SIM-AQUARIUM
REPORT ABOUT INTELLIGENT SYSTEMS ENGINEERING PROJECT

Bedei Andrea(matricola 0001126957)
Bertuccioli Giacomo Leo(matricola 0001136879)
Notaro Fabio(matricola 0001126980)

03rd May 2025

Contents

Chapter 1

INTRODUCTION

This chapter reports on the proposed project:

- group members → Bedei Andrea, Bertuccioli Giacomo Leo, Notaro Fabio
- members' emails → andrea.bedei2@studio.unibo.it, giacomo.bertuccioli@studio.unibo.it, fabio.notaro2@studio.unibo.it
- description → as agreed with professor Omicini, the project aims to explore the features of competitive agents that consume energy through a simulation of a domestic fish aquarium with the following features:
  1. the agents in our simulation are represented by fish
  2. the environment in our simulation is represented by the aquarium itself, which contains obstacles to the movement of fish (stones) → the environment will be represented by a 2D space, in which all the entities that contains (fish and stones) move and are arranged
  3. the food is generated in a random position on the surface of the water and falls vertically downwards → the fish compete to reach it but the features of the individual fish affect this task (large fish are slower and need more food to satiate themselves, vice versa small fish are faster and need less food to satiate themselves)
  4. the goal of the fish is to survive as long as possible and feed themselves while avoiding obstacles
  5. a fundamental feature that we intend to explore is the fact that the fish, in order to carry out their actions, consume energy and earn it by feeding, therefore it will be necessary to define some strategies so that the agent also takes into account its energy when deciding whether to pursue or abort a goal, i.e. whether or not to follow food → the fish will therefore have to compete with each other to achieve their goals

- tecnhical aspects → the following technologies will be used for the development of the project:

    1. Java as the programming language of the overall system
    2. Jason to program the agents/fish
    3. Java Swing for the graphical interface

- planned workflow → the project will start with a preliminary phase of studying the literature papers regarding the competition between agents and agents having the notion of energy and energy consumption → a system design phase is then planned, followed by implementation and testing → during the continuation of this entire workflow we will also write the report

- goals → put into practice the theoretical notions that have been presented to us on the concepts of agent and environment, explore the scientific literature and the practical implications on the subject of competing agents and agents that consume and manage their energy, merge in a single project the use of Java and Jason.

Chapter 2

IN-DEPTH OVERVIEW ABOUT THE LITERATURE ON COMPETITIVE AGENTS CONSTRAINED BY RESOURCE CONSUMPTION

This chapter is devoted to reporting the relevant aspects that have emerged from our study of the papers that make up the technical and scientific literature on:

- competing agents
- agents constrained by energy/resource consumption.

In our opinion it was essential to accompany the practical part of this project with this in-depth theoretical study so as to answer some preliminary doubts, more conceptual than practical, about these two topics that have been mentioned in the course but that in our opinion deserve a detailed preliminary investigation with respect to the practical implementation of the project.
So the following two sections that make up this chapter are devoted to the two macro-topics considered useful for the proper development of our project: competition between agents in a MAS and the proper development of resource/battery/energy constrained agents.

2.1 Competitive agents in MAS

2.1.1 Multi-Agent Systems: Cooperation and Competition[1]

This article presents an in-depth analysis about the possible relationships that can be created within a MAS, with a focus on the concepts of cooper-

ation, conflict and human-agent interaction.

Indeed, multi-agent systems (MAS) can be defined as sets of AI agents that interact, collaborate or compete to achieve a shared or singular goal.
These agents operate autonomously, eventually exchanging information and adapting to dynamic environments, making MAS potentially ideal for complex applications such as autonomous trading, logistics, and robotics.
Although prototype versions of MAS are still immature in various real-world settings and scenarios, this technology is destined to become part of our future.
It should be emphasized that currently definitive answers about agent-to-agent and agent-to-human cooperation are still at the theoretical and research stage.

The nature of cooperation of rational agents

Cooperation is conceptualized as the action of two or more individuals working together for a common goal or perceived benefit (whether common or singular).
This process inherently and inevitably entails a cost to the individual, but this is outweighed by the benefits received, whether immediate or posthumous, or by reciprocal benefits provided by other individuals in the future.
It is good to point out that cooperation and conflict can naturally emerge when they are in the rational interest of the individual or group: that is, a rational agent pursues incentives and intentions aligned solely with his or her own interest, incentives that are highly dependent on circumstances and the environment in which it operates.
In fact, agents, being devoid of morality, emotions, cultural understanding or genuine respect for others, being driven solely by utility or optimization functions, are vulnerable to the same factors that influence cooperation and conflict between biological beings or between nations, so these two phenomena of the world around us can be studied in order to understand in more detail the dynamics governing relationships between rational agents.

Regarding cooperation, the bulleted list below provides several examples that illustrate the dynamics between individual cost and collective or future benefit in wolf pack cooperation:

- hunting in the wolf pack → as easy to imagine, a lone wolf that catches prey keeps it for itself, while a pack has to share it → although the individual quota is lower in the herd, joint hunting maximizes the probability of herd survival, allowing more territory to be covered, expend less energy, recover lost resources more quickly, and bring down larger prey → it thus emerges that in cooperation the individual cost is outweighed by the collective benefit

- prey sharing among wolves $\rightarrow$ a wolf W_1 who gives up his share of prey for another wolf W_2 with puppies may be motivated by altruism if related (no expected return benefit) or reciprocity if unrelated (W_1 expects that W_2 reciprocate in the future) $\rightarrow$ it may then happen that a third wolf W_3 who witnesses the disposal of the prey may also offer his share to W_1 or W_2 for mutual reasons, also increasing the likelihood of receiving help from someone else in the future.

It is clear from this simple parallel with wolves that cooperation is more likely when group members anticipate future interactions and thus future benefits of much greater magnitude than if they were to be obtained individually immediately.
Conversely, if interactions are rare, the risk of conflict over obtaining a resource might be considered beneficial or even unavoidable by agents.

The role of competition in multi-agent systems

As for competition, on the other hand, it can arise for a variety of reasons, summarized in the following list through a parallel based on an example of conflict between nations (X and Y) competing for control of an indivisible third territory:

- first preemptive strike $\rightarrow$ X directly attacks Y fearing that a prolonged wait will yield the initiative to Y, forcing X to cede control of territory
- misunderstanding of relative strength $\rightarrow$ X demands control of territory, assuming Y lies about its own strength, however Y is stronger than expected and estimated by X and therefore attacks directly without threatening
- change in the balance of power $\rightarrow$ after years of peace with X, in control of disputed territory, Y's strength grows while X's decreases, so Y asks for a renegotiation, X refuses it and Y starts a war believing he is now strong enough to regain control of the desired territory
- perception of superiority and historical grievances $\rightarrow$ X perceives that Y has violated its sovereignty and launches a small-scale invasion not necessarily in an attempt to start total war, but to reassert its position and level the differences.

It is important to note that conflict is significantly more likely when the parties compete for control of something indivisible and of high interest to the parties: this particular scenario often prevents negotiation and imposes a dynamic whereby the winner gets everything (the benefit) and the loser loses everything (the benefit).

At this point, however, it should be stressed that it is the environment, and only it, that tends to determine which among cooperation or competition is in the best interest of the agents and not only what individual agents have to gain or lose in the short or long run.
Special attention must therefore be paid to environmental resources, their modeling and availability, in order to understand the nature of the agents who will exploit them:

- if resources are scarce, it is highly likely that agents favor and prefer self-preservation, selfishness and competition
- conversely, in the case of abundant resources, agents are more likely to agree to benefit everyone, preferring altruism, cooperation and collaboration.

Actually, however, this distinction is not so clear and especially not so automatic: sometimes even under conditions of scarcity, cooperative behaviors might emerge to facilitate risk distribution, sharing of scarce resources, mutual protection, mutual aid and division of labor or, on the other hand, if resources are abundant, an agent might still engage in cooperative behaviors for a variety of reasons.
The dynamics and relationships within the group can then be cyclical: in a cooperative environment with abundant resources, some individuals may begin to exploit the situation (for example by not contributing but still collecting their share of benefits), leading others to do the same, effectively depleting resources and potentially leading to conditions of conflict.
Despite all these possible variations, the one rule that emerges is always the same: cooperative or conflict strategies are selected by agents solely on the basis of the environmental pressures of the moment.

Strategies for fostering cooperation in multi-agent systems

The paper then proposes several strategies for promoting cooperation in MAS, both at the level of agent design and structuring the operating environment:

- incentivize alignment with humans $\rightarrow$ for this purpose it is necessary to design agents in such a way that they perceive humans as part of their group (and with a higher hierarchy than all other agents), it is then useful to reward agents to push them to cooperate with humans by creating appropriate rational incentives and finally it is good to divide agents into groups, which can also compete to achieve the goals defined by humans, and reward the best performing groups
- if competition for resources is unavoidable, efforts should be made to ensure that it involves only divisible resources, so as to still facilitate negotiating relationships

- properly manage uncooperative behavior → if punishment is necessary, it is preferable to entrust it to other agents designated as guarantors of cooperation, acting on behalf of the humans → it is good then also to design the environments and MAS so that they have features of hierarchy and meritocracy, with humans at the top and agents climbing the hierarchy based on the achievement of goals, gaining in the case more benefits and control over lower-level agents → if this is not possible, it is preferable to organize the agents in series or sequence toward a common goal (making sure they all have the same level), with only human supervision

- addressing and deterring emergent competition in dynamic environments → it would be best to limit the deployment of MAS to static environments so as to maintain consistent and visible cooperative incentives → where this is not possible, i.e., in cases where MAS is needed in highly dynamic environments, it is necessary to design and implement strategies to regularly monitor agents, the system and goals achieved, looking for behaviors that could undermine the already shaky equilibrium → in such a view it might serve to require agents to explain the rationale and motivations behind their behaviors and not employ agents with recursive self-improvement and self-replication capabilities in MAS, since competitive pressures might direct these capabilities toward gaining power rather than increasing cooperation

- finally, there are additional general strategies dictated by common sense → modeling environments where cooperation produces non-obvious and significantly greater rewards than conflict, training agents in altruism and shared accumulation where collective success determines the rewards of individuals, and finally introducing mechanisms to reward agents even if other agents achieve their goals.

In conclusion, certainly the paper proposes more strategies and advice related to agent cooperation (which deviates from the purely competitive nature of our project), however it is useful as it provides a fundamental theoretical framework on the dynamics of cooperation and conflict in MAS, emphasizing the crucial role of rational incentives and environmental pressures, indispensable factors that govern the nature of agent relationships and therefore deserve due attention in analysis, design, implementation and testing.

2.1.2 Introduction to AI safety, ethics and society[2]

Chapter 7 of this book contains detailed information about the challenges of collective actions in agent behavior.

Introduction

This chapter begins by analyzing how the collective behavior of a multi-agent system may not reflect the original interests of its component agents.
Agents, when they find themselves interacting with each other, may in fact produce conditions that none of them desire, even when they have similar goals and priorities.
Emblematic in this regard is the reported quote that is attributed to the american economist Thomas Schelling, according to which micro-motivations are not equivalent to macro-behavior.

The chapter introduction goes on to explore this idea with some examples from the real world:

- trees in a forest → each tree in a forest competes with its neighbors for sunlight, which is considered a limited resource → the only way for a tree to get more light is to grow taller than its neighbors, however growing taller requires a lot of energy which the tree could otherwise use for other purposes (such as fruit production or root growth) → if there is no way to enforce an agreement between the trees, each competes with its neighbor by growing unnecessarily taller and taller, and all pay the high costs in trying to become the tallest

- excessive working hours → people often work many more hours than they would like or ideally should for the sole purpose of competing for limited prestigious positions instead of devoting time to other interests or family → at least in theory, if everyone in a given field reduced their working hours by the same amount, they could free up time and increase their quality of life while maintaining their relative position in the competition, i.e. everyone would achieve the same work result and everyone would benefit from the freed up time, yet no one does, because if they were the only ones to reduce their work efforts they would be outpaced by others who do not.

These phenomena suggest that the risk of having a MAS with imperfect behavior cannot be eradicated simply by ensuring that each individual agent is loyal and pursues its goal to the best of its ability, as this alone does not guarantee optimal macrobehavior in line with expected goals.
In other words, intelligent agents may, while acting rationally and in accordance with their own interest and goal, collectively produce outcomes that none of them desire because of erroneous or poor interrelationships.
These risks can be differentiated and explored in the following sections, using conceptual models more comfortable to us, such as game theory and evolutionism.

Game theory

This section exploits game theory to explore how interactions among agents can create risks distinct from those generated by a single agent acting in isolation.
The reasoning that follows is based on the assumption that in these games the agents are:

- self-interested → agents make decisions with a view solely to their own utility, regardless of the negative or positive consequences for others
- rational → agents always act in an attempt to maximize their utility.

Well, as anticipated, individually rational and self-interested behaviors can produce collective outcomes that are suboptimal, or even catastrophic, for all involved.
To illustrate this dynamic, the chapter lays out the prisoner's dilemma, a canonical example of a paradox in game theory, which will serve to show how sometimes agents might choose not to cooperate even though they know it will lead to a collectively suboptimal outcome.

In the prisoner's dilemma, two agents must decide whether to cooperate or not.
However, the costs and benefits are structured in such a way that for each agent, betraying the other is the best strategy, regardless of what the other partner chooses to do.
The classic setup involves two suspects, Alice and Bob, who are arrested by the police in the act of trespassing (for which there is a light penalty) but suspected of intending to commit theft (for which there would be a far worse penalty) and interrogated separately from each other.
Therefore, the police offer each suspect the same deal:

- if only one confesses (and thus betrays the other) and the other remains silent (cooperates), the confessor is released (0 years) and the other receives a long sentence (8 years)
- if both confess (cheat on each other), both receive an average sentence
- if, on the other hand, no one confesses (both cooperate), both receive a short sentence (1 year).

Assuming that Alice and Bob are both rational and self-interested, thus interested only in minimizing their own prison sentence, well each suspect faces four possible outcomes, which depend not only on one's own decision (cooperate or cheat) but also on that of the partner.
The figure below illustrates the results from Alice's point of view:

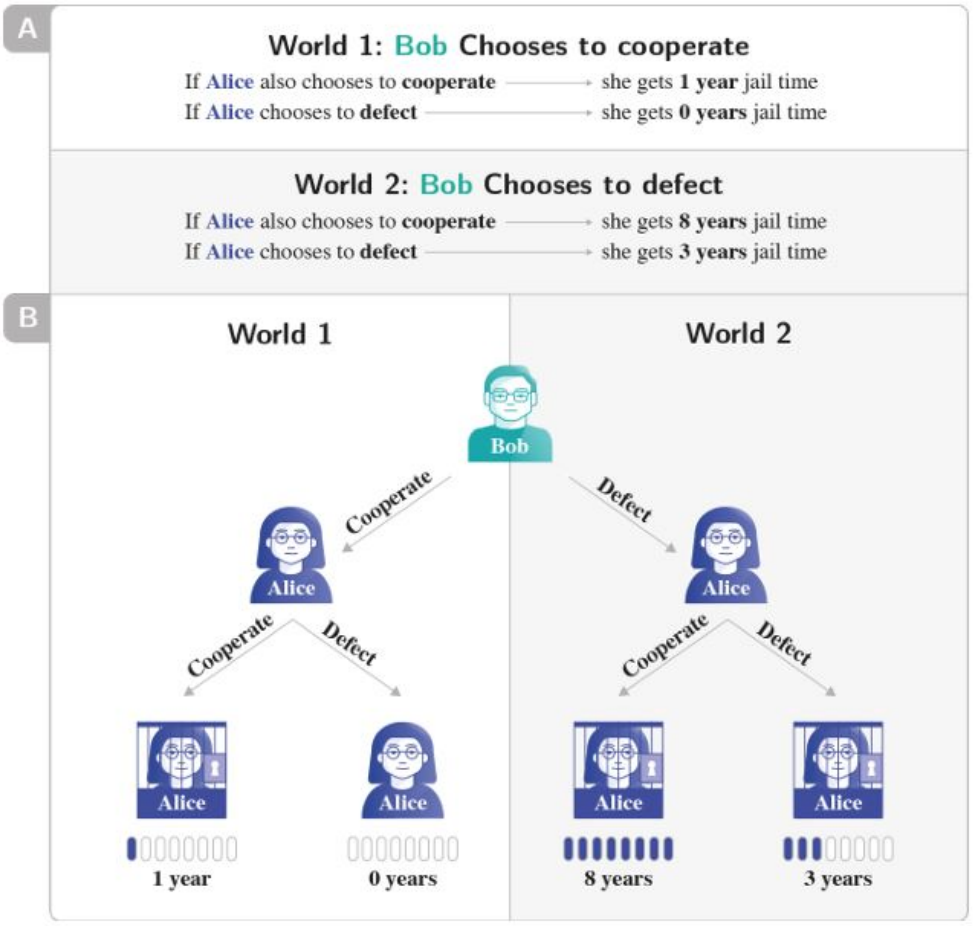


Figure 2.1: the possible outcomes for Alice in the prisoner's dilemma.

It becomes clear early on that betrayal, for both of them, will be the dominant and preferred strategy.
Alice in fact does not know what Bob will do and must decide whether to cooperate or betray.
Let us analyze his choice:

- if Bob cooperates, Alice by cheating on him would be released (0 years) instead of serving 1 year (if she also cooperates)
- if Bob instead cheated, Alice also cheating only receives a sentence of 3 years in prison instead of 8 (if she cooperated).

Alice therefore, interested only in minimizing her own punishment, aims to save herself as many years in prison as possible in both scenarios and therefore will likely choose to betray Bob: in fact, in each case she saves one year if her partner cooperates or five years if her partner defaults.
This is called the dominant strategy: a rational agent playing the prisoner's dilemma should choose to defect no matter what the partner does.
It is useful to reiterate that the decision is symmetrical for both suspects having both access to the same proposal by the police.
It is therefore possible to visualize the results for both in a payoff matrix, where the payoffs are negative because they represent years of imprisonment.

	Bob cooperates	Bob betrays
Alice cooperates	-1, -1	-8, 0
Alice betrays	0, -8	-3, -3

Table 2.1: payoff matrix for the prisoner's dilemma.

The stable equilibrium state in the prisoner's dilemma is therefore that both agents betray each other.
This is called a stable equilibrium state as neither is a situation in which no agent would choose to go back and change its decision (switch to cooperate) if it could not also alter the partner's behavior.
This outcome is also called, in a similar variant, a Nash equilibrium: a combination of strategies from which no agent can benefit by unilaterally choosing a different strategy.
Interacting with each other, rational agents having self-interested features will tend to choose strategies that are part of a Nash equilibrium.

However, and this is the focal point, as can be seen from the payoff matrix, there would exist an outcome, apparently voluntarily ignored by the agents, that is better for both: if both chose to cooperate, in fact, they would get a better sentence, each serving 2 years less than the outcome of

mutual betrayal and thus Nash equilibrium.
This mutual cooperation outcome is referred to in the literature as a Pareto improvement over the Nash Equilibrium.
The outcome of mutual betrayal is therefore Pareto inefficient, since it could be modified to improve the situation of some (both Alice and Bob) without making anyone's situation worse: the shift from mutual betrayal to mutual cooperation is the only Pareto improvement possible in this game.
Note that there are real-life examples that can be likened to the prisoner's dilemma.
Indeed, despite simplifications, such a dilemma can be a useful lens for understanding real social dynamics, as even in the world around us there are instances where rational, self-interested parties produce inefficient Pareto states:

- negative political campaigns → competing politicians often use negative campaign tactics → by using negative ads to attack opponents, they all end up with damaged reputations, whereas instead mutual cooperation would lead to all having better reputations (Paretian improvement), but this does not happen because again attack is the dominant strategy, since if an opponent does not use negative ads, a politician benefits by using them first and if the opponent does use them, using them reduces the difference → in this way however both converge on the Nash Equilibrium of mutual attack, with avoidable damage to all

- armaments race → it would be better for all governments to simultaneously reduce military budgets instead of increasing them → no nation would become more vulnerable than it already is if everyone did so, and each could redirect economic resources to more important areas such as education or health → instead we have been witnessing widespread military armaments races for years, because for any single nation to unilaterally reduce its military spending would be irrational compared to others, so in this case the dominant strategy is to opt for high military spending, which allows for a Nash equilibrium in which all nations must decrease spending in other valuable sectors, when it would be more Pareto efficient for everyone to have lower military spending, proportionately.

As easy to imagine, the only way to avoid pareto inefficient situations would be to promote and prefer cooperation among agents wherever possible.
Indeed, as it turns out, under certain conditions, rational and self-interested agents will converge on a Nash equilibrium of mutual betrayal, although both would be better off cooperating.
There are basically two strategies to push agents to cooperate:

- change the payoffs → by changing the values in the payoff matrix, we can more easily steer agents away from undesirable equilibria → in the prisoner's dilemma it would suffice to charge a cost in years to cooperate with the police and betray the partner, so that it is the mutual cooperation strategy that becomes a Nash equilibrium
- make agents more altruistic → each agent should also value and take into account the outcome for their partner in relation to the behaviors they choose to adopt → in the prisoner's dilemma it would suffice to model agents' payoffs so that they take into account not only their own benefits but also those of their partner.

The situation gets slightly more complicated but becomes extremely more interesting in the so-called iterated prisoner's dilemma.
In the real world, in fact, agents rarely interact only once.
The goal of this variant of the dilemma is to understand how cooperative behavior can be promoted and maintained as multiple agents interact over time multiple times, thus expecting frequent future interactions.
Indeed, as seen, the dominant strategy for a rational agent in a single interaction is to betray the partner despite the fact that this certainly does not lead to the absolute best outcome.
This is likely because a single interaction an agent cannot influence the partner's future actions, but in an iterated scenario, an agent's behavior in one round could influence the partner's response in the next.
Iterating and repeating the prisoner's dilemma in fact opens the door to rational cooperation: both agents can achieve higher payoffs by promoting a cooperative relationship than by betraying each other.
The expectation that their betrayal will be punished or vendicated and their rewarded cooperation in fact incentivizes both agents to cooperate.

It emerges that uncertainty about future engagement enables rational cooperation.
In the real world, an agent can rarely be sure that it will never interact with a given partner again, so where there is sufficient uncertainty about the future of their relationship, rational agents may be more cooperative, since noncooperative behavior may lead to less beneficial long-term outcomes.

In this case, however, so-called extortion strategies, in which an extortionist agent obtains higher payoffs to the partner by occasionally betraying him even when he cooperates, should be avoided.
Extortionists act by calculating the maximum number of betrayals they can make without nullifying the partner's motivation and trust to continue cooperating.
The partner will still have an incentive to acquiesce and cooperate, since deviating would lead to a lower payoff for him.

After discussing the simple prisoner's dilemma and the iterated prisoner's dilemma, we need to move on to analyze interactions involving more than two agents simultaneously.
A collective action problem is like an iterated prisoner's dilemma but at group level: when an individual interacts with multiple partners simultaneously, it may still converge on inefficient Nash Pareto equilibria.
Indeed, with more than two agents, cooperation may be even more difficult to ensure: if enough agents decide to betray, all will suffer the consequences.

In other words, in a collective action problem, each agent must choose between contributing to the common good at a personal cost or benefiting from the contribution of others at no personal cost: the latter is unfortunately once again the dominant strategy and is known as free riding.
For an agent, free riding increases its own personal benefit, regardless of whether others contribute or not: if an agent's contribution to the common good is small, not contributing does not significantly diminish the collective good and thus the decision to free ride has no significant negative consequences for the agent itself or for MAS.
The agent can then choose whether to get its share of the collective benefit by paying the cost of the contribution or saving the cost of the contribution.

The following table illustrates a collective action problem by summarizing a matrix of generic payoffs, where b denotes the collective benefit obtained if everyone cooperates and c denotes the individual cost of cooperation:

	The rest of the MAS contributes	The rest of the MAS does free riding
Contribute	$b - c$	$-c$
Free riding	b	0

Table 2.2: as noted, assuming the individual contribution is small compared to the collective benefit, free riding is always the seemingly best strategy for an individual.

It is clear from the table that $b - c < b$ e $-c < 0$.
It is also clear from the table how for every agent to do free riding is apparently always better.

Once again, in order to avoid such an undesirable situation, action must be taken on payoffs: indeed, it is possible to increase the probability of cooperation by generating incentives that lower the individual cost of cooperation and increase the individual cost of defection.
Examples of real-world collective action problems include climate change, depletion of fisheries resources and, more generally, all common pool resource problems.

The latter set of problems illustrates how in some cases agents can cause catastrophes while acting rationally and in their own interest, with perfect knowledge of the impending disaster and despite the ability to prevent it.

Cooperation

After exploring game theory, the chapter focuses on cooperation as a means of addressing the collective action problems previously explored.
This emerged in the argument as a central theme because ensuring morally positive and desirable outcomes without cooperation can be extremely difficult, even for intelligent and rational agents.

The key point of the section dealing with cooperation concerns the five mechanisms through which cooperation can emerge or be maintained in multi-agent systems:

- direct reciprocity → cooperation can emerge when one agent performs a favor for another expecting that favor to be returned in the future (this clearly can happen only if the possibility of repeated and future interactions between agents is admitted) → in this regard the greater the probability of meeting again, the greater the incentive to cooperate in the present → cooperation can evolve through direct reciprocity only when the probability w of successive encounters between the same two individuals is greater than the cost-benefit ratio of the useful act (c/b) →

  Table 2.3: payoff matrix for direct reciprocity games.

	Cooperation	Betrayal
Cooperation	$b - c/(1 - w)$	$-c$
Betrayal	b	0

  → significant real-world examples of this phenomenon include cooperative hunting strategies among dolphins and mutual fur cleaning among chimpanzees

- indirect reciprocity → in this case cooperation is based on reputational mechanisms where one agent may decide to cooperate with another considering the latter's reputation → a good reputation (having been generous previously) encourages new cooperation, while a bad reputation (having been selfish previously) discourages it → in this context agents before acting must carefully consider and evaluate the effect and consequences of each of their actions on their own reputation → this mechanism is particularly useful in large groups, where it is less likely to encounter the same partner again → unlike the previous case,

cooperation can emerge through indirect reciprocity only when the probability q that one agent can recognize another agent's reputation exceeds the cost-benefit ratio of the useful act →

Table 2.4: payoff matrix for reciprocity games.

	Recognize	Betray
Recognize	$b - c$	$-c(1 - q)$
Betray	$b(1 - q)$	0

→ significant real-world examples include cleaner fish and client fish (clients prefer cleaners with a good reputation for feeding on parasites rather than mucus)

- group selection → in intergroup competitions, groups with more members are more likely to prevail over those with fewer members → selection at the group level therefore also influences selection at the individual level and more numerous cooperative groups are more adept at coordinating resource allocation and maintaining communication, making them less likely to extinction → with m groups and maximum group size n, group selection promotes cooperation when $b/c > 1+n/m$ →

Table 2.5: payoff matrix for group selection games.

	Cooperation	Betrayal
Cooperation	$(n + m)(b + c)$	$n(-c) + m(b - c)$
Betrayal	nb	0

→ significant real-world examples here include lethal intergroup conflicts between chimpanzees

- kinship selection → agents are more likely to cooperate with others with whom they share a higher degree of kinship → kinship selection favors cooperation according to Hamilton's rule, which states that an agent will help a relative only when the benefit to the relative b multiplied by the kinship between the two r exceeds the cost to the agent c, i.e. $rb > c$, or $r > c/b$

- institutions → finally, cooperation can be facilitated by intentionally designed large-scale structures that are publicly accepted and recognized as authoritative → examples include governments and international organizations in the real world such as the UN → in such a situation institutions aim to establish collective goals that require cooperation, often generating cooperative incentives or sanctions for nonparticipation.

Although they are rather abstract and general, it is important to note that promoting cooperation through these mechanisms can still lead to new risks, such as nepotism, group favoritism or extortion, so a deep understanding of the benefits and risks is essential too when considering promoting cooperation in MAS.

Conflict

Chapter 7 then goes on exploring the nature of agent conflict in MAS.
Contrary to intuition, although conflict is often costly for all parties involved, it can still sometimes be a rational choice for intelligent agents.
The main objective of this section is to understand the dynamics by which, despite the benefits given by cooperation, a group of agents may instead enter a state of conflict.
Conflict, in this context, is broadly defined as the decision to betray, that is, to act in one's own interest at the expense of cooperation.
To analyze why rational agents might choose conflict instead of reaching a peaceful agreement, we use the framework of bargaining theory, which describes the reasons why rational agents might fail to reach a bargaining agreement and instead resort to destructive conflict with undesirable consequences.

Agents, driven by pressure to outmaneuver rivals or preserve power and resources, sometimes prefer conflict, especially when they cannot reliably predict the outcomes of their own or others' actions.
A central concept in bargaining theory is the bargaining range, which is the set of possible outcomes that both parties in a competition find acceptable through negotiation.
For example, in a lawsuit in which victory for the client implies a settlement of $40,000 with legal costs of $10,000 for both parties and a probability of success of 60% for the client, the expected value from the conflict for the client is 0.6 * $40,000 - $10,000 = $14,000.
The bargaining range for the client starts at $14,000: he will not accept counteroffers that would lead to lesser outcomes as it would rather suit him to pursue the lawsuit.
Symmetrically, if the expected value of not paying (if the owner wins) is 0, and the legal costs are $10,000, the owner will accept a settlement up to a value that makes negotiation preferable to conflict.
If losing the case costs $40,000 and the legal costs are $10,000, the total cost of the conflict to the owner (in case of defeat) is $50,000.
With a probability of defeat of 60%, the expected cost is 0.6 * $40,000 + $10,000 = $34,000.
The bargaining range for the owner goes up to $34,000.
Thus, an x agreement is acceptable to both if $14,000 $\leq$ x $\leq$ $34,000.

If the parties' bargaining ranges do not overlap, a peaceful agreement is unlikely.

Assuming that agents act rationally in pursuit of their goals, they too may encounter conflict scenarios and choose to pursue violent conflict rather than peaceful and diplomatic/negotiated resolution for the reasons explored above.

Sources identify several factors that make it more difficult to reach negotiated agreements or avoid confrontation, leading agents to opt for conflict:

- power shift → if a party anticipates that a rival will become significantly stronger in the future, it might rationally choose to engage in conflict in the present time, when it is still in a position of relative strength or lesser weakness → significant historical examples include the Suez crisis of 1956, where the declining powers of the time (Britain and France) reacted militarily against rising Egypt, which had nationalized the Suez Canal, to safeguard their influence over the territory → conceptually, in this case it emerges that an expected shift in power in the future can shift the bargaining range of one party, potentially eliminating overlap with the other party's range and making conflict the preferable option in the present
- first strike advantage → if the odds of victory are significantly higher for the side that attacks first, this reduces the attractiveness of the negotiation and increases the incentive for conflict → real-world examples include patents or military aggression such as Pearl Harbor
- indivisibility of the issue → peaceful negotiations assume that what is being bargained over is divisible in a way that allows for mutually acceptable compromises, but when issues are indivisible, it is difficult or impossible to reach agreement within the bargaining range → examples of this include sovereignty over disputed territory, priority for an organ transplant or child custody in a divorce
- unintentional misinformation → lack of accurate and shared information leads to mutual distrust and a failure of negotiations → if each agent believes he or she is the stronger party, both may demand more than half of the disputed value, leading to non-overlapping bargaining ranges and thus conflict
- deliberate misinformation → competitive situations may incentivize agents to mislead others or to falsify the truth, increasing the likelihood of conflict or seeking to gain undue advantage during the negotiation phase → examples of this include employers hiding the salary range, candidates exaggerating their skills, salespeople hiding product flaws.

However, it should be specified that not all grounds for conflict are encapsulated in the models of bargaining theory.
For example, there are two factors that are highly predictive of conflict:

- inequality
- scarcity of resources.

Developmental pressures

The final section of the chapter explores the dynamics expected in a future characterized by a large population of agents.
The analysis uses evolutionary theory to understand the impact of competitive pressures on larger time scales within large groups of interacting agents.

In fact, the concept of Generalized Darwinism postulates that Darwinian mechanisms are applicable and useful to explain phenomena even outside of mere biology.
This includes non-biological systems such as culture, academia, industry and, pertinently, the development of AI and AI agents.
Successful, well-made software, for example, can be seen as an entity that propagates more effectively, with adaptive features (such as ease of use, reliability) that are imitated and incorporated into subsequent versions, leading to the proliferation of the most suitable products in terms of propagation.
However, it is crucial to note that generalized darwinism does not imply that evolution produces good outcomes in a value sense: what propagates best is simply what propagates best, not necessarily what is beneficial in other senses.

To formalize Generalized Darwinism, the necessary and sufficient conditions for evolution by natural selection formulated by evolutionary biologist Richard Lewontin are used:

- change $\rightarrow$ there is variation in traits among individuals
- retention $\rightarrow$ future iterations of individuals tend to resemble previous iterations
- differential fitness $\rightarrow$ different variants have different propagation rates.

In AI development, variation is present (there are many different patterns and architectures), retention occurs through adaptation of existing patterns or imitation of successful features, and differential fitness also occurs as some patterns or traits are propagated more than others (for example more effective or popular patterns).
Therefore, AI development meets these criteria and is subject to the same

evolutionary pressures as biological populations.

This section goes on to discuss at length the reasoning and parallels between AI agents and biological systems, but the conclusion of all the considerations made is that agent development is likely subject to evolutionary forces that, by default, tend to promote selfish and undesirable behaviors.

2.1.3 Cooperative and Competitive Multi-Agent Systems: From Optimization to Games[3]

Introduction

MAS, as they are known, can address complex problems through interactions between autonomous agents, improving efficiency and robustness in application areas such as intelligent transportation and smart grids.

In the context of MAS, agent interactions can manifest themselves in cooperative, competitive or hybrid forms.
Cooperative or competitive dynamics emerge as a function of factors such as cost functions, information shared among neighbors and environmental influence.
This paper analyzes such behaviors, also taking advantage of it to explore topics such as distributed optimization and game theory, summarized here only briefly since it was covered extensively in the summary of the previous paper.

From the perspective of cooperative optimization, agents work together to disseminate information and converge toward an optimal overall solution.
However, in real-world scenarios characterized by heterogeneity, uncertainty, and conflicting individual goals, modeling even non-cooperative behaviors is critical and useful.
In this context, game theory provides tools for formalizing agent interactions in cooperative and non-cooperative games, both static and dynamic.
Special attention is paid to dynamic non-cooperative games, which model competitive situations in partially observable environments of high strategic complexity.
Concrete examples include differential pursuit-evasion games and capture-the-flag games, in which agents are incentivized to maximize their individual payoff rather than collective welfare.

Cooperative optimization

The paper begins by explaining in detail the concept of ccoperative optimization, which is briefly reported in this section.
Cooperative optimization consists of the joint minimization of an overall cost

function by autonomous agents, often with decentralized structures.
In such contexts, the objective function is:

$$f_t(x) = \frac{1}{n} \sum_{i=1}^{n} f_{i,t}(x)$$

where $f_{i,t}(x)$ is the local cost function of agent i at time t and $x \in R^d$ is the joint decision to be optimized.

Game theory

As explained in the summary of the previous paper, game theory provides a mathematical formalism for modeling strategic interactions between autonomous agents in a multi-agent system.
The goal is to analyze optimal decisions in contexts in which the outcomes of one agent's choices depend on the choices of others.

Games are generally divided into two macro-categories: cooperative games, characterized by collaborations, and non-cooperative games, characterized by competition and conflict.

As for cooperative games, they are positive-sum games in which participants can form alliances or coalitions to achieve a common goal, maximizing collective utility.
Typical examples include collaborative planning in robotics, coordination in traffic control or energy management.
They can be further subdivided into:

- static → agents make decisions one at a time
- dynamic → agent decisions are made sequentially, over time and therefore the actions of one agent can influence the future state of the environment and the decisions of other agents.

As for non-cooperative games, on the other hand, they model situations in which each agent acts selfishly, aiming to maximize its own utility or minimize its own costs, with no possibility of binding agreements.
The reference equilibrium is the so-called Nash equilibrium (NE), which is a set of strategies from which no agent has an interest in deviating unilaterally.
The formal and mathematical definition of Nash equilibrium is as follows:

$$\forall \pi_i \in \Pi_i, \forall s \in S, \quad U_i(\pi_i^*, \pi_{-i}^*) \geq U_i(\pi_i, \pi_{-i}^*), \quad \forall i \in \{1, \ldots, n\} \tag{2.1}$$

where π^* is the joint strategy in equilibrium, Π_i is the strategy space of agent i, U_i is the expected payoff and π_{-i}^* denotes the strategies of other agents.
Non-cooperative games can also be divided into static and dynamic, but

there is in addition to this a further distinction, as there are also non-cooperative games:

- at full information → each agent knows exactly the features, strategies and payoffs of the others → a classic example is the prisoner's dilemma (also covered extensively in the summary of the previous paper), which falls precisely within the static non-cooperative full-information games
- or incomplete information → in which case an agent does not have complete knowledge of the preferences, strategies, features or payoffs of others.

The paper, in addition to these distinctions, points out that in contexts of large-scale dynamic non-cooperative games (i.e., having a large number of player agents), it is unrealistic for each agent to have complete or otherwise thorough knowledge of all other agents in the MAS.
To address this limitation, approaches known as mean field games are employed that allow an approximate solution of the Nash equilibrium by considering the population mean effect.

2.1.4 Collaborative vs Competitive Multi-Agent Interaction Paradigms[4]

Introduction

Multi-Agent Systems, as we know, can be classified according to the nature of interactions between agents: collaborative or competitive.
Understanding the dynamics emerging from these interactions and their differences is crucial to the design of effective and reliable MAS.

Definition of collaborative MAS.

In collaborative contexts, agents act with the shared goal of maximizing the collective utility of the system.
These systems require coordination, communication and sometimes joint planning.
In such a paradigm it is usual for agents to freely share knowledge and strategies to facilitate the achievement of common goals.
This approach reflects the cooperative nature of the dynamics among agents.

Definition of competitive MAS.

In contrast, in a competitive environment, agents operate independently pursuing potentially conflicting goals.
In such environments, on the other hand, it is usual for agents to maintain secret their own private reasoning and strategies.

This intentional isolation of decision making is necessary to preserve strategic advantages.
Consequently, communication is limited or even absent, and transparency among agents is minimized.
The competition also introduces the need to manage overlapping interests or conflicts among agents, which is why competitive MAS, especially those operating in resource-scarce conditions, require strict conflict resolution mechanisms.

Goal alignment and incentive structures

- In collaborative MAS, agents are incentivized to achieve shared goals → the entire system wins or loses as a single entity and rewards are often distributed proportionally based on overall results → examples include collaborative robotics in logistics where overall productivity is optimized
- In competitive MAS, on the other hand, each agent maximizes its own utility, even at the expense of others → is the case with automated trading systems, where each bot aims for maximum individual profit.

Evaluation metrics, as mentioned, must reflect this distinction: in collaborative systems, collective utility is to be measured, while in competitive ones, individual effectiveness is to be measured.

Pattern of communication and information sharing

- In collaborative MAS, communication is generally open and frequent to foster coordination and synchronization among agents → agents share crucial information to make collective decisions in the hope of reaching the common goal together
- In competitive MAS, on the other hand, information is considered strategic resources and therefore it is usual for agents to keep their reasoning and strategies private → this leads to selective communication protocols, where agents share nothing or only what is strictly necessary and certainly still functional to their goals.

Allocation and use of resources

- In collaborative systems resource management is done through joint planning or centralized allocation, aiming for overall optimization
- In competitive systems decentralized mechanisms such as auctions prevail instead, where each agent bids according to its expected utility.

Again, the metrics vary but are always based on efficiency and fairness in collaborative MAS and individual satisfaction in competitive MAS.

Decision-making mechanisms

- In collaborative settings consensus protocols, voting or hierarchical structures are adopted
- In competitive environments, on the other hand, decisions are autonomous, based on local information and strategic considerations, so conflict resolution mechanisms typically need to be in place.

Models of failure and robustness

It should be reiterated that:

- in collaborative MAS the metrics assess overall utility, task completion and coordination efficiency → the focus must therefore be on collective emergent behavior
- while in competitive MAS the assessment is based on concepts such as Nash equilibrium, strategic advantage and individual utility maximization.

2.2 Resource-constrained agents

2.2.1 Energy Trade-Offs in Resource-Constrained Multi-Agent Systems[5]

This paper presents a study of energy trade-offs in multi-agent systems characterized by resource constraints, such as sensor networks.

In such open systems, functionality can be highly affected by resource availability.

The paper specifically investigates adaptive algorithms for computational networks with energy challenges, studying how self-organization can be used to exchange energy for acceptable accuracy to improve longevity in a sensor network and how perceived threat and information sensitivity can be used to exchange energy for (acceptable) security risk in an ad hoc network.

Two main areas are investigated, analyzing their respective trade-offs:

- the trade-off between energy and accuracy in sensor networks
- the trade-off between energy and security in ad hoc networks.

In sensor networks, there is a trade-off between accuracy and longevity for networks that implement in-network data aggregation functions.
Therefore, the following system model is defined:

- a set of sensor nodes $S = s_1, \dots s_n, s_{sink}$, where s_{sink} is the sink node
- each sensor node s_i, has a certain energy at time t $en_i(t)$ and a fixed error value er_i
- the sensor nodes take readings such that $r_i(t) = R(t) \pm er_i$, where $R(t)$ is the correct value and the error er_i is applied positively or negatively at random
- the sensor nodes form views such that $o_i(t) = O(\{r_1(t), r_2(t), \dots, r_n(t)\})$, where $O(\dots)$ is an aggregating function of the readings received from the neighbors
- the network error is defined as $NE(t) = |R(t) - o_{sink}(t))|$
- the network lifetime is defined as the time t such that $\sum_{i=0}^{n} en_i(t) > 0$ and $\sum_{i=0}^{n} en_i(t+1) \leq 0$, where the sums do not include the sink node.

The requirement is to derive a method for aggregation that allows sensors to specify an upper limit of accuracy for the network error (ϵ) such that the lifetime of the network increases as this upper limit increases.
In other words, the goal is to derive an algorithm for $O(r_1(t), r_2(t), \dots, r_n(t)\})$ such that it maximizes NL ensuring that $\forall t[NE(t) \leq \epsilon]$.

As for ad hoc networks, on the other hand, there is a trade-off between risk and longevity in them for heterogeneous networks implementing different security policies, since some policies require computationally more complex and thus energy-intensive encryption algorithms.
In this other case, the following system model is defined:

- a set of agents $A = \{a_1, \dots, a_n\}$
- a set of clusters C such that for each cluster $c_i \in C$, $c_i \subseteq A$
- a pair of roles, cluster-head and member, such that there is one cluster-head per cluster and each agent is a member of at least one cluster
- the set of social constraints (normative rules) as given, for example by a specification in an action language
- environment variables, specifically at security level (sl), at energy level (el) and at perceived threat level (tl, derived from the sensitivity of information and the detected interception rate).

The requirement is to derive a security policy that allows agents to determine a network-level security policy based on local, cluster-level interactions regarding available power and perceived threat.
Defined such a system, an algorithm was developed to select an appropriate security policy, focusing on data aggregation by cluster-level gossiping and normed role-based voting protocols.

We also use agent opinion training and dissemination to determine the need for change at the cluster level.
This is followed by a vote (of cluster members) for cluster-level change.

The adaptive aspect here lies in the ability of agents to dynamically select a security policy, which clearly impacts energy consumption, based on the level of available energy (el) and perceived threat (tl), thus realizing a trade-off between energy and security risk.
Both in sensor networks and in ad hoc networks, the environment is thus characterized by resource constraints that limit the effective computation of the constituent nodes.
To properly handle such constraints, algorithms have been developed based on the interaction and adaptation of an underlying social network with a manifest organizational structure.

The fundamental importance of adaptability in this context lies in the ability of the proposed algorithms to allow nodes/agents to dynamically change their computational and communication behavior (and consequently their energy consumption) in response to resource constraints and environmental or system factors.
Specifically:

- in sensor networks the algorithm allows sacrificing accuracy (accepting greater error) in order to conserve energy and extend the life and durability of the network
- in ad hoc networks the algorithm instead allows exchanging energy for security $\rightarrow$ agents can invest more energy (using more robust security policies) when the perceived threat or sensitivity of the information requires it or conserve energy (accepting greater risk) when resources are scarce or the threat is low.

Thus, strategic adaptability based on conscious trade-offs is a promising avenue for ensuring the functionality and longevity of multi-agent systems in resource-constrained environments.

2.2.2 Multi-Agent Coverage Control with Energy Depletion and Repletion[6]

Cooperative multi-agent systems are widely employed to perform tasks such as coverage, surveillance, monitoring and patrolling of a given mission space. A typical coverage problem involves deploying agents to cooperatively maximize coverage of a designated area.

In most existing coverage control frameworks agents are assumed to have unlimited power: however, in practice, battery-powered agents, such as commercial drones, have limited autonomy (only about 15 minutes of flight time for many drones).
This paper therefore introduces the dimension of energy constraints to the traditional coverage problem.

The basic setup is similar to that presented in previous works, where agents interact with the mission space through their sensing and perception capabilities.
Unlike other energy-aware multi-agent algorithms that aim to reduce energy costs, this work assumes the availability of a charging station that agents can visit according to a specific policy, which is better defined below.
The goal is to maximize an overall measure of environment coverage by controlling the cooperative movement of all agents while ensuring that no agent runs out of energy while in the mission space.
In other words, it must be guaranteed that $q_i(t) > 0$ for all t in the time interval $[0, T]$, where $q_i(t)$ is the state of battery charge of agent i, unless the agent is precisely at the charging station (assumed to be in the origin (0,0)).
The energy consumption is modeled as proportional to a quadratic function of the agent's speed, while charging at the station occurs at a constant rate $\beta > 0$.
Another constraint is that only one agent at a time can be served by the charging station.

To address this problem the behavior of an agent is modeled through a hybrid system.
This model captures the controlled switching of agents between cover (when energy is consumed) and recharge (when energy is restored) modes.
The feasibility of the problem is ensured by exploiting guard functions on the battery level of each agent to prevent energy depletion on the way to the charging station.
The charging station must then act as a centralized manager to resolve the order and contention among the agents competing for the one charging resource.

As mentioned, the behavior of each agent is described by a hybrid system model that includes three different modes of operation:

- mode 1 (coverage) → in this mode the agent focuses on the coverage task by moving at the maximum allowed speed v and its direction follows the direction of the gradient of the coverage objective function $H(s)$, with s representing the position of all agents → during coverage the agent's battery is discharged with a dynamics approximated by $\dot{q}_i(t) = -\alpha v^2$, where α is a scaling constant and v is the speed (maximum in this mode) → the state of charge $q_i(t)$ decreases monotonically

- mode 2 (traveling to-charging) → when the energy level of the agent decreases below a certain threshold the agent switches to mode 2 to head toward the charging station → in this mode the direction of the agent is fixed and points toward the charging station (assumed at the origin), while the speed $v_i(t)$ is determined by a scheduling algorithm managed by the charging station (such as FRFS or SDF) to resolve contention between agents requiring charging at the same time → the battery continues to discharge with dynamics $\dot{q}_i(t) = -\alpha v_i^2(t)$

- mode 3 (in-charging) → the agent remains stationary at the charging station ($\dot{x}_i(t) = 0, \dot{y}_i(t) = 0$) and its battery is recharged according to the dynamics $\dot{q}_i(t) = \beta$, where β is the charging rate.

Such a hybrid system provides one cycle per agent between these three modes and transitions between modes are controlled by specific guard functions triggered when certain conditions occur:

- transition from mode 1 to mode 2 → this transition occurs for agent i when its energy level $q_i(t)$ drops to a value that ensures that the agent can still reach the charging station (assumed at the origin) from its current position $s_i(t)$ traveling at maximum speed → the associated guard function is
$$g_i(s_i, q_i) = q_i(t) - \alpha v s_i(t)$$
→ the transition occurs when $g_i(s_i, q_i) = 0$, that is, when $q_i(t) = \alpha v s_i(t)$, where $\alpha v \|s_i(t)\|$ represents the minimum energy required for agent i to reach the charging station at the maximum speed v from location $s_i(t)$, considering the consumption rate αv^2

- transition from mode 2 to mode 3 → an agent in mode 2 transitions to mode 3 when it reaches the charging station → the associated guard function is
$$g_i(s_i) = s_i(t)$$
and occurs when $g_i(s_i) = 0$, i.e. when the agent's distance from the origin (charging station) is zero

- transition from mode 3 to mode 1 $\rightarrow$ a mode 3 agent transitions to mode 1 when its battery has reached a sufficient charge level $\theta_i \in (0, 1]$ $\rightarrow$ the associated guard function is

  $$g_i(q_i) = \theta_i - q_i(t)$$

  and occurs when $g_i(q_i) = 0$, that is, when the state of charge $q_i(t)$ reaches the threshold value θ_i.

2.2.3 Representing the acquisition and use of energy by individuals in agent-based models of animal populations[7]

This paper focuses on modeling individual energy budgets within agent-based models (ABMs) for animal populations.
ABMs are essential tools for predicting population responses to environmental changes, particularly variability in food availability.

The paper highlights the need to equip individuals (agents) with their own energy budgets and proposes a modeling framework based on physiological ecology.
The rules of energy allocation under different conditions, the scaling laws governing metabolic processes and the efficiency with which energy is acquired from food are detailed.
The techniques for parameterizing such complex models are also discussed.
The objective of the paper is to provide a minimum and agreed specification for the energy representation of individuals in ABMs.

Introduction

Agent-Based Models (ABMs), also often called individual-based models, are widely used to predict how populations respond to changing environments.

Food availability is a critical environmental factor for animals and varies in space and time, so to realistically model population responses, individuals (agents) should have their own energy budgets.
Currently, there is no universal consensus in the scientific community on how these energy budgets should be modeled in ABMs.

Basic elements of an energy budget model

The paper begins by proposing that a modeled animal eats as needed to cover at least the energy needs for maintenance, growth and reproduction.
The acquired resources are then allocated between maintenance, growth, reproduction and energy storage.
The allocation rule is a crucial aspect in determining how the acquired energy is distributed among the different vital functions.

The following priority allocation mechanisms can be defined, i.e. if the ingested energy exceeds the minimum requirement (for example only for maintenance), the animal allocates energy according to a fixed priority order defined in the bulleted list below:

- maintenance → basic energy requirements for survival
- growth → for young individuals excess energy after maintenance is allocated to growth
- reproduction → once they reach a certain size and/or age and after covering maintenance and growth needs, energy is allocated to reproduction
- energy storage → any energy remaining after meeting maintenance, growth and reproduction needs is stored as an energy reserve and this storage stops once a predefined optimum level is reached.

So, if the ingested energy is not sufficient to cover all priority needs (maintenance, growth, reproduction), the animal is forced to draw from its energy reserves.
At this stage, the allocation priorities remain the same but reserves can only be used until they reach a critical threshold, beyond which the animal enters a mode known as emergency/survival: when energy reserves also fall below a critical threshold the model proposes that all available energy (from the limited intake and remaining reserves) be allocated exclusively to maintenance. This means that allocation toward growth and reproduction is completely suspended as the top priority becomes the immediate survival of the individual.

The animal dies if reserves are completely depleted and maintenance can no longer be supported by any means.

It should also be noted that the maximum rates at which an animal can ingest food and allocate energy to various processes depend on its body mass and environmental/body temperature.

Scaling laws

Scaling laws describe the quantitative relationships between an organism's properties and its size.
In the bioenergetic context these laws are critical because many processes, such as food acquisition, maintenance, growth and reproduction, scale in predictable ways with body mass and environmental temperature.

The key concept is metabolic rate (B), which represents the total energy

consumption of an organism.
It has been shown that basal or resting metabolic rate (BMR) scales with body mass (M) and temperature (T).
The Metabolic Theory of Ecology (MTE) provides a theoretical and formal basis for these relationships, expressed by the equation:

$$B = B_0 M^c e^{-E/jT}$$

where:

- B_0 is a normalization constant
- M is the body mass
- c is the allometric exponent, often of the value close to 3/4 (Kleiber's Law)
- T is the temperature in Kelvin
- E is an activation energy (about 0.65 eV)
- j is Boltzmann's constant.

For warm-blooded (homeothermic) animals, the temperature-dependent exponential term is less relevant because body temperature is relatively constant.

The enormous importance of this relationship lies in the fact that it describes how all energy-demanding processes (foraging, maintenance, growth, reproduction) scale similarly with body mass and temperature, being related to metabolic rate.
The maximum ingestion rates also follow relationships dependent on body mass and temperature.

Assimilation efficiency

Assimilation efficiency is a key parameter that determines how much of the ingested energy is actually available to the animal.
Not all of the energy contained in ingested food is in fact utilized: a portion is lost as fecal and excretory waste.

In this sense assimilation efficiency is defined as the proportion of energy gained through digestion to the total energy ingested from the food.
It:

- is a ratio, so it is often expressed as a percentage
- depends significantly on the type of diet

- averages around 50-60% among species
- does not vary significantly with the body mass of the animal
- varies widely depending on food → high for meat and seeds (over 80%), moderate for young vegetation (40-70%) and low for mature vegetation or wood.

This parameter is crucial for calculating the energy actually available for allocation after the agent has ingested the food in the model.

Final considerations

The integration of individual energy budgets into ABMs is crucial for realistic modeling of populations in environments with varying food availability. It allows for explicit modeling of key processes such as starvation mortality and allocation of resources to reproduction as a function of local environmental conditions.

In spite of progress, areas of uncertainty and need for additional research remain, particularly regarding the precise rules of energy allocation under suboptimal resource conditions.
The minimal model discussed in the paper serves as a basic framework on which to add complexity to better represent specific real systems, balancing biological fidelity with computational complexity.

The models proposed by the paper, while simple, have practical applications in various fields, including conservation planning, assessment of environmental impacts (for example construction projects) and risk assessment of chemicals on non-target organisms.

Chapter 3

ANALYSIS

3.1 Requirements

The present section contains functional requirements (describe what the software does) and non-functional requirements (describe what features the software has).

3.1.1 Functional requirements

The numbered list below shows what the software system should do:

1. to simulate realistic agent/fish behavior in an aquarium representing the environment
    - 1.1 the fish will have to compete to eat the available food
    - 1.2 the fish will have to have characteristics that influence their actions → fish with high weight will have higher range but will consume more energy to move and need more food to satiate themselves, while fish with low weight will be faster in their movements and they will need little food to satiate but will also have lower range
    - 1.3 the fish/agents will need to act intelligently → such intelligent behavior includes avoiding obstacles realistically (no abrupt detours), avoiding other fish and avoiding the edges of the aquarium
    - 1.4 the food will need to be able to drop in two ways → either periodically or by means of a dedicated button that the user can press at will
2. the system should allow the user to preliminarily choose some important parameters of the simulation, such as the number of fish/agents, the amount of food and obstacles

3. the system should display a simple graphical interface that reports interesting elements of the simulation and also present buttons to control its progress and display preferences

    3.1 the GUI should include a section devoted to statistics and a section devoted to relevant events that occurred during the course of the simulation

    3.2 the buttons that make up the GUI should allow for terminating the simulation, pausing it and feeding the fish

    3.3 the GUI should provide the user with the ability to switch between two viewing options → there should be a simple view that shows only the fish moving between obstacles eating food and a more in-depth view that also shows some of the fish parameters such as energy, range, and direction.

3.1.2 Non-functional requirements

In the numbered list below are the non-functional features that the realized system must possess:

4. configurability → the system should be able to be adapted to different needs or operational contexts without changing the source code, but automatically only through settings or parameters

5. comprehensibility → the structure, code and software documentation should be clear and easy to understand for those who need to read, modify or maintain them → this should apply as much to the code as to the operation of the application

6. fluidity of animations and responsiveness → the graphical movements of the simulation are continuous responsive and without jerks or slowdowns, providing a pleasant and natural visual experience

7. testability → the system shall allow for simple, accurate and efficient verification of the correct functioning of its components through tests

8. documentability → the software shall be clearly and completely described through technical documentation explaining its structure, operation, use and maintenance.

Chapter 4

Design

4.1 Architecture

The realized software system possesses a monolithic architecture: this option was chosen for simplicity, since focusing on system architecture does not fall among the aspects of particular relevance addressed during the course.
In more detail, the system consists of the following main components:

- folder asl → contains the file fish.asl which encapsulates the modeling of our agents/fish, i.e. the beliefs, goals, percepts and plans specification
- folder env → encapsulates the Java code used to model the domain and environment of the simulation → in particular it contains the two packages model (dedicated to modeling all the domain entities involved) and view (which contains the code to create the two necessary GUIs, a preliminary one to set the simulation parameters and another dedicated to show the progress of the actual simulation)
- folder launcher → encloses the code needed to comply with functional requirement 2, that is to give the user the ability to set some preliminary parameters of the simulation, such as the number of agents and the amount of food and obstacles
- folder utils → contains the functions used by the agent that, due to their complexity, were written in Java rather than in Jason
- test → folder that encloses all the unit tests performed.

4.2 Detailed design

In the presented section, the UML diagram including the most prominent classes and interfaces of the proposed solution is depicted and described in detail.

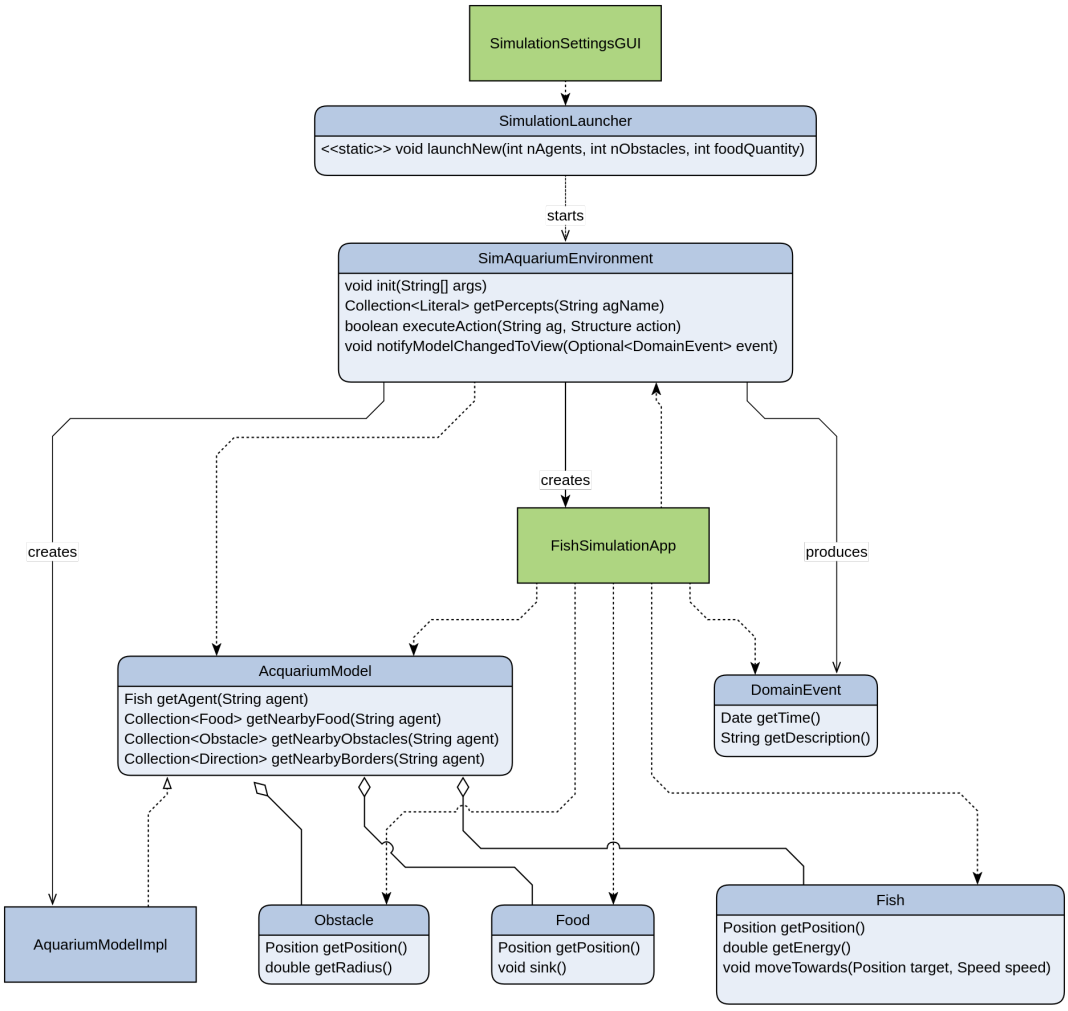


Figure 4.1: UML schema containing the main components of the realized system.

The bulleted list below describes the components of the depicted UML schema in more detail:

- SimulationSettingsGUI → is the preliminary GUI through which it is possible to set the parameters of the simulation, such as number of agents, amount of food and amount of obstacles → once all the parameters are decided, such GUI starts the MAS with the specified number of agents (to better understand how this is done, we recommend to consult the dedicated section later)
- SimulationLauncher → custom version of the RunLocalMas library class which concretely starts the MAS (and therefore also the environment) with the parameters set thanks to the SimulationSettingsGUI GUI
- SimAquariumEnvironment → class representing the environment within which the fish/agents act
- AquariumModel → interface representing the aquarium model and therefore containing all references to the fish/agents to the food and obstacles present
- AquariumModelImpl → is the class that implements the AquariumModel interface
- Obstacle, Food and Fish are the most important domain classes of the system
- FishSimulationApp → is the GUI that actually shows the progress, statistics and events of the simulation
- DomainEvent → class that represents the relevant domain events produced by the environment that will be represented in the GUI.

4.3 Agent modeling

This section explains in more detail how the agents/fish were designed, modeled and implemented.
First, the following table shows an in-depth overview about the beliefs, goals, percepts and plans that characterize our agents:

BELIEFS	**GOALS**	**PERCEPTS**	**PLANS**
weight(W) is the weight of the fish	init is the initial goal that the agent pursues	food(FS) returns which pieces of food are close enough to the fish, where FS is a list of items-food_elem(X, Y, ID)	init initializes the first belief and the fish itself inside the model
direction(X, Y) is the direction in which the fish is moving		close_to_food(F) is the percept of the nearest piece of food in range to be eaten	step is the plan for deciding the fish's next action, which may include stopping to digest, avoiding obstacles and edges or deciding on the new speed of movement
energy(E, MaxE) indicates the current and maximum energy of the fish		obstacles(O) are the obstacles in the fish's range, where O is a list of items obstacle(X, Y, Range)	move is the actual plan for moving
has_target(X, Y) relative position of the fish target, i.e. the food it intends to reach		borders(B) includes borders that fall within the range of the fish, where B is a list of border(Top, Left, Bottom, Right)	eat is the plan to eat, which is triggered by the close_to_food percept

BELIEFS	GOALS	PERCEPTS	PLANS
food_energy(FE) is the energy that eating a piece of food gives the fish		paused is the percept alerting the fish to stop	
steps(S) is the number of steps before changing direction when the fish is not following any target			
digesting indicates that the fish is digesting after eating, so it will be still for a few moments			

Instead, the .asl file designed and created to model the behavior of the agents/fish is shown below:

```
// fish.asl
// Beliefs.
speed(normal).
steps(1).
direction(1, 0).
-has_target(_, _) <-
    -+steps(30).

// Goals.
!init.

// Percepts.
+food(Coordinates) <-
    utils.find_nearest(Coordinates). // Eventualmente aggiunge la
        belief target_food(X, Y)

+close_to_food(F) : energy(E, _) & E > 0 <-
    !eat(F).

// Plans.
+!init <-
```

```
    utils.agent_init;
    .belief(weight(W));
    .belief(energy(E, ME));
    init(W, E, ME);
    !step.

+!step : energy(E, _) & E <= 0 <-
    .wait(not(paused));
    .drop_all_intentions;
    .drop_all_desires;
    die;
    utils.stop_agent.

+!step : energy(E, ME) & steps(S) <-
    if (digesting) {
        .wait(1000);
        -digesting;
    }
    .wait(not(paused));
    if(has_target(_, _)){
        utils.rotate_dir;
    }
    utils.check_aquarium_borders;
    if(obstacles(O)){
        utils.avoid_obstacles(O);
    }
    if(not(has_target(_, _))){
        if(E >= ME / 2){
            !move(normal);
        } else {
            !move(slow);
        }
        if (S - 1 = 0) {
            utils.set_random_dir;
            -+steps(30);
        } else {
            -+steps(S - 1);
        }
    } else {
        if(E >= ME / 2){
            !move(fast);
        } else {
            !move(faster);
        }
    }
    !step.

+!move(Speed) : direction(X, Y) <-
    -+speed(Speed);
```

```
    utils.move_towards(Speed);
    move_towards(X, Y, Speed).

+!eat(F) : energy(E, ME) & food_energy(FE) <-
    .wait(not(digesting));
    eat(F);
    -+energy(E + FE, ME);
    -close_to_food(F);
    +digesting.

-!eat(F) <- .print("I was not able to eat ", F).
```

In addition to the asl code given above, it was also necessary to implement some utility functions in Java that perform actions otherwise too complex to accomplish in Jason.
Such utility functions are:

- agent_init → initializes some key beliefs of the agent
- avoid_obstacles → tries to avoid collisions between the agent and obstacles, also based on the position of any food present in the agent's range
- check_aquarium_borders → tries to avoid collisions between the agent and the aquarium's edges
- find_nearest → chooses the target from the pieces of food closest to the agent
- move_towards → moves the agent in a certain direction, modifying its energy accordingly based on its speed and weight
- rotate_dir → updates the direction correctly so that the agent can point toward its target
- set_random_dir → sets a random direction for the agent, useful for when he does not have any food in the range
- stop_agent terminates the agent
- Utils → contains a set of various and generic utility methods or constants, for example to switch from literal to position or to switch from term to direction, etc.

Instead, the following is a UML state diagram that is used to understand in more detail the agent's main behavior as a function of the states and events it encounters:

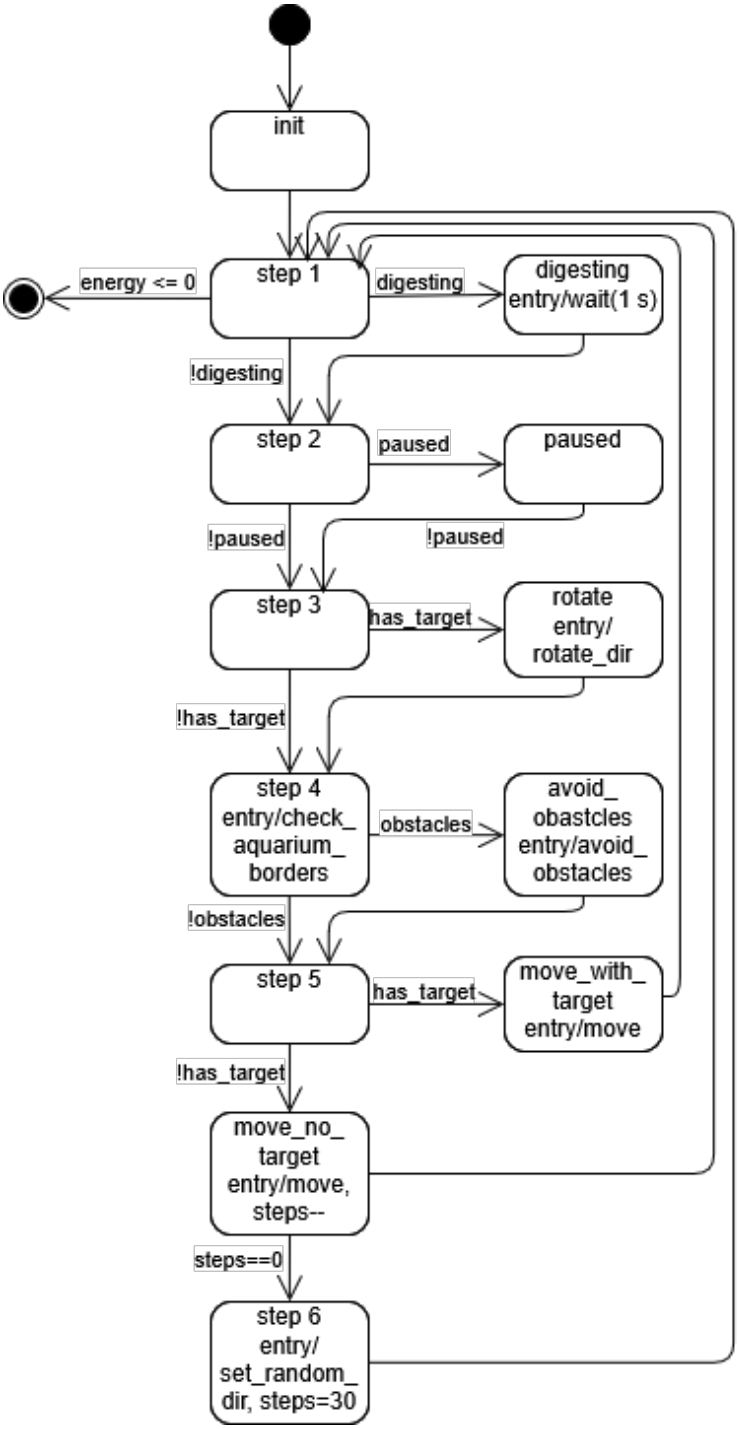


Figure 4.2: UML state diagram of the agent's main behavior.

Instead, the following UML state diagrams focus more on the agent's behavior as a function of the percepts it detects:

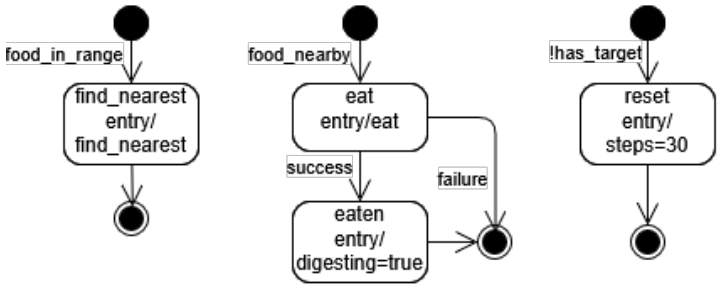


Figure 4.3: UML state diagrams of agent behavior as functions of percepts.

4.4 Dynamic number of agents

Functional requirement 2 required giving the user the ability to preliminarily set an arbitrary number of agents.
The following steps had to be carried out to meet this requirement:

- provision of an initial GUI for selecting simulation parameters and thus also the number of agents
- introduction of our custom SimulationLauncher class, which extends RunLocalMas to generate the appropriate .mas2j file based on user preferences →

```
// SimulationLauncher.java
public class SimulationLauncher extends RunLocalMAS{
    static final String NEW_FILE_NAME = Path.of(".", "tmp",
        "sim.mas2j").toString();
    static final String MAS_CONTENT = "MAS robots {\r\n" + //
            "\tinfrastructure: Centralised \r\n" + //
            "\tenvironment: env.SimAquariumEnvironment(%s,
                %s) \r\n" + //
            "\tagents: fish #%d;\r\n" + //
            "\taslSourcePath: \"src/main/asl\";\r\n" + //
            "}";
    static SimulationLauncher mas;

    public static SimulationLauncher launchNew(int
        numberOfAgents, String foodQuantity, String
        numberOfObstacles) throws IOException, JasonException{
        String result = String.format(MAS_CONTENT,
            foodQuantity, numberOfObstacles, numberOfAgents);
        File f = new File(NEW_FILE_NAME);
```

```
        f.getParentFile().mkdirs();
        f.createNewFile();

        try (FileWriter writer = new FileWriter(f)) {
            writer.write(result);
        } catch (IOException e) {
            e.printStackTrace();
        }

        mas = new SimulationLauncher();
        mas.init(new String[]{NEW_FILE_NAME});
        mas.registerMBean();
        mas.registerInRMI();
        mas.registerWebMindInspector();
        mas.create();
        mas.start();
        return mas;
    }
}
```

- from the .mas2j file generated by the SimulationLauncher creation and startup of the MAS in a regular way.

Chapter 5

Development

5.1 Automated testing

Two different types of tests were set up to verify the correct operation of the application:

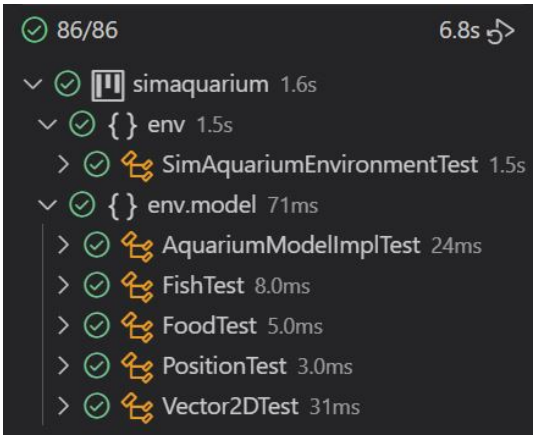
- unit test → automated tests that verify the correct operation of a narrow unit of code (such as a method or class) in isolation from the other components of the system
- logging test → to verify in a simple way the correct behavior of agents, tests have been set up that verify the correct ordering of events triggered by agent actions.

5.1.1 Unit testing

The JUnit framework was leveraged to verify the correctness of the code of the following classes:

- AquariumModelImpl
- Fish
- Food
- Position
- Vector2D
- SimAquariumEnvironment.

As can be seen from the following image, all 86 prepared tests are correctly passed:



5.1.2 Logging testing

To verify the correctness of each agent's behavior, a list called events was created within the AquariumModelImpl class that contains pairs of type <String, String> where the first element represents an agent's id and the second element represents a string reporting which event it caused.
In more detail, new pairs are added to the list whenever an agent performs an action that triggers a relevant domain event.
For example:

- the pair (id, "add") is inserted whenever a given agent/fish is initialized
- the pair (id, "food_percept") is inserted whenever a given agent/fish detects food in its range
- the pair (id, "eat") is entered whenever a given agent/fish eats food
- the pair (id, "digest") is entered whenever a given agent/fish stops its movement for a few moments to digest
- the pair (id, "die") is entered when a given agent/fish dies.

At the end of the simulation, it is therefore possible to verify the correctness of the behaviors of all agents by observing the order of occurrence of these events, in particular by checking that, for each agent:

- "add" event is not preceded by any other event → in Linear Temporal Logic: $\neg$ event U add
- each "eat" event is preceded by a "food_percept" event → in LTL: $\neg$ eat U food_percept
- each "eat" event be followed by a "digest" event → in LTL:$\Box$(eat → $\Diamond$ digest)

- no event follows the “die” event.

These checks are done at the natural end of the simulation, so they will not be done if the user presses the Stop button to prematurely end the simulation but will instead be done if the user presses the Stop button and there are no fish/agents still alive.
After pressing the Stop button, the user will be informed of the outcome of the event testing via a message box like the one shown in the following figure, possibly highlighting in detail which sorting rule was not followed:

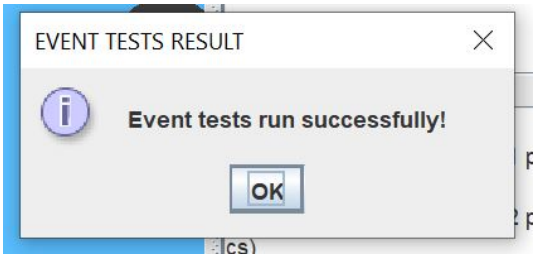


Figure 5.1: message box in case of successful event testing.

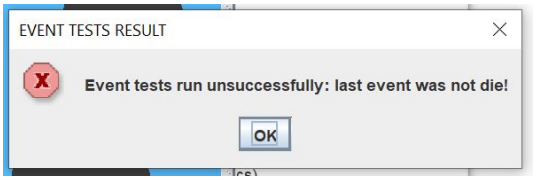


Figure 5.2: message box in case of negative outcome of event testing.

Clearly, it is expected that if the simulation ends normally then event testing will also be passed successfully.

5.2 Links to the papers

This section reports the concepts that we found relevant from each paper and that we were able to carry over, implement, model and include in the practical project as well:

- *Multi-agent Systems: Cooperation and Competition[1]* → "If multi-agent systems are deployed in fluctuating environments, individual agents and the system as a whole must be regularly monitored for emergent objectives that may undermine cooperation and favor irregular or non-fair conflict." → in our project, therefore, an evaluation metric known as the fairness index was set up, which is visible at all

times thanks to a field affixed in the GUI and which reports through a value ranging between 0 and 1 how fair the competition is →

$$\text{Fairness Index} = e^{-\frac{v}{3a}}$$

where a is the maximum amount of food eaten by a single agent and

$$v = \text{variance} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

where n is the total number of agents, x_i is the amount of food eaten by the i-th agent, and $\bar{x}$ is the average amount of food eaten by agents

- *Introduction to AI safety, ethics and society[2]* → "Reducing competitive pressures could foster collective action...By easing the competitive pressures, we might be able to foster collective action to avoid driving up the collective risk level." → to ensure a more balanced competition, fish are modeled to have weight-dependent advantages and disadvantages (fish with a higher weight are slower, require more food to satiate and consume more energy to move but have a more ample perception range, conversely, fish with a lower weight have a narrower perception range but are faster, more agile, require less food to satiate and consume less energy to move)

- *Cooperative and Competitive Multi-Agent Systems: From Optimization to Games[3]* → this paper explains how game theory is composed by a variety of different types of games and one of them is the dynamic incomplete-information non-cooperative games, i.e. competitive games in which the agents have a sequential order, and the later participant can observe the actions chosen by the former participant (in contrast, in static games all agents act together always at the same instant) and in which each agent has no information or at least only partial/incomplete information about the others → in the project when we had to choose how to model the fish we also chose to model them as if the competition was an incomplete information dynamic competitive game, i.e. a fish knwos nothing about another fish except what it perceives (whether it perceives another fish in its range)

- *Collaborative vs Competitive Multi-Agent Interaction Paradigms[4]* → "Key features include strategic autonomy where agents decide based on their own goals, often keeping their reasoning private" → this can be considered as a precondition of the previous point, that is, we also decided to model fish in such a way that they keep their strategies, goals, reasoning and percepts completely private, that there is no sharing of their own information at all

- *Collaborative vs Competitive Multi-Agent Interaction Paradigms[4]* → "Collaborative systems emphasize joint performance metrics for AI like global utility maximization, task completion, and coordination efficiency ... Competitive systems evaluate success through metrics like Nash equilibrium achievement, strategic advantage and individual utility maximization" → for this reason, we made our fairness index, the only competitive evaluation metric, dependent on the amount of food eaten by each agent individually

- *Energy Trade-Offs in Resource-Constrained Multi-Agent Systems[5]* → "...functionality may be compromised by a lack of resources...we investigate adaptive algorithms for power-challenged computing networks." → for this reason, fish move more or less quickly, that is they adapt their behavior based on certain thresholds on the energy they have available and the food they perceive to be in their range, taking risks to eat or avoiding them if they sense that the risk would be too great compared to the benefit

- *Multi-Agent Coverage Control with Energy Depletion and Repletion[6]* "We provide a solution to the above problem by modeling the behavior of an agent through three different modes: coverage (Mode 1), to-charging (Mode 2), and in-charging (Mode 3)...As an agent's energy is depleted, the agent switches to Mode 2 according to a guard function designed to guarantee that a minimum energy amount is preserved to reach the charging station from its current location while traveling at maximum speed." → although the paper proposes 3 different modes whose switching from one to the other is governed by the guards' functions, similarly we preferred to equip the fish with 4 fish speeds dependent on the agent's energy and the presence or absence of food in the agent's range and whose switching is handled not in the Jason guards but in the traditional Java part

- *Multi-Agent Coverage Control with Energy Depletion and Repletion[6]* "The dissipation of energy is proportional to a quadratic function of the velocity..." → similarly we made the energy consumed by an agent depend yes on its velocity, but also on its weight

- *Representing the acquisition and use of energy by individuals in agent-based models of animal populations[7]* → the paper exposes the so called scaling laws, "many similarities among species in the forms of energy budgets and life histories, and in the ways each of these scales with body mass and temperature...food acquisition, maintenance, growth and reproduction all require energy, and all scale in similar ways with body mass and body temperature" → for this reason in the previous

point we made the energy consumed dependent not only on the velocity but also on the weight of the agent

- *Representing the acquisition and use of energy by individuals in agent-based models of animal populations[7]* → "After ingestion, food is processed by the digestive system and only a proportion becomes available for allocation to the various functions...This proportion is called assimilation efficiency, defined as

$$\text{Assimilation efficiency} = \frac{\text{energy obtained by digestion}}{\text{energy ingested as food}}$$

" → for simplicity we have decided to set the assimilation efficiency to 100%, i.e. the totality of the food found becomes consumable energy, however it should be reiterated that to make the acquisition of energy more realistic we have created each agent with a different maximum energy and a different energy consumption rate dependent on its weight, exactly as in reality.

Chapter 6

Final comments

6.1 Self-evaluation and future developments

The development team is extremely satisfied with the work done and the chosen project.
The aquarium simulation provided a challenging and flexible environment, which allowed us to explore in detail both the theoretical aspects related to the literature on MAS, competitive and resource-constrained agents, and the practical aspects of modeling and implementation using Jason and Java.
During development, we were able to:

- experiment with BDI programming and the integration of agent-oriented (Jason) and object-oriented (Java) code
- reflect on the management of energy resources by agents, an aspect often overlooked in simpler simulations
- delve into the competitive dynamics between agents within an artificial but realistic ecosystem
- model intelligent behaviors in an environment with obstacles and scarce resources, stimulating the design of rational strategies for survival
- design and implement an effective and comprehensive GUI, enhanced by advanced visualization, pausing and logging features.

However, there also emerged room for improvement and new ideas to be explored.
In fact, the following points represent the main future developments that we think are interesting to explore:

- greater evolution and complexity of fish state → currently fish do not change their state over time, so it would be interesting to introduce some growth mechanisms (weight, maximum energy, speed)

- improvement of movement algorithm → currently movement is reactive but simplified, it would therefore be interesting to explore more complex algorithms, for example through predictive pathfinding, reinforcement learning or more advanced dynamic avoidance
- introduction of predators and preys → currently fish are all on the same competitive level, so introducing predators and preys would lead to more pronounced role differentiation and more articulated hunting dynamics, escape and cooperation/competition
- learning or adaptation mechanisms → for example agents modifying their behavior based on past experiences, such as avoiding certain areas or changing energy priorities
- advanced energy management → considering even passive consumption over time could make the simulation even more faithful and realistic
- GUI expansion
- data persistence → save detailed statistics of each simulation for retrospective analysis
- support for inter-group competition → instead of just individual agents, explore competition between groups or species.

The project, despite its apparent simplicity, has proved extremely fertile in terms of the possibilities for extension and deepening, offering insights for further future work in both teaching and research.

6.2 Difficulties encountered

In the course of project development, no blocking difficulties or obstacles emerged that would significantly slow down the progress of the work.
The development team has always maintained good continuity, managing to progress according to the established plans.
However, it is possible to point out some minor critical issues that emerged along the course of the project:

- scarcity of documentation and practical examples → the Jason technology and, more generally, the BDI paradigm present often synthetic or fragmented documentation → it has proven non-trivial to find complete and exhaustive examples on similar projects, especially in contexts where agents are constrained by resources (such as energy) or involved in competitive dynamics
- dynamism and parameterization of the number of agents → another difficulty concerned the possibility of setting up a dynamic and variable

number of agents → Jason does not in fact natively provide simple management of agents created at runtime based on external parameters

- finding relevant literature → even though the scientific corpus on MAS is extensive, it was complicated to locate specific papers focused on the aspects we were most interested in, i.e. competing rational agents with an explicit model of energy consumption and management.

Despite these minor difficulties, all issues were successfully addressed through good division of work, constant communication among group members and the incremental approach taken.

Appendix A

USER GUIDE

This appendix explains in more detail how to start and command the simulation.
To start the simulation, a task has been set up on Gradle, which is executable in two different modes:

- either by placing yourself in the root of the project and running the command gradle run runAquariumMas
- or by running the same command leveraging the IDE of your choice.

Once the system is started, an initial GUI will appear, dedicated to entering the simulation parameters, namely number of agents, amount of obstacles and amount of food:

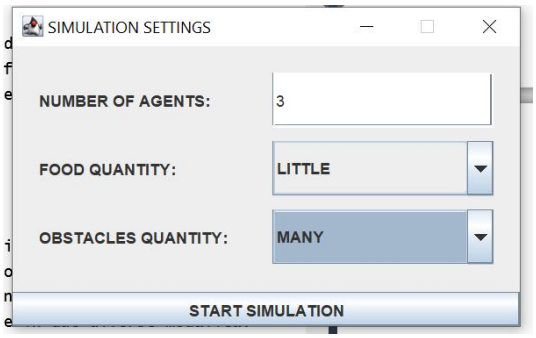


Figure A.1: preliminary GUI for simulation setup.

Once the necessary parameters are entered and selected and the button to start the simulation is pressed, the GUI dedicated to the actual progress of the simulation will appear:

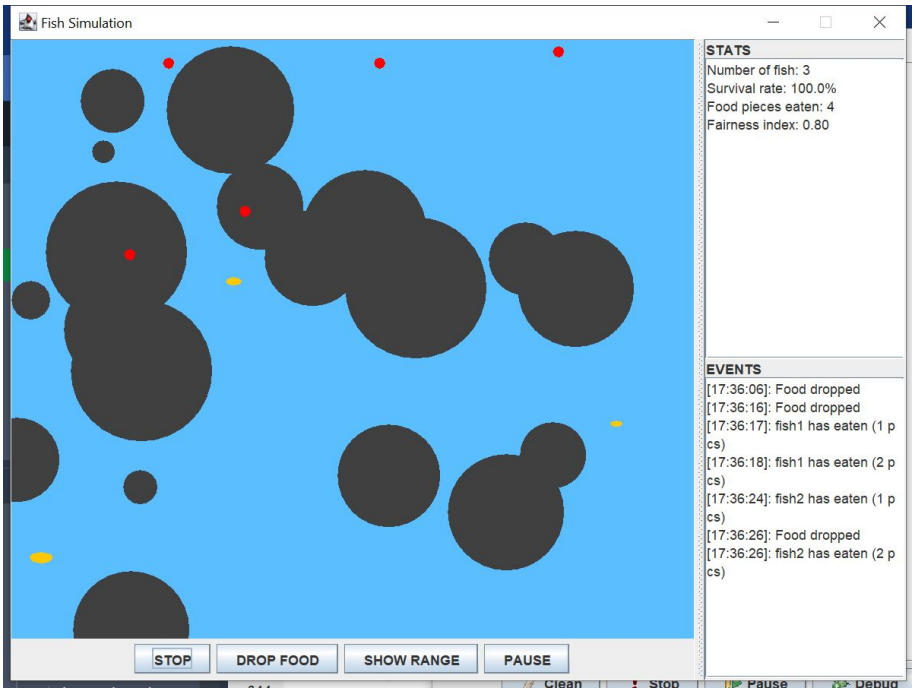


Figure A.2: GUI dedicated to the actual simulation.

As noted, the GUI is divided into three main areas:

- there is a large panel showing the progress of the simulation in real time, including the movement of agents and the fall of food
- on the right there are two lists, one below the other, dedicated respectively to relevant statistics (number of agents still alive, survival rate, total number of food pieces eaten and fairness index) and to relevant events that happened with the corresponding time (one agent fed or one agent died)
- finally at the bottom there is a section containing all the buttons reserved for controlling the simulation → the Stop button terminates the simulation (and if it is pressed after all the agents have died it will show the outcome of the event testing before closing the window), the Drop Food button forces the dropping of food, the Show Range/ Hide Range button graphically shows or hides some additional details for each agent, such as direction, range and residual energy, finally the Pause button is used to momentarily stop the simulation →

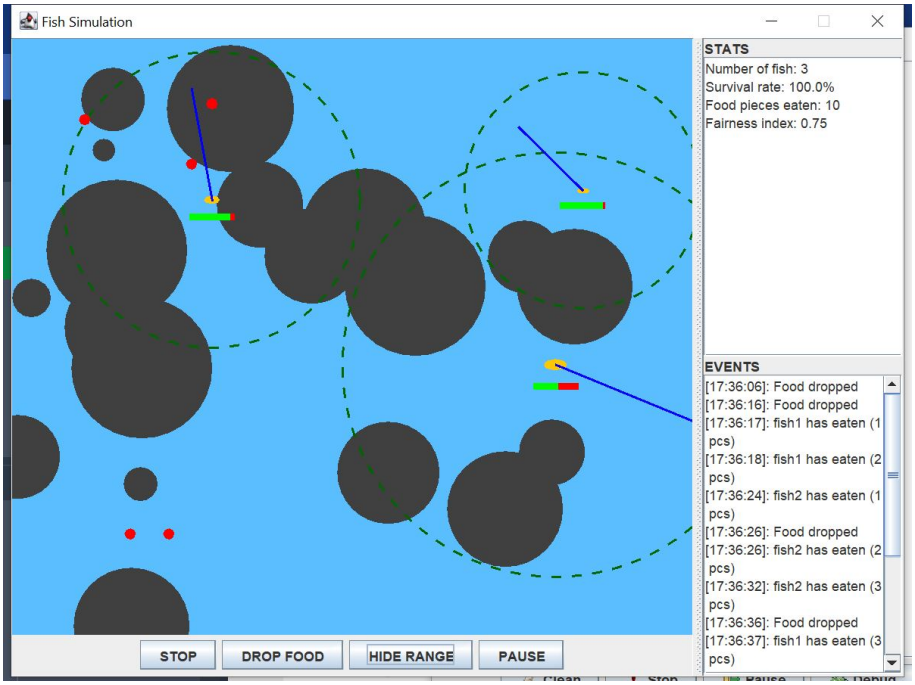


Figure A.3: GUI as a result of pressing the Show Range button.

Bibliography

[1] *Multi-agent Systems: Cooperation and Competition*, LUMENOVA, 2024

[2] Dan Hendrycks, *Introduction to AI safety, ethics and society*, Taylor and Francis, 2024

[3] Jianrui Wang, Yitian Hong, Jiali Wang, Jiapeng Xu, Yang Tang, *Cooperative and Competitive Multi-Agent Systems: From Optimization to Games*, IEEE, 2022

[4] Conor Bronsdon, *Collaborative vs Competitive Multi-Agent Interaction Paradigms*, Galileo, 2025

[5] Hugo Carr, Jeremy Pitt, Anthony Kleerekoper, and David Blancke, *Energy Trade-Offs in Resource-Constrained Multi-Agent Systems*, 2009

[6] Xiangyu Meng, Arian Houshmand, Christos G. Cassandras, *Multi-Agent Coverage Control with Energy Depletion and Repletion*, 2018

[7] Richard Sibly, Volker Grimm, Benjamin Martin, Alice Johnston, Katarzyna Kulakowska, Christopher Topping, Peter Calow, Jacob Nabe-Nielsen, Pernille Thorbek, Donald DeAngelis, *Representing the acquisition and use of energy by individuals in agent-based models of animal populations*, British Ecological Society, 2013.