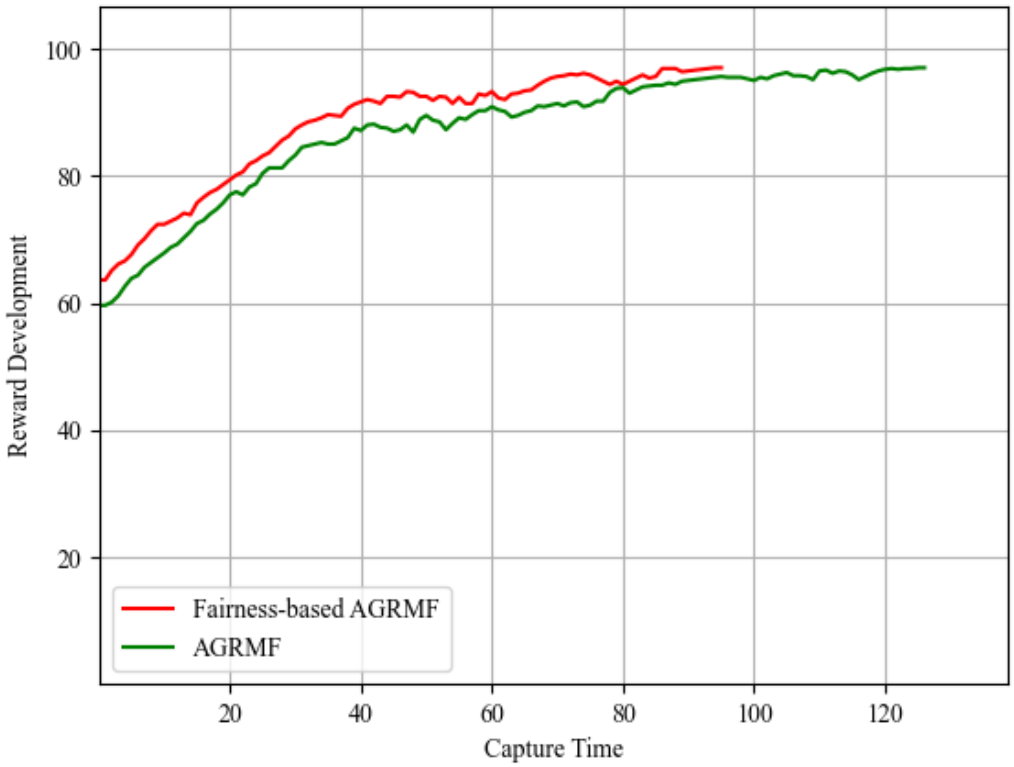

Figure 15: Average reward development (Fast evader)

This chart is also in line with previous results, with the Fairness-based AGRMF permanently staying above AGRMF, even though the charts here are a bit more erratic due to the inconsistency of speed between the pursuers and evader, it still defines a clear advantage for the coalitions that were formed in a fair manner, with the Fairness-based AGRMF remaining on top for the its entire run time.

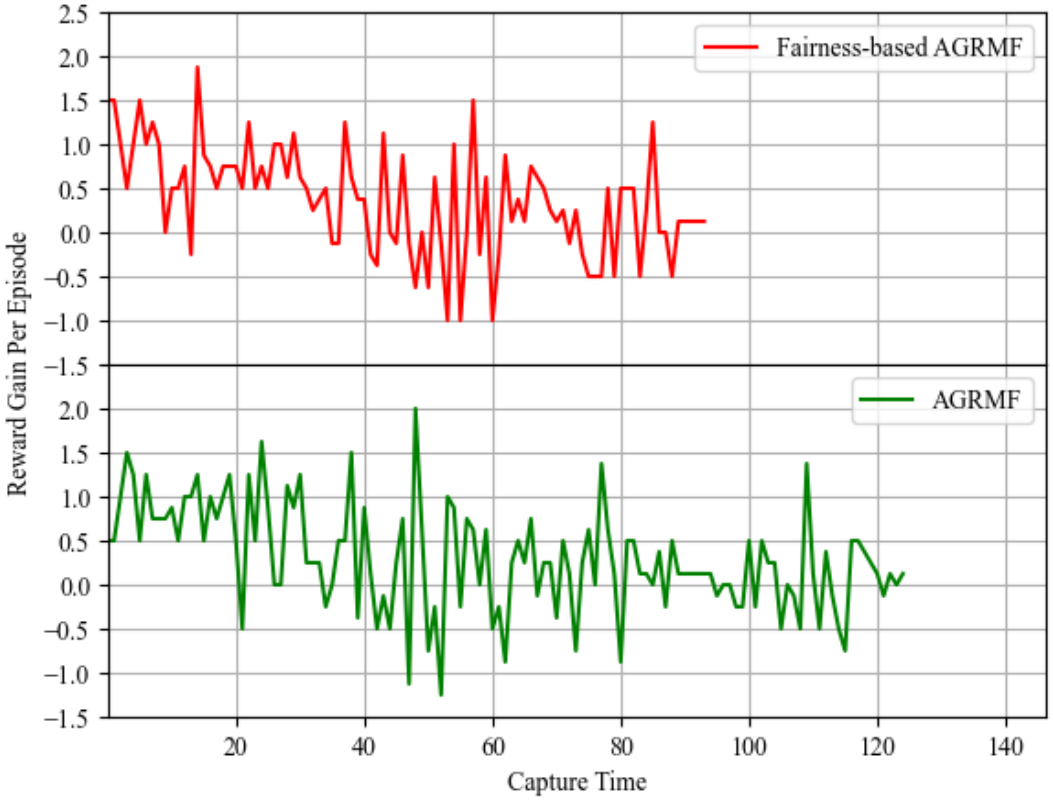

Figure 16: Average reward gain (Fast evader)

These charts are much more erratic compared to the previous ones, but we can still see that AGRMF has more reward losses with 18.25% of the instances or 28 times in the negative versus the 16.84% or 19 instances of the Fairness-based implementation, giving a slight advantage to that implementation.

III.4. Result summary

	Fairness-based AGRMF	AGRMF
Dynamic evader	83	107
	22.5%	
Static evader	65	77
	15.4%	
Fast evader	95	126
	24.6%	

Table 2: Final results

The previous table depicts the final results obtained from all the simulations, and it shows that the coalitions formed through the Fairness-based AGRMF brought an improvement of more than 20% proving that an even distribution of pursuers can dramatically increase the efficiency of the chase.

III.5. Conclusion

This chapter first introduced Agent-Oriented Programing and its history while providing a short comparison between it and the classical programming paradigms. Then we went through the NetLogo platform and its components and features, in its earlier sections, the NetLogo platform is the Agent-Oriented Programming platform used to run our simulations due to its easy-to-understand structure and the features that it provides that are especially useful to us.

The third section dictated the three different scenarios and explained the environment of where the pursuit takes place, the we presented the results of the simulations in three different scenarios in both a visual and a written manner, each scenario had its subsection allowing us to view and understand them separately before it summarized these results in a more digestible manner on the fourth section.

We used three different scenarios for the simulations so that we can verify that the improvements found weren't due to luck by scaling to the different scenarios where we found that the improvements gained through the different scenarios are around 20% across all three of them with the biggest increase in the hardest scenario of the fast evader.

The results we found were consistent throughout the different proving that the mechanism we proposed provided AGRMF with a sizable improvement in both the capture time and the pursuers' performance.

General Conclusion

This thesis seeks to improve the group access mechanism in AGRMF by introducing a fairness-based coalition formation method that tries to eliminate the negative externality that can poorly impact certain pursuers' performance consequentially making the capture time longer than what it could've been.

To carry out this work, we first collected the most documents concerning the subject. MASs being an important subject in distributed intelligence, there were abundant resources with lots of relevant information, and organization however being a relatively recent field, unfortunately, there were fewer papers related to this subject.

After that, we have extensively studied AGRMF and its various components so that we can make proper adjustments to achieve our desired results, this also necessitated a broader understanding of the pursuit-evasion game and how the distribution of the pursuers affected the capture time of each evader. This helped us find the proper algorithm that we could reach a satisfying result.

For the simulations, we devised three different scenarios with different types of evaders and compared the result of both our implementation and AGRMF so that we can find decisive results. All of this was done on NetLogo 5.0.4.

Finally, we have depicted the results of the simulations that reflect the difference between the two implementations through a visual medium of charts, and with written descriptions, these results proved advantageous for the fair distribution of pursuers proposed in this work over the old method of organization used in AGRMF.

We managed to find satisfying results that displayed the advantage that a fair organization has, by lowering external negativity, most pursuers managed to gain an increase in performance by allowing them to join more optimal coalitions allowing them to unlock most of—if not all of their potential.

References

[1] Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, *8*(3), 345-383.

[2] Rocha, J., Boavida-Portugal, I., & Gomes, E. (2017). Introductory chapter: multi-agent systems. In *Multi-Agent Systems*. IntechOpen.

[3] Durfee, E. H., & Lesser, V. R. (1989). Negotiating task decomposition and allocation using partial global planning. In *Distributed artificial intelligence* (pp. 229-243). Morgan Kaufmann.

[4] Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, *11*(3), 387-434.

[5] Norvig, P. R., & Intelligence, S. A. (2002). A modern approach. *Prentice Hall Upper Saddle River, NJ, USA: Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. Knowledge-Based Systems*, *90*, 33-48.

[6] Weiss, G. (Ed.). (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press.

[7] Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, *2*, 331-434.

[8] Feinberg, E. A., & Shwartz, A. (Eds.). (2012). *Handbook of Markov decision processes: methods and applications* (Vol. 40). Springer Science & Business Media.

[9] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[10] Ferber, J., Gutknecht, O., & Michel, F. (2004, July). From agents to organizations: an organizational view of multi-agent systems. In *International workshop on agent-oriented software engineering* (pp. 214-230). Springer, Berlin, Heidelberg.

[11] Jennings, N. R. (2000). On agent-based software engineering. *Artificial intelligence*, *117*(2), 277-296.

[12] Odell, J. J., Dyke Parunak, H. V., & Fleischer, M. (2002, May). The role of roles in designing effective agent organizations. In *International Workshop on Software*

Engineering for Large-Scale Multi-agent Systems (pp. 27-38). Springer, Berlin, Heidelberg.

[13] Gutknecht, O., & Ferber, J. (1998). *A model for social structures in multi-agent systems*. Technical report, LIRMM/CNRS, Université Montpellier II.

[14] Gasser, L. (2001). Perspectives on organizations in multi-agent systems. *ECCAI Advanced Course on Artificial Intelligence*, 1-16.

[15] Horling, B., & Lesser, V. (2004). A survey of multi-agent organizational paradigms. *The Knowledge engineering review*, *19*(4), 281-316.

[16] Barber, K. S., & Martin, C. E. (2001, May). Dynamic reorganization of decision-making groups. In *Proceedings of the fifth international conference on Autonomous agents* (pp. 513-520).

[17] Broek, E. L., Jonker, C. M., Sharpanskykh, A., & Treur, J. (2005, July). Formal modeling and analysis of organizations. In *International Conference on Autonomous Agents and Multiagent Systems* (pp. 18-34). Springer, Berlin, Heidelberg.

[18] Hubner, J., Vercouter, L., & Boissier, O. (2009). Instrumenting multi-agent organisations with artifacts to support reputation processes. *Coordination, Organizations, Institutions and Norms in Agent Systems IV*, 96-110.

[19] Burns, T., & Stalker, G. (1961). *The Management of Innovation*, Tavistock, London.

[20] Posey, Rollin B. (March 1961). "Modern Organization Theory edited by Mason Haire". *Administrative Science Quarterly 5* (4): 609–611.

[21] Hertz, D. B., & Livingston, R. T. (1950). Contemporary Organizational theory: A review of current concepts and methods. *Human Relations*, *3*(4), 373-394.

[22] Corkill, D. D., & Lander, S. E. (1998). Diversity in agent organizations. *Object Magazine*, *8*(4), 41-47.

[23] Mattern, F., & Floerkemeier, C. (2010). From the Internet of Computers to the Internet of Things. In *From active data management to event-based systems and more* (pp. 242-259). Springer, Berlin, Heidelberg.

[24] Rajkumar, R., Lee, I., Sha, L., & Stankovic, J. (2010, June). Cyber-physical systems: the next computing revolution. In *Design automation conference* (pp. 731-736). IEEE.

[25] Momoh, J. A. (2012). *Smart grid: fundamentals of design and analysis* (Vol. 63). John Wiley & Sons.

[26] Abbas, H. A. (2014). Future SCADA challenges and the promising solution: the agent-based SCADA. *International Journal of Critical Infrastructures*, *10*(3), 307-333.

[27] Saha, D., & Mukherjee, A. (2003). Pervasive computing: a paradigm for the 21st century. *Computer*, *36*(3), 25-31.

[28] Friedewald, M., & Raabe, O. (2011). Ubiquitous computing: An overview of technology impacts. *Telematics and Informatics*, *28*(2), 55-65.

[29] Mkadmi, A., & Saleh, I. (2008). *Bibliothèque numérique et recherche d'informations*. Lavoisier.

[30] Akoka, J., & Comyn-Wattiau, I. (2006). *Encyclopédie de l'informatique et des systèmes d'information* (p. 1941). Vuibert.

[31] Rahwan, T., & Jennings, N. R. (2007). An algorithm for distributing coalitional value calculations among cooperating agents. *Artificial Intelligence*, *171*(8-9), 535-567.

[32] Hannoun, M., Boissier, O., Sichman, J. S., & Sayettat, C. (2000). MOISE: An organizational model for multi-agent systems. In *Advances in Artificial Intelligence* (pp. 156-165). Springer, Berlin, Heidelberg.

[33] Souidi, M. E. H., Songhao, P., Guo, L., & Lin, C. (2016). Multi-agent cooperation pursuit based on an extension of AALAADIN organisational model. *Journal of Experimental & Theoretical Artificial Intelligence*, *28*(6), 1075-1088.

[34] Savall, M., Pécuchet, J., Chaignaud, N., & Itmi, M. (2001). YAMAM–An Organisation Model for the Multiagent Systems. In *Implementation in the Phoenix platform, 3rd Francophone Conference of Modeling and Simulation" Conception, Analyze, Management of Industrial Systems" MOSIM*.

[35] Yamashita, M., Umemoto, H., Suzuki, I., & Kameda, T. (2001). Searching for mobile intruders in a polygonal region by a group of mobile searchers. *Algorithmica*, *31*(2), 208-236.

[36] Alonso, L., Goldstein, A. S., & Reingold, E. M. (1992). “Lion and man”: Upper and lower bounds. *ORSA Journal on Computing*, *4*(4), 447-452.

[37] Wan, K., Wu, D., Zhai, Y., Li, B., Gao, X., & Hu, Z. (2021). An Improved Approach towards Multi-Agent Pursuit–Evasion Game Decision-Making Using Deep Reinforcement Learning. *Entropy*, *23*(11), 1433.

[38] Salmon, J. L., Willey, L. C., Casbeer, D., Garcia, E., & Moll, A. V. (2020). Single Pursuer and Two Cooperative Evaders in the Border Defense Differential Game. *Journal of Aerospace Information Systems*, *17*(5), 229-239.

[39] Barbosa, J., Leitão, P., Adam, E., & Trentesaux, D. (2015). Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution. *Computers in industry*, *66*, 99-111.

[40] Singh, V. K., Husaini, S., & Singh, A. (2010, November). Self-organizing agent coalitions in distributed multi-agent systems. In *2010 International Conference on Computational Intelligence and Communication Networks* (pp. 650-655). IEEE.

[41] Abdallah, S., & Lesser, V. (2004, September). Organization-based cooperative coalition formation. In *Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004. (IAT 2004).* (pp. 162-168). IEEE.

[42] Souidi, M. E. H., Piao, S., Li, G., & Chang, L. (2015). Coalition formation algorithm based on organization and Markov decision process for multi-player pursuit evasion. *Multiagent and Grid Systems*, *11*(1), 1-13.

[43] Ferber, J., Gutknecht, O., & Michel, F. (2003, July). From agents to organizations: an organizational view of multi-agent systems. In *International workshop on agent-oriented software engineering* (pp. 214-230). Springer, Berlin, Heidelberg.

[44] Souidi, M. E. H., Songhao, P., Guo, L., & Lin, C. (2016). Multi-agent cooperation pursuit based on an extension of AALAADIN organisational model. *Journal of Experimental & Theoretical Artificial Intelligence*, *28*(6), 1075-1088.

[45] Siam, A., Maamri, R., & Sahnoun, Z. (2014). FuzzyOrganization of Self-Adaptive Agents Based On Software Components. *International Journal of Intelligent Information Technologies (IJIIT)*, *10*(3), 36-56.

[46] Siam, A., Maamri, R., & Sahnoun, Z. (2012, July). An approach based on software components and mobile agents for developing distributed applications with verification of validity criterion. In *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems* (pp. 407-413). IEEE.

[47] Souidi, M. E. H., Siam, A., Pei, Z., & Piao, S. (2019). Multi-agent pursuit-evasion game based on organizational architecture. *Journal of computing and information technology*, *27*(1), 1-11.

[48] Souidi, M. E. H., & Piao, S. (2016). A new decentralized approach of multiagent cooperative pursuit based on the iterated elimination of dominated strategies model. *Mathematical Problems in Engineering*, *2016*.

[49] W. Hao *et al.*, "Pursuers-coalition Construction Algorithm in Multi-robot Pursuit-evasion Game", *ROBOT*, pp. 142–150, 2013.

[50] Souidi, M. E. H., Piao, S., Li, G., & Chang, L. (2015). Coalition formation algorithm based on organization and Markov decision process for multi-player pursuit evasion. *Multiagent and Grid Systems*, *11*(1), 1-13.

[51] Merabet, N. (2019). Proposition d'un mécanisme de coordination de tâches pour les SMAs basé sur la théorie des jeux (Masters thesis). Khenchela.

[52] U. Wilensky. NetLogo. Computer software. Northwestern University, 1999.

[53] D. Stigberg. An Introduction to the NetLogo Modeling Environment. Ecologist-Developed Spatially-Explicit Dynamic Landscape Models, Part of the series Modeling Dynamic Systems, 2012, pp. 27-41.

[54] M. Babiš, and P. Magula. NetLogo – an alternative way of simulating mobile ad hoc networks. Wireless and Mobile Networking Conference (WMNC), 2012, pp. 122–125.

[55] S. Xike, and L. Fengxian. An Agent-based Simulation of Farmer's Behaviors Adapting to Ecological Policies Using NetLogo. International Symposium on Information Science and Engineering (ISISE), 2012, pp. 426–429.

[56] K. Jerry, K. Yujun, O. Kwasi, Z. Enzhan, and T. Parfait. NetLogo implementation of an ant colony optimisation solution to the traffic problem. IET Intelligent Transport Systems, 2015, vol 9, pp. 862–869.