INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Kuramoto Oscillators Bandits for Sequential Partner Selection in Rhythmic Social Interactions

João Pedro dos Santos Florindo de Almeida Lemos

Master in Computer Engineering

Supervisor:
PhD Pedro Figueiredo Santana, Associate Professor,
Iscte – Instituto Universitário de Lisboa

Supervisor:
PhD Pedro Lopes da Silva Mariano, Auxiliar Investigator,
Iscte – Instituto Universitário de Lisboa

November, 2025

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

Kuramoto Oscillators Bandits for Sequential Partner Selection in Rhythmic Social Interactions

João Pedro dos Santos Florindo de Almeida Lemos

Master in Computer Engineering

Supervisor:
PhD Pedro Figueiredo Santana, Associate Professor,
Iscte – Instituto Universitário de Lisboa

Supervisor:
PhD Pedro Lopes da Silva Mariano, Auxiliar Investigator,
Iscte – Instituto Universitário de Lisboa

November, 2025

I would like to thank my parents for their unwavering support and wisdom, my partner for her love and constant motivation, and my close friends, who were always there to help me get back on my feet.

Acknowledgment

My deepest gratitude goes to my supervisor, Professor Pedro Figueiredo Santana, for his invaluable guidance, encouragement, and continuous support throughout the development of this dissertation. His insightful comments and critical perspective were fundamental to refining my ideas and ensuring the scientific rigor of this work.

I am equally thankful to my supervisor, Professor Pedro Mariano, for his constant availability, constructive feedback, and encouragement, which were essential in overcoming the many challenges encountered during this research journey.

Sincere appreciation is extended to the Department of Information Sciences and Technologies at Iscte – University Institute of Lisbon, for providing the academic environment and resources necessary for the successful completion of this work.

I am also grateful to all the professors who have guided and inspired me throughout my academic life, from the very first years of school to the final stage of my university studies. Their dedication and passion for teaching have been a lasting source of motivation.

A special word of thanks goes to my friend Gonçalo Botelho, who was always by my side throughout the Master's program, sharing challenges, completing projects, and offering his help and support in every situation. His friendship, companionship, and dedication were fundamental to the success of this journey.

I would also like to express my heartfelt gratitude to my Erasmus colleagues and friends, who made that experience truly unforgettable, enriching me not only academically but also personally and culturally.

Finally, I wish to thank the Iscte Students' Association (AEISCTE) for its vital role in promoting student life and fostering a participatory, supportive, and inspiring academic environment. My involvement with AEISCTE also enabled me to develop personal, social, and professional skills that I will undoubtedly carry with me throughout my life.

Resumo

A sincronização é um fenómeno fundamental em sistemas naturais e artificiais, presente em processos sociais, cognitivos e cooperativos. Esta dissertação aborda o problema da seleção de parceiros em interações sociais rítmicas, como na dança a pares ou em tarefas colaborativas, formulando-o como um processo de decisão sequencial no qual agentes autónomos escolhem parceiros com o objetivo de maximizar a sincronização.

Para tal, combina-se o modelo de osciladores de Kuramoto, que descreve as dinâmicas rítmicas intrínsecas de cada agente, com algoritmos de Multi-Armed Bandits, nomeadamente Upper Confidence Bound (UCB), Discounted Upper Confidence Bound (D-UCB) e EXP3. Esta integração permite que os agentes aprendam incrementalmente a partir de interações sucessivas, equilibrando exploração e aproveitamento, e adaptando-se tanto a ambientes estacionários como a ambientes sujeitos a variações dinâmicas.

Foi desenvolvido um motor de simulação que possibilita experiências sistemáticas em múltiplos cenários, incluindo diferentes topologias de rede, níveis de ruído e alterações estruturais no conjunto de agentes. Os resultados demonstram que modelos de seleção de parceiros baseados em osciladores constituem estratégias eficazes para promover a sincronização, revelando capacidade de adaptação a mudanças graduais ou súbitas no ambiente. A comparação entre algoritmos evidencia a vantagem do D-UCB em contextos dinâmicos e confirma a robustez do UCB em cenários estacionários.

Este trabalho contribui para a compreensão dos mecanismos de sincronização em sistemas multiagente e oferece perspetivas relevantes para áreas como interação humano-robô, coordenação colaborativa e desenvolvimento de agentes artificiais adaptativos.

Palavras-chave: Sistemas Multiagente, Aprendizagem por Reforço, Osciladores de Kuramoto, Multi-Armed Bandits, Seleção de Parceiros, Sincronização

Abstract

Synchronization is a fundamental phenomenon in natural and artificial systems, present in social, cognitive, and cooperative processes. This dissertation examines the problem of partner selection in rhythmic social interactions, such as paired dancing or collaborative tasks, formulating it as a sequential decision-making process in which autonomous agents choose partners with the goal of maximizing synchronization.

To this end, the Kuramoto oscillator model, which captures the intrinsic rhythmic dynamics of each agent, is combined with Multi-Armed Bandit algorithms, namely UCB, D-UCB, and EXP3. This integration enables agents to learn incrementally from successive interactions, balancing exploration and exploitation while adapting to both stationary environments and those subject to dynamic variations.

A simulation engine was developed to support systematic experimentation across multiple scenarios, including different network topologies, noise levels, and structural changes in the agent population. The results show that oscillator-based partner selection models constitute effective strategies for promoting synchronization, demonstrating the ability to adapt to gradual or sudden environmental changes. The comparison between algorithms highlights the advantage of D-UCB in dynamic contexts and confirms the robustness of UCB in stationary settings.

This work contributes to a deeper understanding of synchronization mechanisms in multi-agent systems and offers relevant perspectives for areas such as human–robot interaction, collaborative coordination, and the development of adaptive artificial agents.

Keywords: Multi-Agent Systems, Reinforcement Learning, Kuramoto Oscillators, Multi-Armed Bandits, Partner Selection, Synchronization

Contents

Acknowledgment iii

Resumo v

Abstract vii

List of Tables xiii

List of Figures xv

List of Acronyms xvii

Chapter 1. Introduction 1
1.1. Context 1
1.2. Research Questions 1
1.3. Goals 2
1.4. Planning 2
1.5. Research Methodology 3
1.6. Dissertation Structure 4
1.7. Contribution to Code Implementation 5
1.8. Dissemination 5

Chapter 2. State of the Art 7
2.1. Ad hoc Search 7
2.1.1. Rhythmic Interactions in Humans 7
2.1.2. Rhythmic Interactions in Artificial Agents 7
2.1.3. Agent-Based Modeling for Multi-Agent Coordination 8
2.1.4. Sequential Decision-Making and Reinforcement Learning 8
2.1.5. Kuramoto Oscillators and Biological Inspiration 10
2.1.6. Multi-Agent Reinforcement Learning (MARL) 10
2.1.7. Convergence Challenges in MARL 11
2.2. Systematic Literature Review 12
2.2.1. Keywords 12
2.2.2. Inclusion Criteria 12
2.2.3. Exclusion Criteria 13
2.2.4. Databases 13
2.2.5. Query Results 13
2.2.6. Literature Review 14

2.3. Summary and Research Gap 16

Chapter 3. Proposed Model 19
3.1. Project Contextualization 19
3.2. Kuramoto Model 20
3.3. Agents Interaction 22
3.4. Reinforcement Learning Algorithms 23
3.4.1. UCB and D-UCB 23
3.4.2. EXP3.P 25
3.4.3. Reward Calculation 26

Chapter 4. Simulations 31
4.1. Technologies and Tools Used 31
4.2. Construction of Agent Environments 33
4.3. Simulation Parameterization 35
4.3.1. Partner Selection Parameters 35
4.3.2. Oscillator Parameters 37
4.3.3. Reward Parameters 37
4.3.4. Environment Parameters 37
4.3.5. Phase Changes Parameters 38
4.4. Simulation Logic 38
4.4.1. Logic Before Iterations 38
4.4.2. Iterations Logic 39
4.4.3. Possible Logic Variations 40
4.5. Evaluation Metrics 41
4.6. Scenario Variations and Experiments 42
4.7. Output File Structure 43
4.7.1. State File 43
4.7.2. Trajectory File 44
4.7.3. Graph File 44

Chapter 5. Results 45
5.1. Exploration Guidelines 45
5.1.1. Impact of Reward and Network Characteristics on Agent Decisions 45
5.1.2. Adaptability in Stationary and Non-Stationary Environments 46
5.1.3. Comparison of Choices with Ideal and Direct-Communication Models 47
5.2. Exploratory Simulations Configurations 48
5.2.1. Relative contribution of the reward function components 49
5.2.2. Robustness to different network topologies 49
5.2.3. Robustness to noise 49
5.2.4. Adaptation to dynamic environments 50
5.2.5. Comparison with direct communication 51

5.3. Exploratory Simulations Results 52
5.3.1. Relative contribution of the reward function components 52
5.3.2. Robustness to different network topologies 53
5.3.3. Robustness to noise 53
5.3.4. Adaptation in dynamic environments 54
5.3.5. Comparison with direct communication 58
5.4. Final Results 59
5.4.1. Stationary Environment Comparison 60
5.4.2. Abrupt Alteration Environment Comparison 61
5.4.3. Gradual Alteration Environment Comparison 62
5.4.4. Close phase partner selection consistency 63
5.5. Discussion 64

Chapter 6. Conclusions and Future Work 65
6.1. Conclusion 65
6.2. Limitations and Challenges 66
6.3. Future Work 67

References 69

Appendix A. Additional Figures 73
A.1. Agents' Selections for Remaining Reward Combinations 73

Appendix B. Additional Plots 75
B.1. In-Depth Partner Selection Plots 75

List of Tables

1 Document amount with and without Inclusion Criteria and Exclusion Criteria across databases 13

2 Main parameters of the simulation and their explored ranges. 36

3 Simulations baseline parameters 48

4 Variations from the baseline used in the reward simulations 49

5 Variations from the baseline used in the noise simulations 50

6 Variations from the baseline used in the dynamic environments simulations 51

7 Variations from the baseline used in the simulations with and without communication 51

List of Figures

1.1 Planning schedule 3
1.2 DSR Structure 3

3.1 Comparison between an agent's trajectory without partner influence (a) and with partner influence (b). 21
3.2 Coupling Calculation Based on Selections 22
3.3 Illustration of phase coherence: individual neighbor phases represented on the unit circle and the average phase vector of the neighborhood. 28

4.1 Project workflow showing simulation, configuration, and result generation. 32
4.2 Different neighborhood configurations 34
4.3 Environments: All-connected, Perlin noise, Fade, and Maze. 34
4.4 Example of natural frequency alterations within a simulation. 35
4.5 Simulation Loop 39

5.1 Comparison of the reward combinations for each algorithm under the same simulation conditions and selections from Partial Reward 2. 53
5.2 Comparison of the different algorithms for each environment under the same simulation conditions. 54
5.3 Comparison of the different noise values for each algorithm under the same simulation conditions. 55
5.4 Comparison of the rewards in environments without frequency alterations for each algorithm under the same simulation conditions. 56
5.5 Comparison of the rewards in environments with abrupt frequency alterations for each algorithm under the same simulation conditions. 57
5.6 Comparison of the rewards in environments with gradual frequency alterations for each algorithm under the same simulation conditions. 58
5.7 Comparison of the rewards in environments with direct communication for each algorithm under the same simulation conditions. 59
5.8 Comparison of the performance in stationary environment for each algorithm under the same simulation conditions. 61
5.9 Comparison of the performance in environments with abrupt changes for each algorithm under the same simulation conditions. 62
5.10 Comparison of the performance in environments with gradual changes for each algorithm under the same simulation conditions. 63

A.1 Agent selections for the first two partial reward configurations. 73
A.2 Agent selections for the next two partial reward configurations. 73
A.3 Selections with Partial Rewards 1 and 3. 74

B.1 Agents' partner selection dynamics under the *no alteration* condition for different algorithms. 76
B.2 Agents' partner selection dynamics under the *abrupt alteration* condition for different algorithms. 77
B.3 Agents' partner selection dynamics under the *gradual alteration* condition for different algorithms. 78

List of Acronyms

MAB: Multi-Armed Bandits

UCB: Upper Confidence Bound

D-UCB: Discounted Upper Confidence Bound

DSR: Design Science Research

GSL: GNU Scientific Library

SLR: Systematic Literature Review

MARL: Multi-Agent Reinforcement Learning

ABM: Agent-Based Modeling

LLM: Large Language Model

CTDE: Centralized Training with Decentralized Execution

MADDPG: Multi-Agent Deep Deterministic Policy Gradient

IC: Inclusion Criteria

EC: Exclusion Criteria

HRI: Human-Robot Interaction

HRC: Human-Robot Collaboration

RL: Reinforcement Learning

PLV: Phase Locking Value

DFT: Discrete Fourier Transform

CHAPTER 1

Introduction

1.1. Context

Synchronization is a fundamental phenomenon observed across natural and artificial systems, from fireflies flashing in unison to humans walking in step or engaging in musical ensembles [**1**]. In social and cooperative settings, rhythmic synchronization plays a central role in enabling coordination, trust, and collective performance. These processes are not limited to humans: they are also evident in animal behavior, social groups, and artificial systems such as multi-robot teams or human–robot interactions [**2**].

In computer science and engineering, synchronization has become increasingly relevant in the design of adaptive and cooperative artificial agents. This paradigm extends to domains such as collaborative robotics in industrial settings and socially assistive agents in healthcare and education, where effective coordination with human partners depends on rhythmic and timely interactions [**2**]. By understanding how agents autonomously form partnerships and sustain synchronization, it becomes possible to design more natural and efficient systems for real-world collaboration.

Within artificial intelligence, this problem lies at the intersection of multi-agent systems, reinforcement learning, and computational models of rhythmic behavior. Multi-Agent Reinforcement Learning (MARL) provides a framework in which agents can learn to act autonomously while adapting to others [**3**, **4**], but most approaches rely on direct communication or pre-defined coordination strategies. In contrast, many natural systems achieve synchronization implicitly, through local interactions and dynamic adaptation. Exploring how artificial agents can replicate these processes without explicit communication offers both theoretical insight and practical benefits.

This dissertation situates itself as the study of partner selection in rhythmic multi-agent environments. By examining how agents choose interaction partners and adapt their rhythms to maximize synchronization, the work contributes to ongoing research in cooperative AI and dynamical systems. It builds on established theories of oscillatory synchronization while addressing open challenges in adaptive partner selection, with implications for fields as diverse as human–robot collaboration, distributed control, and the modeling of social coordination.

1.2. Research Questions

The following research questions were defined to guide the development of this study:

RQ1 - How can agents learn to autonomously select partners and achieve synchronization in multi-agent environments?

RQ2 - What mechanisms and reward structures can best promote coordinated oscillatory behavior and stable synchronization among agents?

RQ3 - How can Kuramoto oscillators, combined with a multi-armed bandit formulation, be used to represent the partner selection problem and evaluate different learning strategies?

1.3. Goals

The main goal of this dissertation is to model the problem of rhythmic partner selection in multi-agent environments, correct and update a simulation prototype, and conduct experiments that allow answering the research questions defined in the previous section. Specifically, the work aims to design and implement a framework where agents, modeled as Kuramoto oscillators, must autonomously select partners in order to maximize synchronization and cooperation.

To achieve this, the prototype integrates Multi-Armed Bandit algorithms for partner selection (UCB, Discounted-UCB, and EXP3.P), together with different reward formulations that capture synchronization performance, through systematic simulations under varied environmental conditions including stationary, abrupt, and gradual changes. The project evaluates how each algorithm adapts and which reward components are most effective.

By combining problem modeling, prototype development, and experimental validation, the dissertation seeks to contribute with both practical insights into the design of adaptive artificial agents and theoretical understanding of synchronization in complex multi-agent systems. Ultimately, the goal is to demonstrate how multi-agent reinforcement learning can replicate or even surpass communication-based approaches to partner selection, offering a scalable and flexible solution for real-world applications.

1.4. Planning

The planning schedule presented in Figure 1.1 organizes the work to be developed across the academic year 2024/2025. The initial months (October to December 2024) were dedicated to defining the scope of the project through discussions on the theme, as well as conducting a thorough literature review to identify research gaps. In parallel, the elaboration of the state of the art extended into early 2025, refining the research objectives and consolidating the theoretical foundation of the work.

From February 2025, the focus shifted to the design and implementation of the simulation platform, followed by the execution of simulations between March and May 2025. These two phases are closely linked, since the development of the simulation engine provided the basis for experimentation under different configurations. The subsequent stage, between April and July 2025, involved analyzing the results obtained, validating the models, and comparing algorithmic performance under diverse conditions.

In addition, the plan includes the preparation of a scientific article during the summer months (June to August 2025), ensuring dissemination of the main contributions.

Finally, the production of the dissertation spans from December 2024 until September 2025, accompanying all stages of the project, but intensifying in the last months with the integration of the results, conclusions, and final writing.

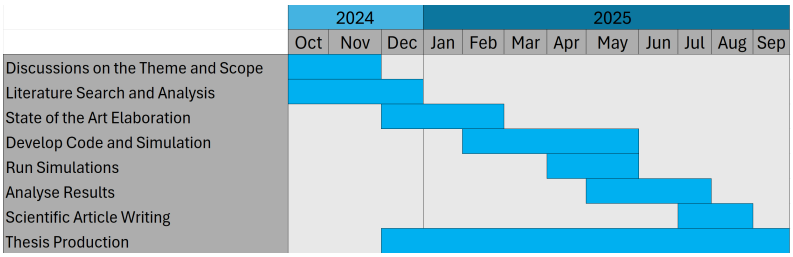


FIGURE 1.1. Planning schedule

1.5. Research Methodology

The investigation methodology used is Design Science Research (DSR), which is widely used in research fields such as computer science, engineering, and management. It focuses on creating and evaluating artifacts (such as models, frameworks, software, or processes) to address specific real-world problems. The DSR process typically involves six key phases, which the steps shown above can be split into.

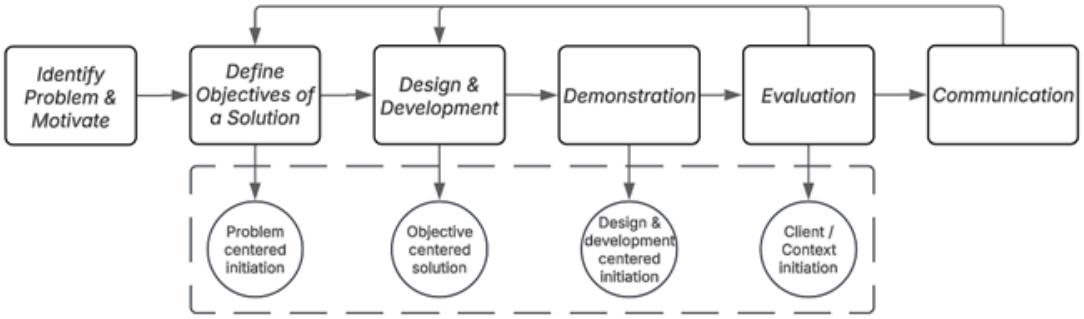


FIGURE 1.2. DSR Structure

Discussions on the Theme and Scope: This step aligns with Problem Identification and Motivation, where the research topic of partner selection in rhythmic multi-agent synchronization was chosen. The scope was narrowed to simulations with Kuramoto oscillators and multi-armed bandit algorithms (UCB, D-UCB, EXP3.P), focusing on how rewards influence adaptability in dynamic environments.

Literature Search and Analysis: Part of Defining Objectives of a Solution, this phase involved reviewing academic work on synchronization, reinforcement learning, and partner selection. Special attention was given to prior studies on multi-agent bandit algorithms and rhythm-based coordination, identifying gaps in how partial rewards and environment variability were handled.

State of the Art Elaboration: Within the objectives definition stage, this task deepened the investigation of current methods and frameworks, such as bandit formulations, MARL approaches, and oscillator models. From this, the goals were refined to

design a simulation platform that could test different algorithms under varied environmental conditions.

Develop Code and Simulation: This phase corresponds to Design and Development, during which the main simulation engine originally implemented in C++ using GNU Scientific Library (GSL) and JsonCpp was extended and refined. While the core system, responsible for modeling oscillators, reward functions, and partner selection algorithms, had been previously developed, several improvements and complementary tools were added as part of this work. These included the implementation of additional simulation features (Section 1.7), thorough code verification and testing procedures, corrections to existing modules, and the creation of Python scripts for generating plots and automating the configuration and execution of multiple simulation runs.

Run Simulations: Mapping to the Demonstration phase, this step executed systematic experiments under multiple environments (stationary, abrupt changes, gradual changes, maze, Perlin noise, etc.) using different parameter configurations, enabling controlled comparisons of algorithm performance.

Analyse Results: This task corresponds to Evaluation, where outputs such as reward trajectories, partner selection stability, and network visualizations were analyzed. The results validated the artifact's ability to highlight differences between algorithms and to answer the research questions about adaptability and reward structure.

Scientific Article Writing: Part of Communication, this task focused on preparing a paper where the methodology, experiments, and results will be structured and submitted for peer review.

Dissertation Production: The final step in Communication, formalizing the project's methodology, findings, and contributions in a comprehensive dissertation that synthesizes the literature, the simulation framework, and the experimental evaluation.

1.6. Dissertation Structure

The remainder of this dissertation is organized as follows:

Chapter 2 – State of the Art: reviews the literature on synchronization, partner selection, reinforcement learning, and oscillator models, highlighting existing approaches and identifying open challenges.

Chapter 3 – Proposed Model: presents the formulation of the problem, the design of the reward functions, and the integration of partner selection algorithms with Kuramoto oscillators.

Chapter 4 – Simulations: describes the simulation framework, experimental setup, and the different environments and conditions under which the algorithms were tested.

Chapter 5 – Results: reports the outcomes of the simulations, comparing the performance of UCB, D-UCB, and EXP3.P across different scenarios.

Chapter 6 – Conclusions and Future Work: analyzes and interprets the results in relation to the research questions, implications, and limitations of the work, and concludes with directions for future research.

1.7. Contribution to Code Implementation

The implementation of the code used in this dissertation was originally initiated by Professor Pedro Mariano, who provided a preliminary version of the codebase. Building upon this initial foundation, the author of the present dissertation undertook the tasks of debugging, correcting, and substantially expanding the code in order to meet the specific requirements of the research.

This process involved not only the refinement of the existing implementation but also the development of new functionalities, including the integration of the UCB algorithm, the implementation of the communication reward system, the introduction of mechanisms to modify and adapt the environment dynamically, and the creation of environment generators such as the Perlin Noise generator. Additionally, the output files were improved to enhance the quality and usability of the generated results. The author also automated a range of experiments and visualization tools, resulting in a framework that represents a significantly extended and adapted version of the initial code, tailored to the objectives of this work.

1.8. Dissemination

An academic article is currently under development, intended for submission in the field of multi-agent systems and artificial intelligence. The target venue is still being considered. The article aims to provide a comprehensive account of the research carried out in this dissertation, with particular emphasis on the proposed model of rhythmic partner selection based on Kuramoto oscillators and multi-armed bandit algorithms.

The structure of the article is being designed to offer a thorough overview of the problem addressed, the theoretical background, and the methods employed. It will include a review of the literature on rhythmic interactions and partner selection, situating the study within the broader academic discourse and highlighting the novelty of integrating oscillator dynamics with reinforcement learning techniques.

A substantial section will be dedicated to describing the design of the simulation environment and the implementation of the algorithms. This will include details on parameterization, evaluation metrics, and the variations of experimental scenarios used to assess the robustness and adaptability of the approach.

The article will conclude with an analysis of the results obtained so far, outlining how the proposed framework contributes to a deeper understanding of cooperative behavior in artificial agents and what potential applications it may have in human–robot interaction and industrial collaboration contexts. As the article is still under preparation, further refinements and extended analyses are expected before submission.

CHAPTER 2

State of the Art

The different fields that were studied had seen significant advancements in recent years, driven by interdisciplinary research and technological innovation. Understanding the state of the art was crucial for identifying gaps, building on existing knowledge, and guiding future research directions. This literature review aimed to provide a comprehensive analysis of the field by combining two complementary approaches: an ad hoc search and a Systematic Literature Review (SLR). The ad hoc search offered a flexible and exploratory overview of the different fields, highlighting key themes, seminal works, and emerging trends. This approach allowed for a broad understanding of the landscape, ensuring that foundational and cutting-edge research was acknowledged. Building on this, the SLR provided a rigorous and structured analysis, ensuring the review was thorough, unbiased, and methodologically sound.

2.1. Ad hoc Search

The ad hoc search conducted for this dissertation was primarily based on documents and references suggested by the supervisor, Professor Pedro Santana. These initial materials served as a foundation for exploring relevant topics and identifying key research directions. Building upon these references, additional sources were consulted to deepen the understanding of the subject matter and to ensure that the literature review was both comprehensive and aligned with the objectives of the work.

2.1.1. Rhythmic Interactions in Humans

Rhythmic interactions play a pivotal role in a wide range of agent interactions, from basic cooperative tasks to complex conversational exchanges requiring precise turn-taking [**5**]. Humans excel in these interactions, which enhance activities such as music-making, dance, and social bonding, ultimately fostering prosocial behavior [**6**]. Beyond their functional purposes, synchronization among agents strengthens social connections and promotes cooperation [**7**]. Research into rhythmic interactions has identified five key mechanisms that drive human cooperation: direct reciprocity, indirect reciprocity, spatial selection, multilevel selection, and kin selection [**8**].

2.1.2. Rhythmic Interactions in Artificial Agents

These principles extend beyond human contexts, finding applications in engineering and robotics, such as robotic locomotion [**9**], the coordination between robotic and human drummers [**10**], and conversational systems where turn-taking plays a critical role in human-robot interaction [**11**]. A critical factor in achieving stable cooperation is partner

selection, which has been extensively studied in both human and robotic dynamics [**12**,**13**]. In artificial systems, partner selection has increasingly been explored through reinforcement learning approaches [**14**], offering adaptive mechanisms for agents to optimize their interactions. Nevertheless, significant challenges remain in transferring these methods effectively to human-robot interaction contexts [**15**].

A possible interplay between rhythmic interactions and partner selection is synchrony detection for partner selection in human-robot interaction settings [**16**], which has also been applied to the study of coordination between patients with schizophrenia and social robots [**17**].

2.1.3. Agent-Based Modeling for Multi-Agent Coordination

Agent-Based Modeling (ABM) is a well-established paradigm for simulating and analyzing complex systems composed of multiple interacting and autonomous entities. Such systems are typically decentralized, non-linear, and characterized by emergent behavior arising from local interactions among agents [**18**]. In social and organizational contexts, computational agent-based models have been shown to provide richer insights into dynamic phenomena than traditional analytical approaches, as they explicitly capture adaptation, feedback, and heterogeneity [**19**]. Sequential decision-making in ABM is commonly formulated as a reinforcement learning problem, enabling agents to learn coordination strategies through interaction with the environment and other agents.

Recent work has focused on extending ABM with complementary data-driven and reasoning-based approaches. The integration of agent-based modeling with data mining allows agent behaviors and interactions to be derived and validated using real event logs, improving model realism and empirical grounding [**20**]. In parallel, the emergence of Large Language Model (LLM) based agents has introduced new possibilities for flexible communication, reasoning, and coordination in multi-agent systems. As surveyed by Guo et al. [**21**], LLM-driven multi-agent frameworks show promising results in collaborative and cooperative tasks, while also raising open challenges related to scalability, reliability, and evaluation. Together, these developments indicate a trend toward hybrid ABM frameworks that combine learning, data-driven modeling, and language-based interaction for complex multi-agent coordination.

2.1.4. Sequential Decision-Making and Reinforcement Learning

Sequential decision-making is often formulated as a reinforcement learning problem, with Multi-Armed Bandits (MAB) [**22**] algorithms providing a suitable framework for simplified settings [**23**], [**24**], [**25**]. MAB algorithms balance exploration and exploitation, enabling agents to maximize cumulative rewards based on environmental feedback. Among the most influential algorithms is the UCB [**26**], which operates under the principle of "optimism in the face of uncertainty." UCB assigns each action an index composed of the empirical mean reward and a confidence bound that decreases as the action is sampled more frequently. This ensures that actions with high uncertainty are explored,

while actions with consistently high rewards are exploited, leading to logarithmic regret bounds in stochastic environments.

A refinement of this approach is the D-UCB [**27**], [**28**], which introduces a discount factor on past rewards, giving more weight to recent interactions. This modification enables agents to adapt more effectively in non-stationary environments, where reward distributions evolve over time. In such cases, older experiences may no longer be reliable indicators of current dynamics, and D-UCB mitigates this by progressively "forgetting" outdated information.

Complementary to these stochastic approaches, the Exponential-weight algorithm for Exploration and Exploitation (EXP3) was introduced by Auer, Cesa-Bianchi, Freund, and Schapire [**29**] to address adversarial bandit settings, where rewards are not assumed to follow a fixed or stochastic distribution. EXP3 maintains a probability distribution over actions, updated multiplicatively according to the observed rewards, and ensures that even poorly performing actions retain a non-zero probability of selection. This probabilistic exploration mechanism guarantees robustness in environments where reward structures may be arbitrary or adversarial, providing regret bounds that hold without stochastic assumptions. However, the theoretical guarantees of EXP3 rely on the assumption of an oblivious environment, where the sequence of rewards is independent of the agent's previous actions. In other words, the adversary is not reactive but predefined, allowing any randomized policy to be viewed as an a priori random choice among deterministic strategies [**29**]. When this independence assumption is violated as in interactive or co-adaptive multi-agent systems where actions influence future rewards EXP3 may exhibit reduced stability and performance. Together, UCB, D-UCB, and EXP3 represent foundational strategies that extend the applicability of MAB frameworks from stationary stochastic environments to non-stationary and adversarial ones, thereby broadening their relevance in multi-agent and human-robot interaction contexts.

Beyond stochastic and adversarial formulations, extensions such as contextual bandits incorporate side information in the form of observable features that guide action selection. This formulation is particularly relevant in human-robot interaction scenarios, where contextual cues such as gestures, affective states, or task-specific conditions influence the likelihood of successful synchronization. From a learning perspective, contextual bandits provide a principled framework for exploiting shared structure across tasks and agents while adapting to heterogeneous context distributions. Recent work on distributed and multi-task contextual bandits [**30**] highlights how incorporating context under communication and coordination constraints can improve learning efficiency and robustness. In such settings, leveraging human-specific contextual signals not only enhances adaptability in dynamic environments but also supports more natural, trustworthy, and effective collaboration between humans and artificial agents.

2.1.5. Kuramoto Oscillators and Biological Inspiration

Kuramoto oscillators, widely recognized for their ability to characterize cortical network dynamics [**31**, **32**], have provided a simple yet powerful paradigm for understanding synchronization phenomena across natural and artificial systems. Since their original formulation [**33**], these models have been instrumental in explaining how large populations of coupled oscillators achieve self-entrainment and collective order, despite differences in individual frequencies.

In neuroscience and cognitive science, Kuramoto-based approaches have been applied to study synchronization in consensus formation [**34**] and to determine the maximum team size that still allows for effective coordination [**35**]. Phase synchrony in brain signals, often measured through phase-locking and coherence metrics, serves as a key indicator of neural coordination and communication [**36**]. By capturing these dynamics, Kuramoto oscillators provide a theoretical foundation for linking micro-level rhythmic interactions with macro-level patterns of cooperation among humans.

Beyond neuroscience, classical and diverse examples of synchronization observed in nature from Huygens' pendulum clocks and the coordinated flashing of fireflies to the rhythmic oscillations of cardiac pacemaker cells illustrate the universality of phase coupling [**1**, **32**]. These biological systems provide strong motivation for oscillator-based models such as Kuramoto's and inspire the design of artificial agents capable of achieving coordinated behavior in multi-agent environments. By drawing on both neural and natural synchronization phenomena, Kuramoto oscillators offer a principled approach to modeling and engineering coordinated interactions in complex distributed systems.

2.1.6. Multi-Agent Reinforcement Learning (MARL)

MARL has emerged as a powerful framework for addressing decision-making problems involving multiple agents interacting within a shared environment. Building on the principles of single-agent reinforcement learning, MARL extends the paradigm to scenarios where agents must adapt their policies not only in response to environmental dynamics but also to the evolving behaviors of other agents. The formal foundation for MARL was established through the introduction of Markov games [**37**], which generalize Markov Decision Processes to multi-agent settings, enabling the modeling of purely cooperative, purely competitive, or mixed interactions.

Over the past decades, numerous surveys have consolidated theoretical and practical advances in this field. A comprehensive review by Busoniu et al. [**38**] synthesized early approaches, encompassing tabular methods, heuristic coordination mechanisms, and distinctions between centralized and decentralized learning, while highlighting fundamental challenges such as scalability, non-stationarity, and credit assignment. With the advent of deep learning, MARL evolved into deep MARL, capable of operating in high-dimensional state and action spaces, thus widening its applicability to complex, real-world domains.

Hernandez-Leal et al. [**39**] provided a critical survey of this evolution, examining modern architectures, communication protocols, and stability issues in training, as well as strategies for balancing exploration and exploitation in multi-agent contexts.

2.1.7. Convergence Challenges in MARL

Despite the practical success of MARL, convergence of learning algorithms in multi-agent settings is generally not guaranteed. This stems from the inherent non-stationarity of the environment: as each agent adapts its policy, it simultaneously alters the dynamics experienced by other agents, violating the stationary assumptions underpinning classical reinforcement learning convergence proofs [**37**]. Early surveys highlighted additional challenges, including the exponential growth of the joint state-action space and difficulties in credit assignment, which further complicate stable learning [**38**]. Even with advances in deep MARL, strategies such as Centralized Training with Decentralized Execution (CTDE) and communication protocols can improve stability but do not ensure theoretical convergence in general cooperative, competitive, or mixed settings [**39**], [**40**]. Consequently, most MARL research emphasizes empirical stability and performance rather than formal guarantees, acknowledging that oscillatory or non-convergent behaviors may emerge in complex multi-agent interactions.

In addition to theoretical developments, practical algorithms have been proposed to address the unique challenges of MARL. Notably, the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm [**40**] implements a CTDE paradigm, where each agent has its own actor network but shares a centralized critic with access to the joint state and actions. This design enables agents to operate effectively in continuous and mixed cooperative-competitive environments and has become a cornerstone in modern MARL research.

Communication in MARL also plays a pivotal role in enabling agents to share information, coordinate strategies, and adapt to dynamic conditions. Communication can be explicit, where agents exchange direct messages through differentiable channels such as CommNet or TarMAC [**41**, **42**], or implicit, where coordination emerges from observable behaviors in the environment, such as movement patterns or timing signals. It may follow centralized paradigms, typically during training, where a global entity aggregates and distributes information (CTDE), or decentralized paradigms, where agents independently decide when and what to communicate based solely on local observations. The choice of communication paradigm influences scalability, robustness, and the ability to operate under partial observability. By capturing the adaptive and interactive nature of multi-agent systems, MARL provides a comprehensive foundation for modeling scenarios where coordination, competition, and learning occur simultaneously.

This study investigates MARL with a particular emphasis on the conditions under which learning convergence can be achieved or approximated in multi-agent settings, alongside an analysis of how different communication paradigms ranging from centralized information sharing during training to fully decentralized and emergent communication

mechanisms influence stability, scalability, and learning dynamics. By emphasizing these aspects, this work seeks to contribute to a more principled understanding of MARL in complex coordinated environments.

2.2. Systematic Literature Review

2.2.1. Keywords

To guide the identification of relevant literature, a set of specific keywords was defined to capture the scope of the research. The search was conducted using the following keywords: ("Resonance" OR "Synchrony" OR "Synchronization" OR "Rhythmic interaction") AND ("Partner Selection" OR "Cooperation" OR "Collaboration") AND ("Robot" OR "Agent") AND ("Learning" OR "Adaptive") AND ("Bandit" OR "Multi-armed bandit")".

These keywords were chosen to encompass both the conceptual and technical dimensions of the study, including the core phenomena of rhythmic interactions, the mechanisms of partner selection and cooperation, and the computational approaches used in learning and adaptive multi-agent systems. Using this set of keywords ensured that the literature review remained focused on studies directly relevant to human-robot and multi-agent interaction, while also including methods grounded in reinforcement learning and adaptive algorithms. By combining terms related to social and behavioral concepts with those related to computational techniques, the search strategy was able to capture interdisciplinary research spanning robotics, artificial intelligence, and social coordination studies. This approach facilitated the identification of high-quality, peer-reviewed articles that address both the theoretical foundations and practical implementations of synchronization, partner selection, and adaptive learning in agent-based and robotic systems

2.2.2. Inclusion Criteria

To ensure that the selected documents were both relevant and of high academic quality, a set of inclusion criteria was established. Documents were required to be written in English to guarantee accessibility and consistency in language. Additionally, the documents had to contain the specified keywords either in the title or abstract, ensuring that they addressed the main topics of interest. Only works within relevant subject areas, such as Computer Science, were considered, which helped focus the review on literature directly related to the study's scope. Furthermore, only documents published within the last five years were included, ensuring that the research considered was up-to-date and reflected the most recent developments in the field.

Finally, to maintain academic rigor and credibility, only peer-reviewed articles or papers were selected from the databases in Subsection 2.2.4. Collectively, these criteria allowed for the inclusion of literature that was current, reliable, and highly relevant to the research objectives.

2.2.3. Exclusion Criteria

Exclusion criteria were applied to remove documents that did not meet the quality or relevance standards required for the study. Documents were excluded if they were duplicates, as including the same work multiple times could bias the review and affect the synthesis of results. Studies that were not directly related to the defined keywords were also excluded, ensuring that only literature pertinent to the research questions was considered.

Additionally, publications from sources that were not peer-reviewed, such as informal reports, preprints without validation, or non-academic outlets, were removed to maintain the scholarly quality of the dataset. By systematically applying these exclusion criteria, the review process focused on retaining documents that were both relevant and methodologically sound, ultimately enhancing the reliability and validity of the study's findings.

2.2.4. Databases

The literature review for this study was conducted using articles and papers sourced from well-established academic databases, specifically IEEE Xplore, ACM Digital Library, and ScienceDirect. These platforms were selected due to their extensive collections of high-quality, peer-reviewed publications in the fields of computer science, engineering, and related disciplines.

By focusing on these databases, the study ensures access to up-to-date research, authoritative sources, and a broad range of relevant studies, thereby providing a solid foundation for a comprehensive and reliable review of the literature.

2.2.5. Query Results

During the search across various databases for the most relevant articles, a total number of documents was retrieved. Table 1 shows the number of documents obtained before and after applying the Inclusion Criteria (IC) and Exclusion Criteria (EC), as well as the final set of documents considered for analysis. Since there was a large amount of documents and not all retrieved documents were relevant to the research objectives, their contents were examined in detail, allowing for the selection of the most pertinent works and the identification of key themes presented in the Literature Review section.

TABLE 1. Document amount with and without Inclusion Criteria and Exclusion Criteria across databases

	Not using IC and EC	**Using IC and EC**
IEEE Xplorer	129	76
ACM Digital Library	2589	161
Science Direct	7869	50

2.2.6. Literature Review

2.2.6.1. *Synchronization in Human and Artificial Agents*

Synchronization is a fundamental aspect of interaction across biological, social, and artificial systems, influencing processes from human cooperation to multi-agent coordination. Previous research has explored how agents select partners for rhythmic entrainment, where synchrony serves as a critical factor in optimizing collaboration and social bonding [**43**]. This phenomenon is evident in natural group dynamics, where synchrony enhances social cohesion, trust, and mutual understanding, whether in simple interactions like walking in sync or more complex collective activities such as choir singing or team sports [**44**].

The significance of synchrony extends to artificial agents as well, particularly in robotics and multi-agent systems. In the realm of Human-Robot Interaction (HRI), synchronization is crucial for achieving natural and fluid cooperation between human and artificial agents. Robots designed to engage in social and collaborative environments often require mechanisms that enable them to synchronize their behaviors with human partners, thereby improving task performance, reducing cognitive workload, and enhancing user experience [**45**]. Sensorimotor synchronization, in which an artificial agent aligns its actions with external rhythmic stimuli, has been actively investigated through the use of adaptive virtual partners that dynamically adjust their behavior to human timing. Recent studies show that such virtual agents can successfully synchronize with humans in rhythmic tasks and even enhance coordination performance in joint interaction settings [**46**].

2.2.6.2. *Joint Action and Adaptive Synchronization*

Joint action, defined as the ability of multiple agents to coordinate their actions toward a shared goal, is deeply rooted in synchronization mechanisms. Whether in human-only or human-robot interactions, entrainment plays a key role in dynamically aligning actions, predicting partner movements, and improving overall performance in cooperative tasks [**47**]. This process is not limited to explicit communication; instead, it often emerges implicitly through mutual adaptation and continuous feedback loops between agents. In robotic applications, adaptive synchronization strategies have been proposed to enhance cooperative behavior in multi-agent settings. These strategies include phase locking, where the oscillatory dynamics of agents adjust to achieve stable synchronization and behavioral coupling, where agents modify their actions based on partner responses [**48**]. By leveraging these principles, multi-robot systems can perform highly coordinated tasks such as swarm robotics, automated assembly lines, and autonomous driving [**48**].

Additionally, body movement mirroring has been explored as an effective mechanism for fostering synchrony in human-robot teams. Studies have shown that when robots replicate human movements in real time, users perceive them as more intuitive and socially attuned, leading to improved interaction quality and collaboration outcomes [**49**]. Insights from human and animal synchronization, such as implicit cues, entrainment, and collective timing strategies, provide valuable guidance for designing robotic coordination

mechanisms. By mimicking these natural processes, artificial agents can achieve more fluid and robust interactions. This approach has been particularly beneficial in applications like therapy, rehabilitation, and educational robotics, where non-verbal synchronization plays a key role in engagement and learning.

2.2.6.3. *Reinforcement Learning for Partner Selection*

Partner selection in multi-agent environments is a complex problem, often framed within the context of Reinforcement Learning (RL). In scenarios where direct communication between agents is limited or unreliable, reinforcement learning provides a powerful framework for enabling agents to learn optimal interaction strategies autonomously [**50**]. Agents must iteratively explore potential partners, adapt their behaviors based on observed outcomes, and refine their selection criteria to maximize long-term rewards. In multi-agent partner selection, agents face a fundamental trade-off between exploring new potential partners and exploiting previously successful collaborations. Balancing these strategies is critical to optimizing long-term cooperation and avoiding suboptimal fixed patterns. Additionally, the dynamic nature of multi-agent systems introduces non-stationarity, as each agent's policy evolves over time. This requires agents to continuously adjust their partner selection strategies to maintain effective coordination.

Reinforcement learning has been successfully applied in various aspects of HRI, including emotional adaptation, where artificial agents learn to modulate their behavior based on human affective states [**51**]. By incorporating intrinsic motivation mechanisms, such as curiosity-driven exploration and social reinforcement signals, agents can develop more natural and context-aware synchronization behaviors. Furthermore, entrainment models inspired by human group dynamics offer valuable insights into optimizing collaboration, revealing patterns of synchronization that contribute to effective teamwork and coordination [**52**]. Overall, reinforcement learning enables adaptive, scalable, and context-sensitive partner selection, providing a robust foundation for both human-robot and multi-agent interactions.

2.2.6.4. *Semantic Communication and Initial Interaction Phases*

In addition to rhythmic synchronization, the role of semantic communication in multi-agent collaboration has been extensively studied. Recent research has emphasized the importance of shared representations and mutual understanding in facilitating coordination between agents, particularly in heterogeneous teams where humans and robots must work together [**53**]. Establishing common ground is a critical factor in ensuring smooth interactions and various computational models have been proposed to enhance semantic alignment through adaptive learning and contextual reasoning. The initial phase of human-robot communication plays a crucial role in shaping long-term collaboration outcomes. Studies have analyzed how early interactions influence subsequent engagement, trust formation, and synchrony development [**54**]. Robots that can effectively interpret

and adapt to human communicative cues from the outset are more likely to establish strong cooperative relationships. Early interaction phases have a lasting impact on subsequent collaboration, as initial patterns of communication and alignment influence the efficiency of later synchronization and the overall stability of cooperative behavior.

Additionally, virtual agents have demonstrated the ability to learn synchronization capabilities through adaptive decision-making frameworks, an approach particularly useful in applications such as psychotherapy, where aligning with human affective states is essential for successful interaction [**55**].

2.2.6.5. *Applications in Industry 4.0 and Beyond*

In industrial environments, Human-Robot Collaboration (HRC) has become a cornerstone of modern manufacturing, and recent trends highlight a shift from the traditional Industry 4.0 paradigm towards the emerging Industry 5.0 vision. While Industry 4.0 focuses on the integration of intelligent automation, cyber-physical systems, and data-driven technologies to enhance efficiency, Industry 5.0 places a stronger emphasis on human–centricity, resilience, and symbiotic interaction between humans and robots. Within this framework, adaptive synchronization models are essential to support dynamic and robust coordination between human operators and robotic systems, addressing variability in work rhythms, task demands, and individual operator behaviours [**56**].

Industry 5.0 envisions robots that not only augment human capabilities but also collaborate safely and intuitively with workers across tasks such as assembly, inspection, and logistics. Human–robot synchronization in such contexts can lead to improvements in productivity, safety, and operator well-being by allowing humans to focus on creative and judgment-intensive tasks, while robots execute repetitive or hazardous operations. However, heterogeneous teams of humans and robots introduce challenges related to adaptability, variability in human pacing, and unforeseen workflow interruptions. Consequently, there is a need for adaptive strategies including reinforcement learning, oscillator-based models, and semantic communication frameworks that enable robots to iteratively adjust their synchronization policies in real time, anticipate operator behaviour, and maintain stable performance across diverse operational conditions. By leveraging continuous feedback loops and learning mechanisms, HRC systems can improve coordination efficiency, reduce cognitive load on human workers, and support the broader human-centric goals of Industry 5.0.

2.3. Summary and Research Gap

This chapter reviewed the state of the art across rhythmic synchronization (see Subsections 2.1.1 and 2.1.2), partner selection (see Subsections 2.1.2 and 2.2.6.3), and adaptive decision-making in both human and artificial agents (see Subsections 2.1.4 and 2.1.6). The literature reveals that rhythmic entrainment plays a central role in cooperation, while reinforcement learning and multi-armed bandit algorithms provide powerful tools for adaptive partner selection. Moreover, oscillator-based models such as Kuramoto's

(see Subsection 2.1.5) offer a biologically inspired framework for describing synchronization dynamics in multi-agent systems.

Despite these advances, few studies have explored the direct integration of rhythmic synchronization mechanisms with learning-based partner selection frameworks. Most reinforcement learning and multi-armed bandit approaches focus on abstract reward optimization, without explicitly modeling temporal or oscillatory coordination between agents. Conversely, existing synchronization models rarely incorporate adaptive decision-making or reward-driven adaptation.

This dissertation addresses this gap by proposing a unified model that combines the rhythmic coupling of Kuramoto oscillators with decision-making algorithms based on the multi-armed bandit framework. By doing so, it seeks to bridge the divide between continuous-time synchronization and discrete-time partner selection, providing a novel approach to studying adaptive coordination in non-stationary multi-agent environments.

CHAPTER 3

Proposed Model

In order to formalise the partner selection problem, it was necessary to establish a theoretical foundation that combines principles from oscillator models and reinforcement learning. These concepts provide the basis for defining the mechanisms through which agents interact and adapt to their environment. Building on this foundation, new formulas were introduced to refine the agents' learning process and enhance their ability to make effective partner choices.

3.1. Project Contextualization

The proposed model assumes a set of agents A, distributed in a square bidimensional grid L of dimension $w_r \times w_c$. The interactions between agents can assume two distinct configurations: a Moore neighborhood, in which each agent $i \in A$ interacts only with its immediate neighbors $M_i \subseteq A$, thus limiting connectivity to adjacent agents; or a fully connected topology, in which all agents can potentially interact with each other.

At each iteration $n = 1, 2, \ldots$, each agent $p \in A$ selects an interaction partner from its set of neighbors M_p, according to a policy π_p. The partner selected by agent p at iteration n is denoted by $I_p(n) \in M_p$. This choice is modeled as a MAB problem, in which each neighbor corresponds to an arm, and the agent incrementally learns which partner maximizes its expected long-term return. Thus, the partner selection problem is formulated as an optimal sequential decision-making process, whose objective is to learn a policy π_p that maximizes the accumulated reward. Formally, the optimal policy π_p is the one that minimizes the expected accumulated regret over a finite horizon H, with regret being defined as the expected difference between the reward obtained by the learned policy and the reward that would have been collected by an optimal policy (an oracle that always chooses the neighbor with the highest expected value), where $\mu_p(n, q)$ represents the expected reward of agent p at iteration n if it selects neighbor q:

$$\mathbb{E}_{\pi_p}\left[\sum_{n=1}^{H}\left(\max_{j \in M_p}\mu_p(n, j) - \mu_p(n, I_p(n))\right)\right]. \tag{1}$$

Although the interaction of each agent with its neighbors can be represented as a local bandit problem, the overall system exhibits characteristics similar to those of MARL environments. This happens because multiple agents learn simultaneously in a shared environment; the environment is non-stationary from the perspective of each agent, since the reward distribution of each neighbor depends on the evolving policies of the others; the rewards of an agent depend not only on the characteristics of the environment but also on the adaptive behavior of other agents; and coordination and learning occur in

a decentralized manner. Regarding the communication paradigm, this model is similar to the implicit and decentralized communication models in MARL. It is called implicit because the agents do not exchange information directly with each other; instead, they "communicate" through their rhythmic dynamics, where the degree of synchronization achieved serves as an indirect signal of the quality of the interaction. This type of communication is also considered decentralized, since there is no global entity that aggregates or distributes information to all agents. Each agent only observes the results of its own local interactions and autonomously adjusts its policy. In this way, coordination emerges naturally from local interactions, rather than being imposed by explicit communication channels or centralized mechanisms.

3.2. Kuramoto Model

In this model, an oscillator is characterized by its natural frequency ω_p, its phase θ_p, and an influence component calculated from the interaction with its partners, weighted by the coupling coefficient $W_{p,q}$. Let $t = 0, \delta, 2\delta, \ldots, T$ denote the sequence of $S = \lfloor T/\delta \rfloor$ interaction time steps, each with duration δ ($T \bmod \delta = 0$). Depending on the strength of the connection between agents, the oscillator phase of agent p gradually adjusts to approach the phases of its most compatible neighbors, leading to partial or total synchronization.

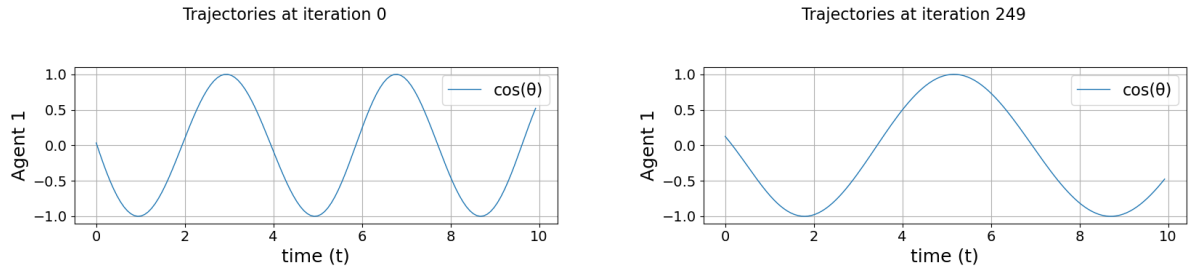
To represent realistic dynamics, two independent noise components are introduced. The first, an *action noise* term $\xi_p(t) \sim \mathcal{N}(0, \sigma)$, perturbs the agent's phase velocity to model uncertainty in its intrinsic behavior or decision-making process. The second, an *observation noise* term $O_{p,q}(t, n) \sim \mathcal{N}(\theta_q(t, n), \sigma_\theta)$, affects how agent p perceives the phase of its neighbor q, representing imperfect or noisy communication.

Considering both sources of noise, the phase evolution of each oscillator $\theta_p(t, n)$ follows the stochastic Kuramoto model:

$$\frac{\partial \theta_p(t,n)}{\partial t} = \omega_p(n) + \xi_p(t) + \sum_{q \in M_p} W_{p,q}(n) \sin\big(O_{p,q}(t,n) - \theta_p(t,n)\big), \tag{2}$$

where $\omega_p(n)$ is the natural frequency of oscillator p at iteration n, and M_p is the set of neighboring agents interacting with p. The combination of the two stochastic components ensures diversity in trajectories and prevents perfect deterministic synchronization.

The Figure 3.1 illustrates the temporal evolution of the trajectory of an agent in two distinct situations: in the first, at the initial moment of the simulation (iteration 0), a clean harmonic trajectory defined in timesteps $t \in [0, 10]$ is observed, without interference from partners, corresponding only to the natural frequency of the agent; in the second, at iteration 249, it is noted that the trajectory of the same agent slightly decreased its frequency, in addition to showing small perturbations and deviations in the waveform resulting from the influence of neighbors through the coupling $W_{p,q}$. This effect demonstrates how the Kuramoto model incorporates the interaction between agents, allowing the phases to dynamically adjust according to the established connections.



(A) Trajectory without partner influence (B) Trajectory with partner influence

FIGURE 3.1. Comparison between an agent's trajectory without partner influence (a) and with partner influence (b).

As mentioned earlier, the parameter ω_p is the component of the agent's frequency without the influence of the environment and can be defined as follows. This parameter is later used in the computation of one of the partial rewards. For this purpose, a noise component ξ is explicitly introduced in the frequency dynamics, which is essential to enable its subsequent removal during the reward calculation, thereby ensuring an accurate evaluation of the reward. The motivation for introducing this noise term is better discussed later (see Subsection 3.4.3):

$$\frac{d\theta_p^*(t,n)}{dt} = \omega_p(n), \tag{3}$$

$$\frac{d\theta_p^*(t,n)}{dt} = \xi_i(t) + \omega_p(n). \tag{4}$$

The Kuramoto model also defines the level of coherence among a set of agents, through the calculation of an average vector of the positions of the various oscillators. This value is greater the closer the phases of all oscillators are to each other:

$$r_p(t,n) = \left| \frac{1}{|M_p|} \sum_{q \in M_p} \exp(i\theta_q(t,n)) \right|. \tag{5}$$

According to this formulation, it is possible to compute a local synchronization metric that not only accounts for the phases of neighboring agents but also incorporates the agent's own phase. To achieve this, the term $\exp(i\theta_p(t,n))$ is explicitly included in the numerator and the term 1 in the denominator. These additional terms correspond to the contribution of agent p itself, effectively treating the agent as coupled to itself with unit weight. By doing so, the metric $r_p^w(t,n)$ reflects a weighted combination of the agent's and its neighbors phases, providing a more complete measure synchronization:

$$r_p^w(t,n) \;=\; \left| \frac{\exp\big(i\,\theta_p(t,n)\big) + \sum_{q \in M_p} W_{p,q}(n)\, \exp\big(i\,\theta_q(t,n)\big)}{1 + \sum_{q \in M_p} W_{p,q}(n)} \right|, \tag{6}$$

where the self-weight is defined as $W_{p,p}(n) \equiv 1$, ensuring that the agent's own trajectory is included on equal footing with its neighbors in the calculation of local synchrony.

3.3. Agents Interaction

To determine how strong the relationship between agents is, the parameter W_p, called coupling strength, is used. This is determined according to the choices of the agent and those of its neighbor in a given iteration. The relationship will be stronger if the agents choose each other mutually. $I_p(n)$ corresponds to the choice made by agent p in iteration n. $\hat{W}$ corresponds to the sum of v_1 and v_2 if both agents choose each other mutually. The value v_1 corresponds to the weight of being chosen by another partner, while the value v_2 corresponds to the weight of choosing a partner. As a rule, when summed, these should total one unit:

$$W_{p,q}(n) = \frac{\hat{W}_{p,q}(n)}{\sum_{k \in M_p} \hat{W}_{p,k}(n)}, \tag{7}$$

$$\hat{W}_{p,q}(n) = \mathbb{1}_{\{I_p(n)=q\}}\nu_2 + \mathbb{1}_{\{I_q(n)=p\}}\nu_1. \tag{8}$$

The diagram presented in Figure 3.2 simplifies an environment with three agents (p, q, k). In this case, three types of interaction can be observed: no interaction, unilateral interaction, and mutual interaction. The first case is the simplest, since as none of the agents choose each other, both couplings are zero, i.e., $W_{q,k} = 0$ and $W_{k,q} = 0$. The unilateral interaction between q and p shows that $W_{q,p}$ is only v_2 and $W_{p,q}$ only v_1, meaning that q chose p, but p did not choose q in return, resulting in a lower coupling strength. In the case of mutual interaction, both $W_{p,k}$ and $W_{k,p}$ have the same value because both agents chose each other. From this, it can be understood that the strength of coupling in each type of interaction is always higher in mutual interactions.

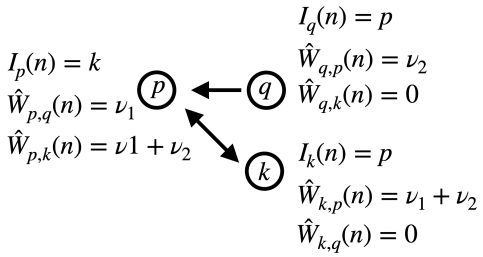


FIGURE 3.2. Coupling Calculation Based on Selections

In the context of MARL, the phase θ_p can possibly be interpreted as the internal state of agent p, reflecting its rhythmic dynamics over time. Similarly, the coupling $W_{p,q}$ represents the direct impact of partner selection actions, since it depends on the choices made by agents in each iteration. Finally, the reward function, explained in the following section, is built upon the synchronization capacity achieved, reflecting the quality of the emergent coordination between pairs or groups of agents.

The coherence and synchronization metrics derived from the Kuramoto model do not function merely as analytical descriptors of the dynamic system, but also constitute the

basis for the calculation of the partial rewards. In this way, the mathematical evolution of phases and couplings is directly connected to the learning process, since agents adjust their partner selection policies based on the observed levels of synchronization. This link between oscillatory dynamics and reward will be detailed in the subsection 3.4.3, dedicated to the calculation of the reward function.

3.4. Reinforcement Learning Algorithms

3.4.1. UCB and D-UCB

In terms of algorithms, two were used throughout this study, namely Upper Confidence Bound (UCB) and Discounted Upper Confidence Bound (D-UCB). The UCB and D-UCB algorithms follow the same basic logic but differ in a fundamental aspect. The distinction between them will be presented by first introducing the UCB algorithm, followed by the differences that lead to the D-UCB variant.

The UCB algorithm makes it possible to balance exploration and exploitation by choosing the action (or arm) with the highest upper confidence bound. For each action, the algorithm calculates the sum of the rewards already observed and the number of times that action was selected, dividing these two values to obtain the simple average of the rewards. To this average is added an uncertainty term (confidence bound), which decreases with the number of times that the action has been chosen. Thus, the algorithm favors actions that present good expected results, exploiting what has already proven advantageous, but also ensures that actions that have been chosen less often, and therefore carry greater uncertainty, are tested regularly, thereby promoting exploration:

$$U_{p,s}(n) = \bar{X}_{p,s}(n) + C\sqrt{\frac{\log(n_p(n))}{N_{p,s}(n)}}, \tag{9}$$

where $U_{p,s}(n)$ represents the upper confidence bound of agent p with respect to partner s at iteration n, $\bar{X}_{p,s}(n)$ corresponds to the simple arithmetic mean of the rewards already observed, $N_{p,s}(n)$ is the number of times partner s has been chosen, $n_p(n)$ is the total number of selections made by agent p up to iteration n, defined as

$$n_p(n) = \sum_{k \in M_p} N_{p,k}(n), \tag{10}$$

and C is the exploration parameter, which adjusts the weight of the confidence term. The higher the value of C, the more the algorithm favors the exploration of less-selected partners.

The Discounted UCB (D-UCB) algorithm is a variation of UCB that introduces an exponential discount factor over the interaction history. This factor makes it possible to give greater weight to more recent rewards and gradually reduce the influence of older ones, making the algorithm more adaptable to non-stationary environments in which the rewards associated with actions may change over time. The main formal difference

between UCB and D-UCB is precisely the introduction of the discount factor γ_d in the calculation of the statistics, which is absent in classical UCB.

In D-UCB, the value of $X_{p,s}(n)$, which corresponds to the discounted sum of the observed rewards, is obtained through the recursive formula:

$$X_{p,s}(n) = \gamma_d X_{p,s}(n-1) + \mathbb{1}_{\{I_p(n)=s\}} X_p(n), \tag{11}$$

where γ_d is the discount factor that takes values between 0 and 1, $\mathbb{1}_{\{I_p(n)=s\}}$ is the indicator function that equals 1 if agent p chose partner s at iteration n and 0 otherwise, and $X_p(n)$ is the reward received by agent p at iteration n. This mechanism ensures that older rewards gradually lose weight over time, allowing the agent to adapt better to changes in the environment.

Analogously, the discounted number of selections $N_{p,s}(n)$ is updated according to:

$$N_{p,s}(n) = \gamma_d N_{p,s}(n-1) + \mathbb{1}_{\{I_p(n)=s\}}, \tag{12}$$

which means that, as with rewards, older selections are progressively forgotten. The average used in D-UCB is no longer the simple arithmetic mean but instead becomes a weighted average with discount:

$$\bar{X}_{p,q}(n) = \begin{cases} \dfrac{X_{p,q}(n)}{N_{p,q}(n)}, & N_{p,q}(n) > 0, \\ 0, & N_{p,q}(n) = 0. \end{cases} \tag{13}$$

With these definitions, the D-UCB index retains the same structure as UCB, given by (9), but now with $\bar{X}_{p,s}(n)$ and $N_{p,s}(n)$ calculated with discount. This makes the algorithm more reactive to recent changes, since it gives less importance to past behaviors. The closer the value of γ_d is to 1, the more the behavior of D-UCB approaches that of classical UCB, since in that case there is no forgetting.

To ensure the proper functioning of the algorithm from the beginning, it is important that the initial values of U are defined as sufficiently high, thereby promoting exploration in the first iterations. The initial values of X and N should be set to zero, since no action has yet been taken and no reward observed:

$$X_{p,q}(0) = 0, \quad N_{p,q}(0) = 0, \tag{14}$$

The intention of both algorithms, after calculating the U values for all neighbors, is to allow the agent to select the action with the highest U:

$$I_p(n) = \arg\max_{q \in M_p} U_{p,q}(n). \tag{15}$$

This choice represents a decision that may favor a partner with higher average rewards, corresponding to exploiting a good option already known, or a partner still little explored, and therefore with greater uncertainty, which corresponds to exploring new alternatives.

With the introduction of the discount factor γ_d in D-UCB, the importance of the exploit component may become less dominant, since the algorithm gradually forgets past behaviors. This makes the confidence term gain greater influence in decisions, promoting adaptation and reactivity to changes in the environment.

3.4.2. EXP3.P

In addition to UCB and D-UCB, the Exponential-weight algorithm for Exploration and Exploitation with probability (EXP3.P) was employed to address partner selection in environments where rewards may be adversarial or non-stationary. Unlike UCB-based algorithms, which rely on confidence bounds of empirical averages, EXP3.P maintains a probability distribution over all possible partners and updates it multiplicatively according to the observed rewards. This probabilistic formulation allows the agent to balance exploration and exploitation even when the reward structure is unpredictable, providing robustness in scenarios where stochastic assumptions may not hold.

At each iteration n, agent p selects a partner $I_p(n)$ based on the probability distribution $P_p(n)$ over the set of available partners M_p:

$$I_p(n) \sim P_p(n), \tag{16}$$

The probability of selecting partner q is given by:

$$p_{p,q}(n) = (1 - \gamma_e)\frac{w_{p,q}(n)}{\sum_{k \in M_p} w_{p,k}(n)} + \frac{\gamma_e}{|M_p|}, \tag{17}$$

where $w_{p,q}(n)$ represents the weight associated with partner q at iteration n, $|M_p|$ is the total number of possible partners, and $\gamma_e \in (0, 1]$ is the exploration parameter that guarantees all partners are occasionally selected, preventing premature convergence to suboptimal options. The first term favors partners with higher accumulated rewards (exploitation), while the second term ensures a uniform probability of selection across all partners (exploration).

The weights are updated recursively using the observed reward $X_p(n)$ as follows:

$$w_{p,q}(n) = w_{p,q}(n-1)\ \exp\left(\mathbb{1}_{\{I_p(n)=q\}}\frac{\gamma_e X_p(n)}{|M_p| p_{p,q}(n-1)}\right), \tag{18}$$

where $\mathbb{1}_{\{I_p(n)=q\}}$ is an indicator function that equals 1 if agent p selected partner q at iteration n, and 0 otherwise. This update mechanism increases the probability of selecting partners that yield higher rewards while maintaining non-zero probabilities for less-selected partners, promoting continued exploration.

The initial weights are set uniformly to one for all partners:

$$w_{p,q}(0) = 1, \tag{19}$$

ensuring that the algorithm begins with no prior preference and allows for unbiased exploration in the first iterations.

The design of EXP3.P enables agents to remain responsive to fluctuating reward conditions. High-reward partners gradually gain higher selection probabilities, but due to the exploration term, all partners continue to be sampled, which prevents overcommitment to early successes. By adjusting the exploration parameter γ_e, the agent can control the relative influence of exploration versus exploitation: lower γ_e values prioritize exploitation, while higher values promote exploration and adaptability.

It is important to note that the theoretical guarantees of the EXP3 family assume an *oblivious* or non-reactive environment, where the sequence of rewards is independent of the agent's previous actions [**29**]. In such settings, any randomized policy can be viewed as an a priori random choice among deterministic strategies, simplifying the analysis and ensuring valid regret bounds. In interactive and co-adaptive multi-agent systems, however, this assumption is violated because each agent's decisions affect the future rewards of others through dynamic coupling and phase synchronization. As a result, EXP3.P may exhibit greater variability and slower convergence, not because of algorithmic inefficiency, but because it operates outside its theoretical validity regime. Despite this, its inclusion in this study remains valuable for assessing how an adversarial bandit algorithm behaves in an interactive, non-oblivious setting, and for contrasting it with the stochastic methods UCB and D-UCB.

3.4.3. Reward Calculation

This section presents two types of rewards used in the simulations. The first three are combined into one total reward for environments without communication, while the fourth reward presented, the communication reward, is used independently in environments where agents can exchange information.

3.4.3.1. *Phase Alignment Reward*

The first component of the reward function, represented by $R_{p,1}(n)$ (or R1 in the remainder of this work), corresponds to a measure inspired by the concept of Phase Locking Value (PLV), frequently used in neuroscience to quantify phase synchronization between two oscillatory signals. In this context, the metric is used to measure the degree of alignment between the phase trajectory of an uncoupled oscillator (that is, without influence from neighbors) and the phase trajectory of the same agent after the influence of neighbors, during iteration n. The equation calculates, for each time step s over the total interaction interval S, the phase difference between the two trajectories: the natural phase $\theta_p^*(s\delta, n)$, which represents the evolution of agent p's oscillator independently, and the coupled phase $\theta_p(s\delta, n)$, which results from interaction with neighbors. This phase difference is represented as a unit value on the complex plane, through the imaginary exponential $\exp(i \cdot \Delta\theta)$, so as to represent the angular phase on the unit circle.

The complex vector sum of the terms over the S steps, normalized by $1/S$, results in a complex number whose magnitude is then extracted by the absolute value operator. This

final value lies in the interval $[0, 1]$, being closer to 1 if the phase difference over time is constant. On the other hand, if the difference varies in a disorderly manner, the vector sum tends to cancel out and the result approaches 0.

This term favors situations in which the oscillatory dynamics of the agent, even under interaction with neighbors, remain close to its natural trajectory. The underlying idea is that, from the point of view of comfort or efficiency of the agent, the more its rhythmic behavior remains close to its natural behaviour, the better the result of that interaction will be. Thus, this component of the reward penalizes interactions that excessively deviate the agent from its comfortable oscillation.

An additional and important aspect to consider is the treatment of noise present in the coupled phase. Since the phase $\theta_p(s\delta, n)$ includes noise due to imperfect observation of the neighbors' phases, this noise could introduce instability or unjustified bias in the comparison with the natural phase. To mitigate this effect, the same noise added to the coupled phase is also applied to the natural phase θ_p^* during the reward calculation. This ensures that the comparison term reflects only the distortion caused by coupling and not the variations introduced by noise, thereby maintaining consistency in the evaluation of synchronization.

This careful modeling allows the agent to maximize $R_{p,1}(n)$ whenever its oscillation after coupling remains well synchronized with its own frequency and natural trajectory, promoting a form of rhythmic homeostasis within a dynamic social environment:

$$R_{p,1}(n) = \left| \frac{1}{S} \sum_{s=0}^{S-1} \exp(i(\theta_p^*(s\delta, n) - \theta_p(s\delta, n))) \right|. \tag{20}$$

3.4.3.2. *Coherence Reward*

The second component of the reward function, represented by $R_{p,2}(n)$ (or R2 along the study), is derived from the concept of phase coherence as formalized in the Kuramoto model. This metric aims to quantify the degree of synchronization between agent p and its neighbors during iteration n, assessing whether the oscillatory behavior of the agent is in tune with the behavior of the elements with which it interacts. The equation consists of the average, over S discrete time steps along the trajectory duration, of a local coherence measure $r_p^w(s\delta, n)$, which already incorporates the effect of coupling with neighbors. This local measure evaluates, for each instant s, how the agent's phase aligns with the average phase of its neighbors, taking into account the coupling strength assigned to each neighbor relationship.

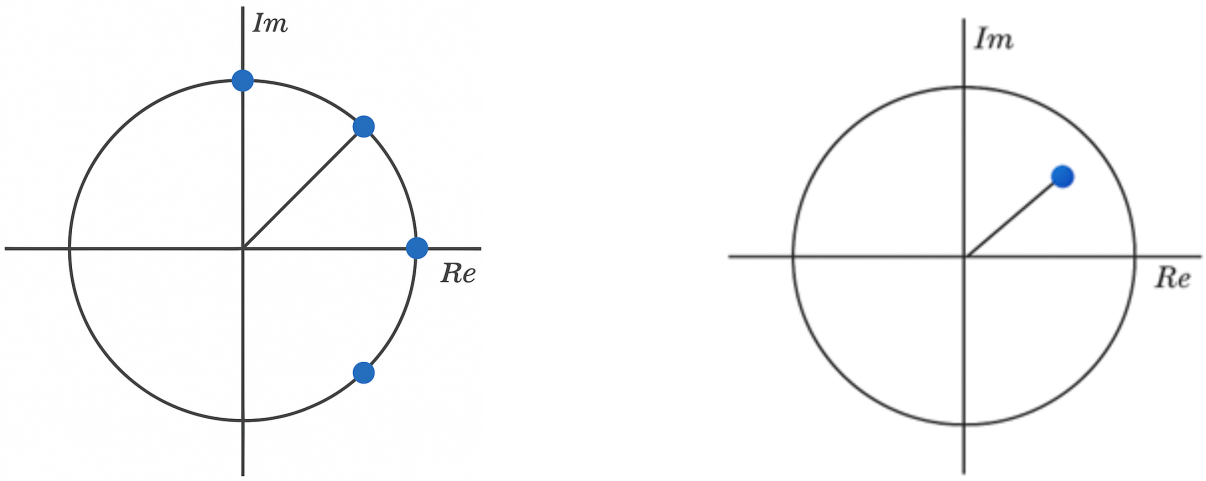
In this context, the term $r_p^w(s\delta, n)$ is calculated based on a Kuramoto-type formula, in which the agent receives weighted influence from the phases of its neighbors, modulated by a parameter that represents the coupling strength. The greater this coherence (that is, the more the neighbors' phases are aligned with the agent's), the higher the value of r_p^w, and consequently, the higher the final value of $R_{p,2}(n)$ after summation and normalization.

This reward term therefore values situations in which the agent is well synchronized with its neighbors, promoting a harmonious collective dynamic. Conversely, if the agent's phase deviates significantly from the phases of its neighbors, local coherence decreases, penalizing such desynchronization. Thus, the agent is encouraged to adjust its oscillatory dynamics to approach the global behavior of the surrounding network, reinforcing the idea of collective alignment.

It is also important to note that this reward component captures the quality of the agent's local coupling over time, being sensitive to fluctuations in synchronization along the trajectory of the iteration. The average over the S steps ensures that the evaluation is not based on a single instant but on a global analysis of the stability of synchronization, making the metric more robust and representative:

$$R_{p,2}(n) = \frac{1}{S} \sum_{s=0}^{S-1} r_p^w(s\delta, n). \tag{21}$$

The illustrations in Figure 3.3 serve to facilitate the understanding of the concept of phase coherence used in the second component of the reward function. In Figure 3.3a, the phase of each neighbor is represented as a point on the unit circle, corresponding to complex vectors of unit modulus. Their angular dispersion reflects the degree of misalignment among neighbors. In the right image, on the other hand, the average vector resulting from the sum of all neighbor phases is shown. The position of this vector inside the circle indicates the magnitude of coherence, being closer to the circumference when there is greater synchronization among neighbors' phases. It is important to stress that these images are merely illustrative and do not result from real data. They serve only to convey the geometric idea behind the calculation, and their numerical accuracy should not be considered rigorous.



(A) Neighbors Phases on Unit Circle (B) Mean Phase on Unit Circle

FIGURE 3.3. Illustration of phase coherence: individual neighbor phases represented on the unit circle and the average phase vector of the neighborhood.

3.4.3.3. *Spectral Similarity Reward*

The third component of the reward function, represented by $R_{p,3}(n)$ (also called R3 during the study), is based on the spectral analysis of the phase trajectory of agent p's oscillator throughout iteration n. This analysis is carried out through the Discrete Fourier Transform (DFT), applied to the sequence of values $\cos(\theta_p(s\delta, n))$, corresponding to the angular trajectory of the agent. The DFT makes it possible to decompose this signal into different frequency components, providing the complex coefficients $F_p(f, n)$ for each frequency f, which describe the energetic contribution of each frequency to the agent's oscillatory behavior.

The calculation of $F_p(f, n)$ follows the classical definition of the DFT, where each term results from the weighted sum of $\cos(\theta_p(s\delta, n))$ with complex exponentials, representing each frequency f in terms of periodic curves on the unit circle. This operation allows the identification of dominant frequencies in the oscillation during the trajectory of the iteration.

With the Fourier coefficients obtained, the reward $R_{p,3}(n)$ is calculated based on the spectral energy contained in those frequencies, with a Gaussian weighting centered on the agent's desired natural frequency $f_{\omega_p}(n)$. More precisely, the energy associated with each frequency f is given by $|F_p(f, n)|^2$, and it is multiplied by a Gaussian weight that reaches its maximum value when $f = f_{\omega_p}(n)$, decaying as f moves away from this value. The parameter σ controls the width of the Gaussian, that is, the degree of tolerance in the proximity to the natural frequency: lower values of σ make the reward narrower, rewarding only spectral energy concentration very close to the natural frequency, while higher values allow some flexibility.

The final value of $R_{p,3}(n)$ is the ratio between the total energy weighted by the Gaussian and the unweighted total energy. This ratio is always between 0 and 1, and represents the degree of alignment between the spectral distribution of the agent's phase trajectory and its ideal natural frequency. The more concentrated the energy is around $f_{\omega_p}(n)$, the higher the reward value. This mechanism makes it possible to evaluate the spectral quality of the oscillatory behavior, favoring trajectories in which the dominant frequency approaches the expected natural frequency.

This reward therefore acts as an evaluation filter of the purity of the agent's oscillatory behavior, reinforcing patterns that clearly and consistently express the intended frequency, and penalizing trajectories with dispersed or unstable components:

$$F_p(f, n) = \sum_{s=0}^{S-1} \cos\left(\theta_p(s\delta, n)\right) \cdot \exp(-i2\pi f s/S), \tag{22}$$

$$R_{p,3}(n) = \frac{\sum_f \left(|F_p(f, n)|^2 \cdot \exp((-(f - f_{\omega_p(n)})/(2\sigma))^2)\right)}{\sum_f |F_p(f, n)|^2}. \tag{23}$$

3.4.3.4. *Communication Reward*

The communication reward quantifies the natural compatibility between agents' intrinsic oscillations. It assumes that each agent can access the natural frequencies of its neighbors information that, in this context, would require communication with the environment. Its formulation follows the same logic as the first reward, using the PLV metric, but is applied directly to the natural phases of two distinct agents p and q, without noise or coupling effects. Importantly, this reward will be used separately from the partial rewards. This separation ensures that standard rewards, which evaluate agent performance without explicit communication, are not influenced by information sharing. In other words, communication is only leveraged when computing this specific reward, keeping the other reward mechanisms purely intrinsic.

This reward measures how closely two agents' natural rhythms evolve in synchrony, independently of any partner selection or external influence. It is computed as the magnitude of the average complex phase difference across the trajectory:

$$R_c(n) = \left| \frac{1}{S} \sum_{s=0}^{S-1} \exp(i(\theta_p^(s\delta, n) - \theta_q^(s\delta, n))) \right|. \tag{24}$$

The resulting value ranges from 0 (complete desynchronization) to 1 (perfect natural synchronization). Because it reflects intrinsic compatibility rather than learned coordination, the communication reward serves as a reference or upper bound for evaluating learning outcomes. It represents the ideal level of synchronization that agents could achieve if their natural frequencies were already aligned.

It is important to note that natural frequencies are not directly observable during interaction agents only perceive partner phases affected by coupling and perceptual noise. Consequently, partner selection is guided by the overall synchronization achieved through interaction rather than by direct frequency matching. Changes in natural frequency are perceived only indirectly, through phase variations, and cannot be isolated from other dynamic influences.

3.4.3.5. *Total Reward*

The calculation of the total reward X_p is computed as the weighted sum of the different partial rewards. When in a direct communication simulation, the total reward is instead defined solely by the Communication Reward, and the other partial rewards are not considered:

$$X_p(n) = \sum_{m=1}^{3} \zeta_m R_{p,m}(n), \tag{25}$$

$$X_p(n) = R_c(n). \tag{26}$$

The choice between these two approaches depends on the type of simulation being performed. In a simulation without communication, X_p is computed as the sum of the partial rewards, reflecting intrinsic agent interactions. In a simulation with communication, X_p can instead be set equal to the Communication Reward, since agents have access to each other's natural frequencies and can leverage this information.

CHAPTER 4

Simulations

To validate the functioning of these formulations, a prototype of the system was developed, in which simulations could be run to study the dynamics involved. These simulations require a set of agents and certain parameters, which will be presented below. From them, several conclusions were drawn that can be observed in the following chapters

4.1. Technologies and Tools Used

The development of this work was carried out in a Linux operating system environment. Although the prototype and simulator are not specific to this platform, using Linux proved to be more practical due to its stability, compatibility with low-level libraries, and ease in the automation of tasks. The main programming language used was C++, due to its high performance and direct control over memory management and computational structures. This language was responsible for the entire implementation of the simulation engine, where the agents, the neighborhood structure, the Kuramoto oscillatory models, and the partner selection algorithms, including UCB, Discounted-UCB and EXP3.P, were defined. A diagram with the code logic implemented in C++ is presented in the Section 4.4.

For computational support of the most demanding mathematical operations, the GSL was used. It was essential for the numerical integration of the ordinary differential equations that govern the evolution of the oscillators' phases. Through the use of the higher-order Runge–Kutta method, it was possible to accurately calculate the natural and coupled trajectories of agents over time. GSL was also used for the implementation of the Fourier transform, required for the spectral analysis component of the reward, as well as for the generation of Gaussian noise in the observation and action processes.

The configuration of simulation parameters, agents, and scenarios was performed externally to the source code, through files in JSON format. These files contained, among others, the definition of natural frequencies, initial phases, neighborhood type, exploration parameters, reward weights, and number of iterations. For efficient handling of these files within the C++ program, the JsonCpp library was used. This separation between configuration and simulation logic facilitated the reproduction and modification of scenarios while keeping the base code unchanged.

Data analysis and visualization of results were carried out using the Python language, chosen for its versatility and wide range of available scientific libraries. Several auxiliary scripts in Python were developed for the automatic reading and processing of the output files generated by the simulations. These scripts made it possible, for example, to generate

plots of the evolution of agents' average rewards, visualize the trend of UCB values over time, and build heatmaps of interactions and partnerships between agents. The Graphviz library was used together with Python to generate graphical representations of agent networks, highlighting, at each iteration, the selected partners, coupling weights, and the natural frequency of each node.

The entire project structure was organized in a modular way. The C++ code was isolated in the simulation component, the Python scripts handled visualization and analysis, the .json files defined the experimental parameters, and the results were automatically produced in .csv and .jpeg formats, organized into directories separated by simulation. The data produced by the simulations included phase trajectories, partial reward values, communication between agents, UCB values, and selections made over time. These data were stored in .csv files, later analyzed with libraries such as Pandas and NumPy, and visualized with Matplotlib and Seaborn. The general workflow of the system, including simulation execution, data handling, and result visualization, is illustrated in Figure 4.1.

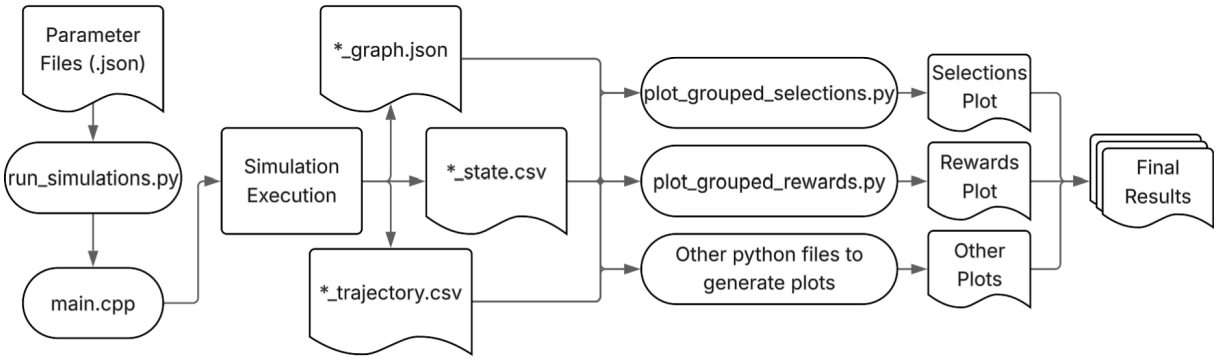


FIGURE 4.1. Project workflow showing simulation, configuration, and result generation.

Additionally, a Python script named `run_simulations.py` was developed, whose main function is to automate the execution of batches of simulations with different combinations of experimental parameters. This script allows dynamic modification of the values defined in the .json files that specify environments, generate consistent variations of the environment, and invoke the simulation binary with specific configurations, without requiring manual intervention. By passing arguments in the command line, the user can define lists of values for parameters such as the weights of the partial rewards (r1, r2, r3), the exploration factor (`-exp`), the discount factor (`-disc`), the gamma weight (`-gamma`), the standard deviations of noise (`-std_phase`, `-std_action`), as well as the activation of simulations with abrupt agent replacement (`-replace_sim`) and activation of simulations with gradual agent drift (`-gradual_drift`). A concrete example of execution is as follows:

```
python3 run_simulations.py ./simulator parameters.json \
    --exp=0.1,0.2 --disc=0.8,0.9 --gamma=0.3 --r1=0.2,0.4 --r2=0.3,0.3 \
    --r3=0.5,0.3 --cr=0,0 --std_phase=0.01,0.02 --std_action=0.05,0.07 \
    --replace_sim --gradual_drift --rewards --ducb,
```

all the parameters refered in this Section are presented with their respective ranges in Table 2 (see Subsection 4.3.5).

This command generates all possible combinations among the indicated values, executes the simulation for each of them, and saves the results in directories organized by configuration. In addition, the script can automatically trigger the visualization scripts, producing plots and images at the end of each simulation, plots about selections with "–ucb" and reward plots with "–rewards". This tool proved fundamental for ensuring reproducibility, efficiency in the experimentation process, and the ability to systematically analyze system behavior in varied scenarios.

The average time per simulation varies according to the number of agents, the number of iterations, the complexity of the selection algorithms used, and especially the amount of noise considered. However, the modular architecture and automation made it possible to run hundreds of simulations sequentially while maintaining total reproducibility.

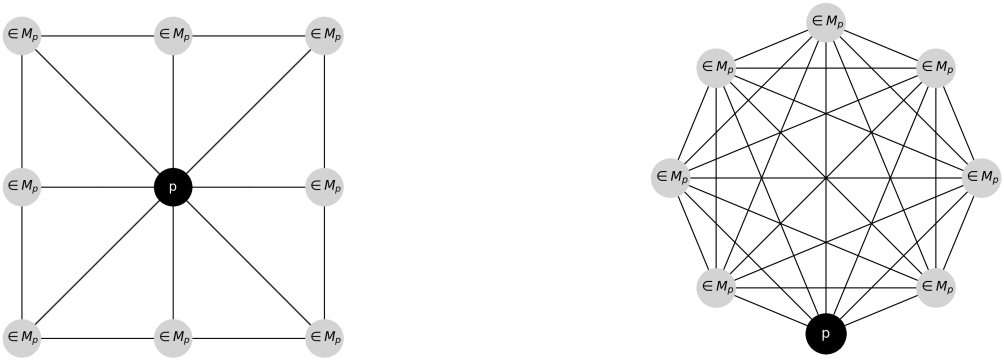
In summary, the development of this work involved a solid combination of tools and programming languages, where C++ ensured the computational performance required for the simulation, and Python provided an effective environment for the analysis and communication of results. The use of specialized libraries such as GSL, JsonCpp, and Graphviz, as well as the methodical organization of the project, contributed decisively to the quality and scalability of the simulations developed.

4.2. Construction of Agent Environments

For the simulations, a set of environments was created by varying both the spatial distribution of agents and their intrinsic properties, namely natural frequencies and initial phases. It is important to distinguish between two independent design dimensions: *neighborhood models* and *environment types*. Neighborhood models define how agents interact with one another, whereas environment types define how agents and their properties are distributed in space.

Two neighborhood models were considered. In the *Moore neighborhood model*, each agent p interacts only with its adjacent agents on the grid, forming a local neighborhood N_p composed of the surrounding cells. This model captures localized interactions and spatial constraints. In contrast, the *all-connected model* assumes global interaction, where the neighborhood of each agent contains all other agents in the system, i.e., N_p includes the entire population. Figure 4.2 illustrates these two interaction models, where agent p is highlighted and the corresponding neighborhood N_p is indicated.

While neighborhood models specify how agents interact, environment types specify where agents are placed and how their natural frequencies and initial phases are distributed. Several environment types were designed to expose agents to diverse and challenging conditions:



(A) Moore Model Neighbors (B) All Connected Neighbors

FIGURE 4.2. Different neighborhood configurations

- **All-connected random environment**: agents are assigned random natural frequencies and initial phases, and interactions follow the all-connected neighborhood model. This configuration removes spatial constraints and serves as a baseline for global synchronization behavior.
- **Perlin noise environment**: agents are arranged on a grid with natural frequencies distributed according to Perlin Noise [**57**], producing smooth, spatially correlated variations. Interactions in this environment follow the Moore neighborhood model, emphasizing local coupling under structured heterogeneity.
- **Fade (gradient) environment**: agents are placed on a grid with frequencies and phases varying gradually along a spatial gradient. As in the Perlin case, the Moore neighborhood model is used, allowing the study of synchronization under slowly varying local conditions.
- **Maze environment**: agents are distributed in a maze-like spatial structure, introducing irregular boundaries and obstacles that constrain interaction pathways. Natural frequencies are assigned according to the spatial layout, and interactions again follow the Moore neighborhood model.

Figure 4.3 presents examples of these environment types. In all cases, the color gradient represents agents' natural frequencies, with darker shades indicating lower values. By combining different environment types with distinct neighborhood models, the simulations explore how spatial heterogeneity and interaction topology jointly influence synchronization dynamics and agent behavior.

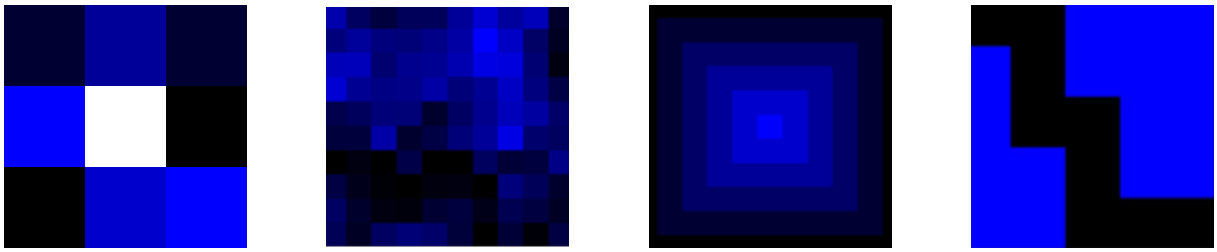


FIGURE 4.3. Environments: All-connected, Perlin noise, Fade, and Maze.

In addition, in some simulations the agents had their frequencies and phases altered to assess whether they would be able to adapt to adversities introduced after their learning and partner selection. In these cases, frequencies and phases were randomly altered at two points during the simulation: midway through and at three-quarters of the total duration, to further dynamize the environment. In parallel with these abrupt changes, there were also simulations in which the frequencies and phases were not changed suddenly but rather gradually, increasing or decreasing the frequency at each 5 iterations, the values added were generated by a normal distribution. This made it possible to understand in which of these cases the agents would adapt more easily.



(A) Environment Before Changes

(B) Environment After Changes

FIGURE 4.4. Example of natural frequency alterations within a simulation.

4.3. Simulation Parameterization

The simulations carried out in this study are based on a set of fundamental parameters that determine the behavior and interactions of the agents over time. Each simulation is accompanied by a configuration file with the extension `.parameters.json`, in which all the values necessary for the execution of the model are defined. This file serves as the structural basis that guides the decision-making process, the dynamics of the oscillators, and the way rewards are calculated.

Table 2 provides an overview of the main simulation parameters and the ranges of values explored. The parameters are grouped by category, and each will be described in detail in the subsequent subsections, highlighting their role in the simulations and the effects they have on the agents' behavior.

Among these parameters, the first one specified in the configuration file is the seed of the random number generator, which guarantees the reproducibility of results. By keeping the seed value the same across simulations, the observed behaviors will be identical in different runs, allowing for more rigorous and controlled analysis.

4.3.1. Partner Selection Parameters

These parameters concern the functioning of the reinforcement learning algorithm used for partner choice. The field `type` defines the active algorithm in the simulation, which can take the values `UCB`, `discounted-UCB` or `EXP3.P`, depending on the desired

TABLE 2. Main parameters of the simulation and their explored ranges.

Category	**Parameter**	**Range / Values**
Partner Selection	Type	{UCB, D-UCB, EXP3.P}
	Exploration factor (c)	$[0.1, 2.0]$
	Discount factor (D-UCB, γ_d)	$[0.8, 0.995]$
	Gamma weight (EXP3.P, γ_e)	$[0.01, 1.0]$
	Coupling strength selected	$[0.0, 1.0]$
	Coupling strength selector	$[0.0, 1.0]$
	(Sum of couplings)	$= 1.0$
Oscillator	Std. deviation phase observation (σ_θ)	$[10^{-4}, 1.0]$
	Std. deviation action (ξ_i)	$[10^{-4}, 1.0]$
	Trajectory duration (t)	$\{10\}$
	Trajectory sampling period	$[0.05, 0.1]$
Reward	Weight first partial reward (ζ_1)	$[0.0, 1.0]$
	Weight second partial reward (ζ_2)	$[0.0, 1.0]$
	Weight third partial reward (ζ_3)	$[0.0, 1.0]$
	Weight communication partial reward	$\{0.0, 1.0\}$
	(Sum of weights)	$= 1.0$
Environment	Number iterations	$[1000, 20000]$
	Population type	{moore_rectangle, all_connected}
	Population rows, cols	$[2, 10]$
	Agents initial phase (θ_p)	$[0, 2\pi]$
	Agents natural frequency (ω_p)	$[0.5, 6.0]$
Phase Changes	Replace Sim	{True, False}
	Gradual Drift Sim	{True, False}
	Drift Base	$[10^{-5}, 5 \times 10^{-3}]$
	Drift Std.	$[10^{-5}, 1 \times 10^{-3}]$
	Sigma Rw.	$[10^{-6}, 5 \times 10^{-4}]$
	Wmin, Wmax	$[0.5, 1.0], [5.0, 6.0]$

method. For the D-UCB algorithm, the parameter `discount_factor` becomes relevant, as it allows greater or lesser weight to be assigned to more recent actions. This factor varies between 0 and 1: the closer to 1, the more similar the behavior of D-UCB will be to that of classical UCB, since there will be less forgetting of past decisions.

Another important parameter in this section is the `exploration_factor`, applicable to both UCB and D-UCB. This defines the degree of exploration of the algorithm, that is, how much the agent is willing to try new options. Low values promote a more conservative and exploitative behavior, whereas high values make the selection closer to a random choice.

There is also the parameter `gamma_weight`, which is specific to the EXP3.P algorithm. This parameter controls the degree of exploration versus exploitation by determining the weight of the uniform probability component in the selection distribution. Low values of `gamma_weight` make the algorithm rely more heavily on the learned weights, favoring partners that have historically yielded higher rewards (exploitation). Higher values increase

the influence of the uniform term, causing the agent to explore all partners more evenly, including less-selected ones, and thus enhancing adaptability in dynamic or adversarial environments. In this way, `gamma_weight` directly affects how reactive and exploratory the agent's behavior will be during the simulation.

The parameters `coupling_strength_selected` and `coupling_strength_selector` define the intensity of influence between agents. The former refers to the influence felt by the agent when it is chosen by a partner, while the latter refers to the influence exerted when selecting a partner. Both impact how the agent's oscillator is altered as a result of interaction, and their sum should not exceed the value of 1, in order to maintain coherence and normalization of the coupling dynamics.

4.3.2. Oscillator Parameters

This section concerns important parameters to characterize the internal behavior of each agent's oscillator. The first two, `standard_deviation_oscillating_bandit_phase_observation` and `standard_deviation_oscillating_bandit_action`, represent the noise introduced in the observation of the phase and in the agent's action, respectively. These values, expressed as standard deviations, simulate uncertainty or imperfection in the perception and execution of oscillatory behavior. The larger they are, the greater the variability of the observed and executed data. Also defined are `trajectory_duration`, which specifies the temporal duration of each trajectory simulated per iteration (that is, the evolution time of the oscillator), and `trajectory_sampling_period`, which indicates how often phase values are recorded along that trajectory.

4.3.3. Reward Parameters

The parameters define the weights assigned to each of the components that make up the global reward function. The `standard_deviation_energy_based_gaussian_weight` is used in the calculation of the third reward, which depends on the spectral analysis of the trajectory. The weight `weight_first_partial_reward`, `weight_second_partial_reward`, and `weight_third_partial_reward` indicate the proportion with which each of these components contributes to the final value of the agent's total reward. In addition to these reward weights, there is also `weight_communication_reward`, which indicates the weight of the communication reward.

Finally, the parameter `number_iterations` indicates the total number of iterations of the simulation, that is, how many times each agent will make decisions and interact with its neighbors.

4.3.4. Environment Parameters

The definition of the agent population is described in the section `population`. In this case, a neighborhood model of type `moore_rectangle` is used, which means that agents are arranged in a rectangular grid and interact with neighbors surrounding them in all directions (including diagonally). It is also possible to use the model `all_connected`, in which all agents are part of each agent's neighborhood. The parameters `rows` and `cols`

determine the size of the grid, while the field `agents` contains the list of agents with their individual attributes. Each agent is defined with an `initial_phase`, that is, its initial phase at the start of the simulation, which must be expressed in radians and lie in the interval $[0, 2\pi]$, as well as its `natural_frequency`, which dictates the agent's natural oscillation rhythm.

4.3.5. Phase Changes Parameters

The evolution of agent phases is controlled by a set of parameters that govern drift, noise, and the limits of natural frequencies. The boolean parameter `Replace Sim` determines whether the simulation environment has the random abrupt phase changes at 50% and 75% iterations. Similarly, `Gradual Drift Sim` enables or disables gradual deterministic drift in the agents' natural frequencies over time. The following parameters are only applied to gradual drift sims, `Drift Base` specifies the mean deterministic drift per iteration, which should be small enough to produce slow frequency evolution over thousands of iterations, while `Drift Std` sets the standard deviation of random variation in the drift between agents and should remain smaller than the base drift to maintain stable dynamics. Short-term random fluctuations in frequency are controlled by `Sigma Rw`, which defines the magnitude of random walk noise; values that are too large may cause chaotic jumps in frequency. Finally, the natural frequency of each agent is constrained by `Wmin` and `Wmax`, which define the lower and upper bounds of allowable natural frequencies, respectively.

This file, as a whole, represents all the variables involved in the simulation and allows the behaviors of the agents to be controlled, modulated, and replicated in a rigorous and flexible way. The organization into thematic blocks facilitates the customization of specific scenarios, which is essential for the detailed analysis of different algorithms and interaction strategies. In Table 2 are presented the parameters that the `.parameters.json` needs to start a simulation.

4.4. Simulation Logic

Each simulation, although it may vary in its parameters and in the type of environment considered, follows a well-defined sequential logic common to all scenarios. This execution logic is implemented in the main C++ source file and is based on the formulas and theoretical principles previously discussed, such as reinforcement learning algorithms, the coupling dynamics of oscillators, and the calculation of the various components of the reward. The Figure 4.5 presents the simulation loop that will be explained along this Section.

4.4.1. Logic Before Iterations

The process begins even before the first iteration with the calculation of the natural trajectory of each agent. This trajectory corresponds to the evolution of the agent's oscillator over a given time interval, without any influence from neighbors. This simulation serves as a reference for later comparisons with coupled trajectories. The calculation of

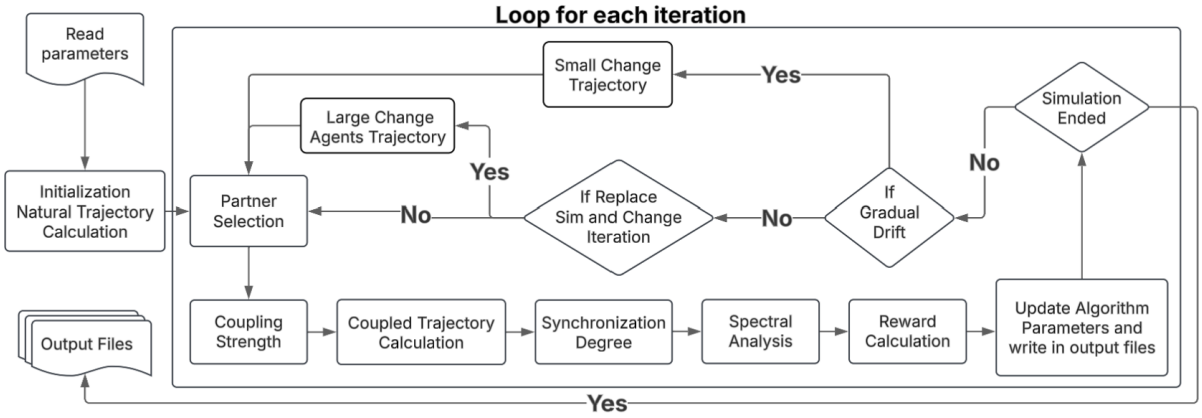


FIGURE 4.5. Simulation Loop

these natural trajectories is based on the natural frequency of the agent and on the parameters defined for trajectory duration and sampling frequency, and does not include any type of noise, being calculated deterministically from the natural frequency and simulated time, so that it can serve as a stable reference trajectory for comparison with coupled trajectories.

4.4.2. Iterations Logic

From this point, the iterative cycle that defines the learning and adaptation of the agents over time begins. In each iteration, the first operation is the initialization of the relationship state between the agents. For each agent, all neighbors are temporarily considered with `coupling_strength = 0`, and the states `selected` and `coupled` are set to false. This step ensures that each new iteration starts in a neutral state, avoiding interference from previous choices.

Next comes the partner selection phase, conducted by the learning algorithm defined in the simulation parameters. Depending on the configured value, the algorithm may be UCB, D-UCB or EXP3.P. In the case of UCB-based algorithms, the confidence value $U_{p,q}(n)$ is calculated based on the weighted average of rewards and on an exploration term that depends on the number of times each partner has been selected. Partner selection is performed independently and simultaneously by all agents (see Equation 9).

For EXP3.P, partner selection is performed probabilistically where each agent maintains a weight for every potential partner, which is updated multiplicatively based on the rewards obtained from previous interactions. The probability of selecting a partner is determined by a combination of these weights and a uniform exploration term controlled by the parameter `gamma_weight`. Partners that have historically yielded higher rewards have higher selection probabilities, but even less-explored partners retain a chance of being chosen, allowing the agent to continuously balance exploration and exploitation in dynamic or adversarial environments (see Equation 17).

After this phase, the coupling strength values are determined. Each agent calculates the intensity of influence assigned to each selected partner, and this intensity is normalized

to ensure that the sum of the coupling forces assigned to partners equals 1. The coupling strength between two agents also depends on the parameters `coupling_strength_selected` and `coupling_strength_selector`, which regulate the weight given to the fact that an agent is chosen or that it chooses another agent. These values are combined to determine the degree of influence each partner will have on the agent's final trajectory.

With the coupling forces defined, the agent's trajectory is modified, i.e., a new trajectory (the coupled trajectory) is calculated, influenced by the trajectories of the selected neighbors. The impact of this influence depends on the coupling strength assigned to each partner and on the natural trajectories of those partners. During this process, action noise is again introduced, simulating execution imprecision.

Once the coupled trajectory is determined, several auxiliary calculations are carried out to obtain the components of the reward function. The first calculation is the synchronization degree between natural and coupled trajectories, according to the Phase Locking Value metric. In addition, phase coherence with selected partners is evaluated, quantifying phase alignment in real time during the trajectory. Finally, spectral analysis is carried out through the DFT, applied to the coupled trajectory. This transformation makes it possible to identify the dominant frequencies of the oscillator and compare the spectral energy with the agent's natural frequency, weighting the energy concentration with a Gaussian profile.

Each of these analyses leads to the definition of a partial reward (see Equations 20, 21, 23): the first related to maintaining the natural rhythm, the second to coherence with selected partners, and the third to the spectral purity of the trajectory. With these partial rewards, a weighted combination is then performed using the weights defined in the parameter file, resulting in a total reward for each agent.

Before the end of each iteration, the reward and selection frequency data are used to update the internal parameters of the learning algorithm. In the case of UCB, the averages and counts are updated. This update allows the agent to refine its future decisions based on the reward obtained in the current iteration.

4.4.3. Possible Logic Variations

Additionally, during simulation execution, modifications may occur that directly affect the natural frequencies of the agents. The first mechanism corresponds to gradual change, activated through the option `-gradual_drift`. In this mode, the natural frequency of an agent is progressively modified over several iterations. In the gradual drift mechanism the natural frequency of some agents is not replaced at once but slowly adjusted over many iterations. Instead of an abrupt change, the value moves step by step, combining a fixed slope with small random perturbations, until it stabilizes within the allowed bounds.

This behavior mimics slow adaptation processes such as wear or environmental influence and allows the algorithms to be tested in non-stationary conditions. The parameters that regulate this process are the base slope of the drift, the random deviation applied to each agent's slope at the start, the standard deviation of the noise added at every iteration

and the minimum and maximum limits that clamp the frequency to a realistic range. By adjusting these values the simulation can represent anything from almost imperceptible long-term changes to faster drifts that still occur progressively. In each iteration in which this variation occurs, the agent's natural trajectory is recalculated, ensuring that the Fourier transforms and all subsequent calculations correctly reflect its current dynamic state.

The second mechanism corresponds to abrupt change, activated through the option `-replace_sim`. In this case, it is checked whether the current iteration coincides with the moment when the system must be updated and, when that happens, the initial phase and natural frequency parameters of the selected agents are replaced with new values, simulating the sudden entry of different individuals into the system. This replacement represents a sudden change in the environmental conditions and requires the immediate recalculation of the agent's natural trajectory and Fourier transform, in order to ensure that future decisions faithfully incorporate its new dynamic characteristics.

This cycle repeats until the defined number of iterations is reached, allowing observation of how agents evolve, learn, and adapt their behavior based on the rewards they receive and the interactions they establish over time. The C++ code manages this entire process with high precision, ensuring a realistic and coherent simulation of distributed learning in systems of oscillatory agents.

4.5. Evaluation Metrics

This section details the metrics used to evaluate the system's performance during the simulations. The main metric considered throughout the experiments was the average reward per iteration, computed as a weighted mean of three components: phase synchronization (via the Phase Locking Value), synchronization with neighbors (through the order parameter), and spectral similarity between the agent's trajectory and its natural frequency. This combined measure captures both the preservation of intrinsic dynamics and the effectiveness of synchronization with selected partners.

Although regret is commonly used in multi-armed bandit problems, it is unsuitable in this context due to the non-stationary and interdependent nature of the agents' couplings. Because the optimal reward at each iteration cannot be precisely defined, regret would not meaningfully represent system performance. For this reason, direct reward-based metrics were preferred, as they are measurable, comparable, and provide a reliable basis for evaluating the different algorithms and conditions.

Cumulative reward over time was also analyzed to identify long-term learning and adaptation trends, offering a stable view of overall performance while attenuating short-term fluctuations observed in per-iteration rewards.

An additional and particularly informative metric is the communication reward, which quantifies the rhythmic compatibility between an agent and its chosen partner based solely on their natural frequencies and phases, without coupling effects. It represents how well two agents would synchronize if interacting only through their intrinsic dynamics, thus

serving as an indicator of the natural affinity underlying partner choices. This metric complements synchronization-based measures by isolating the quality of partner selection independently of interaction dynamics.

For visual analysis, several representations were employed. Temporal plots show mean agent rewards with standard deviations, revealing stability and variability across iterations. Interaction matrices (heatmaps) illustrate how frequently each agent selected each neighbor, effectively mapping the learned social structure. Additional plots display the evolution of upper confidence bounds in algorithms such as UCB, allowing direct observation of the exploration–exploitation balance.

Complementary visualizations, such as animations depicting the evolution of the interaction matrix and confidence values over time, were also produced to provide a more intuitive understanding of decision dynamics particularly during perturbation phases when odd-indexed agents are replaced highlighting how rapidly the system reestablishes coordination.

In summary, these metrics and visual tools jointly provide a comprehensive evaluation of the agents' individual and collective behavior, their adaptive learning capability, and the evolution of decision-making over time.

4.6. Scenario Variations and Experiments

To assess the robustness, adaptability, and effectiveness of the proposed partner-selection model based on coupled oscillators and multi-armed bandit algorithms, several experiments were conducted with systematic variations in the system's structural and behavioral parameters. These variations were designed to expose agents to diverse topological and dynamic challenges, enabling comparative analysis under different levels of complexity, noise, and unpredictability. Two network topologies were tested: the Moore neighborhood, where each agent interacts only with adjacent neighbors on a two-dimensional grid, and the fully connected topology, where any agent can select any other as a partner. The fully connected structure offers greater choice but higher learning complexity, while the Moore neighborhood, though more constrained, may promote local stability and faster convergence in low-diversity contexts.

Agents natural frequencies and phases were initialized using four distributions: all connected with random distribution, linear gradients (fade), maze distribution and Perlin Noise patterns (as described in Section 4.2). These setups simulate environments with varying rhythmic organization. Smooth gradients or clustered Perlin Noise distributions are expected to facilitate local synchrony, whereas fully random configurations pose greater challenges for stable pair formation.

The main comparison focused on three partner-selection algorithms: UCB, Discounted-UCB (D-UCB), and EXP3.P. UCB, relying on stationary rewards, should perform best in stable environments; D-UCB, which discounts older observations, is expected to adapt better to gradual changes; and EXP3.P, with its probabilistic exploration, is suited to non-stationary or adversarial reward dynamics.

A parameterization study was also performed, varying the weights of reward components (Section 3.4.3), the exploration factor, and the noise levels associated with actions and observations. This analysis aimed to determine how fine-tuning these parameters affects reward stability, communication coherence, and convergence speed.

To test adaptability, two types of dynamic perturbations were introduced: abrupt replacement (`replace_sim`), where frequencies and phases are reset when the simulation reaches 50% and 75% of the total iterations, and gradual drift (`gradual_drift`), where slow, continuous changes affect only odd-indexed agents. The latter creates an asymmetric scenario that tests the network's ability to maintain coordination when only part of it evolves. These experiments evaluated how well the system readapts after the establishment of initial pairing policies. Algorithms such as D-UCB and EXP3.P, which emphasize recent experiences or sustained exploration, were expected to recover more effectively than UCB, whose convergence tends to reduce sensitivity to change.

All combinations of topological, initialization, and algorithmic variations were executed systematically using the `run_simulations.py` script (Section 4.1). Taken together, these experiments aim to identify the conditions under which each algorithm achieves stable rhythmic coordination and to delineate the boundaries of the model's applicability.

4.7. Output File Structure

The developed framework produces three main output files that record the state, temporal evolution, and network topology of the oscillating bandit system. These files are denoted as `state.csv`, `trajectory.csv`, and `graph.json`. Each of them stores complementary information that enables a complete reconstruction and analysis of the agents' behaviour during the simulation.

4.7.1. State File

The `state.csv` file captures the instantaneous configuration of all oscillating agents and their local neighbourhoods at each iteration. It contains the intrinsic properties of each agent, its partial rewards, and several parameters related to the coupling with its neighbors.

Each agent is characterised by its *natural frequency*, which defines its intrinsic oscillatory rate in the absence of external influence. The *selected neighbour index* identifies which neighbouring agent was chosen for coupling during the current iteration. The file also stores four types of rewards that contribute to the overall performance measure of the agent: the *first*, *second*, and *third* partial rewards, and the *communication reward*, the latter being associated with information exchange and cooperative synchronisation effects within the network.

For every neighbour of an agent, several coupling-related parameters are recorded. These include the *incremental discounted reward*, which accumulates past rewards using an exponential discount factor; the *number of selections* and its discounted counterpart, both quantifying how frequently a neighbour has been chosen; the *upper confidence bound*,

which encodes the exploration–exploitation trade-off within the selection policy; the *coupling strength*, representing the intensity of interaction between connected agents; and finally the *selection weight*, which defines the likelihood of selecting a given neighbour in subsequent iterations.

4.7.2. Trajectory File

The `trajectory.csv` file stores the temporal evolution of each oscillating agent over all time steps within an iteration. This file provides dynamic information necessary to assess synchronisation and spectral behaviour.

For every agent, the *iteration trajectory* represents the time series of its oscillation phase, reflecting how it evolves under the influence of coupling with selected partners. The *synchronisation degree* quantifies the local coherence of phase among interacting agents at each time step and serves as a basis for computing the synchronisation-related reward. The *spectral analysis* corresponds to the Fourier spectrum of the coupled trajectory, providing insight into the frequency-domain characteristics of the oscillatory dynamics and supporting the computation of spectral rewards.

4.7.3. Graph File

The `graph.csv` file defines the structural topology of the network of oscillating agents. Each entry corresponds to a single agent and specifies its unique *index* and the list of *neighbors* with which it is directly connected. This representation captures the underlying adjacency relationships that determine the flow of coupling interactions across the network. Together with the `state` and `trajectory` files, it enables the reconstruction of both the static and dynamic aspects of the multi-agent system.

CHAPTER 5

Results

This chapter presents the main results of the proposed framework. It first outlines the exploration guidelines designed to assist in analysing how reward structures and network characteristics influence agent decisions. The exploratory simulations evaluate adaptability and robustness to topology changes, noise, and dynamic environments. Finally, the results across stationary and non-stationary settings are compared, highlighting the impact of algorithmic configurations on synchronization and partner selection performance.

5.1. Exploration Guidelines

Based on the proposed model and partner-selection algorithms, a set of guiding questions was defined to structure the analysis, reflecting theoretical expectations from coupled oscillators, reward functions, and multi-armed bandit behavior.

These exploratory questions are directly aligned with the main research questions presented in Section 1.2, providing a more detailed, simulation-oriented perspective on them. Specifically, they contribute to RQ1 by analyzing how agents autonomously adapt and select partners under varying environmental and structural conditions. They address RQ2 by investigating how different reward components and communication structures affect synchronization and coordinated behavior. Finally, they support RQ3 by evaluating the performance of multiple multi-armed bandit algorithms in the Kuramoto-based framework, assessing their ability to represent and solve the partner selection problem effectively. The following exploratory guidelines serve as a practical framework for systematically testing and interpreting the model's behavior.

5.1.1. Impact of Reward and Network Characteristics on Agent Decisions

5.1.1.1. *Do all components of the reward function contribute significantly to improving the agents choices, and which one proves most decisive?*

Answering to this question is crucial to determine which aspects of the reward function most effectively guide agents in selecting partners, directly informing how to optimize the learning process. All the reward components phase synchronization (PLV), weighted coupling with neighbors, and spectral coherence are expected to contribute positively, but the coupling degree is anticipated to be the most decisive because it strengthens reciprocal interactions and promotes stable synchronization between neighbors.

By identifying the key driver reward function of agent performance, the simulations can focus on the most influential metrics, ensuring faster and more reliable coordination in the system.

5.1.1.2. *Is the partner selection algorithm robust to different types of network topology?*

This question aims to assess how the partner selection algorithm performs across different network topologies, which is crucial for understanding its robustness and adaptability. The algorithm is expected to maintain satisfactory performance in all topologies, but behaviors may vary, for example: in fully connected networks, it is expected that the exploration may take longer due to the larger choice space, though higher rewards are possible; in Moore neighborhoods, reduced partner options should accelerate convergence but may limit maximum synchronization.

Understanding these dynamics helps identify how topology influences learning and guides adjustments to improve the system's efficiency and stability across different network structures.

5.1.1.3. *What will be the behavior of the algorithm with increasing noise?*

Increasing noise poses a significant challenge to synchronization, making it essential to evaluate the algorithm's robustness under such conditions. Performance is expected to progressively decline as noise levels rise, especially for metrics that depend on precise phase alignment. However, the learning mechanisms should help mitigate some of these effects, with balanced exploration allowing agents to adapt their choices and preserve a degree of coordination. Understanding this behavior is key to identifying noise thresholds and refining algorithm parameters to enhance resilience and maintain effective synchronization in uncertain environments.

5.1.2. Adaptability in Stationary and Non-Stationary Environments

5.1.2.1. *Is the algorithm capable of effectively adapting to environmental variations, and what is the role of discount, gamma weight and exploration factors in this process?*

This question evaluates the algorithm's ability to adapt to environmental variations and the role of key parameters: the discount factor (γ_d) in D-UCB, the gamma weight (γ_e) in EXP3, and the exploration factor (ϵ). In changing environments, D-UCB and EXP3 are expected to adapt better than classical UCB: the discount factor γ_d emphasizes recent rewards, allowing outdated choices to be replaced, while the gamma weight γ_e in EXP3 controls the learning rate and probability distribution update, enabling the algorithm to respond effectively to non-stationary conditions. Exploration (ϵ) further allows agents to discover better partners after environmental changes. Prior studies show that discounted UCB variants and adversarial algorithms maintain higher performance in non-stationary scenarios [**28**]. Consequently, these algorithms are expected to adjust effectively to both slow and abrupt changes, sustaining superior synchronization and partner selection.

5.1.2.2. *Does UCB exhibit better performance in environments with stationary rewards, as theoretically expected?*

Analyzing whether UCB performs optimally in stationary environments, which is important for understanding the algorithm's suitability when rewards are stable. UCB is expected to converge rapidly to advantageous partners because constant reward distributions allow reliable accumulation of evidence. As exploration decreases with repeated selections, the algorithm increasingly favors higher-reward partners, resulting in efficient

and stable choices. Classical results confirm that UCB achieves asymptotically optimal regret bounds in stationary scenarios [**26**]. Therefore, UCB is anticipated to outperform adaptive algorithms like D-UCB and EXP3 in stable contexts, since discounting or adversarial adjustments in these algorithms may introduce unnecessary variance and slow convergence. Understanding this behavior helps justify parameter choices and guides comparisons with adaptive methods.

5.1.2.3. *Do D-UCB and EXP3 demonstrate greater adaptability to non-stationary environments compared to traditional UCB?*

Adapting to non-stationary environments is essential for maintaining performance when reward conditions change over time. D-UCB is expected to outperform traditional UCB by "forgetting" outdated experiences and prioritizing recent rewards, allowing agents to adjust more rapidly to gradual environmental changes [**27**, **28**]. Similarly, EXP3.P should perform well in non-stationary or adversarial settings by continuously updating selection probabilities while maintaining uniform exploration. These mechanisms keep agents responsive to evolving partner behaviors. As a result, both D-UCB and EXP3.P are anticipated to achieve higher performance in dynamic scenarios, whereas classical UCB may converge to suboptimal choices due to its assumption of stationary rewards. Understanding these dynamics is key to selecting and tuning algorithms for adaptive partner selection.

5.1.3. Comparison of Choices with Ideal and Direct-Communication Models

5.1.3.1. *To what extent can models without explicit communication approximate those with direct communication between agents?*

Evaluating performance against a direct communication model (Equation 24) helps determine how effectively implicit synchronization signals support coordination. The algorithm is expected to approximate the behavior of systems with explicit communication, as local synchronization acts as an indirect information channel. However, without direct exchanges about internal states or intentions, convergence is likely to be slower and more variable. With sufficiently informative rewards and sustained exploration, the system should still achieve near-comparable results, particularly in progressive phase alignment and the formation of stable, mutually beneficial partnerships.

5.1.3.2. *Do agents consistently select partners whose phases are closest to their own over the iterations?*

Partner selection is expected to increasingly favor agents with similar phases, as synchronized interactions yield higher rewards and are reinforced over time. This should gradually reduce phase deviation, strengthen local synchronization, and lead to the emergence of stable partnership patterns. The speed and consistency of this convergence, however, will depend on factors such as noise, network topology, and environmental non-stationarity. In stable settings, clear convergence toward rhythmically compatible partners is anticipated, while in dynamic environments this consistency may be harder to maintain though still significantly better than random selection.

5.2. Exploratory Simulations Configurations

In order to address the questions raised in Section 5.1, a set of simulations was designed to explore different dimensions of the proposed model. Each group of simulations corresponds to a specific axis of analysis, allowing the impact of factors such as the composition of the reward function, network topology, noise levels, and environmental dynamics to be systematically assessed.

To ensure consistency and comparability across experiments, all simulations share a common baseline configuration, summarized in Table 3. This table defines the parameters that remain fixed throughout the experimental study, including the general simulation setup, partner-selection mechanisms, oscillator dynamics, and the default reward structure. Unless explicitly stated otherwise, all experimental results reported in this chapter rely on this baseline configuration.

TABLE 3. Simulations baseline parameters

Category	**Parameter**	**Values**
General	Pseudo random seed	10 different numbers
	Number iterations	4000
Partner Selection	Type	D-UCB/UCB/EXP3.P
	Coupling strength	0.5
	Exploration factor (UCB)	2.0
	Exploration factor (D-UCB)	0.1
	Discount factor (D-UCB)	0.995
	Gamma weight (EXP3.P)	0.1
	Coupling strength selected	0.3
	Coupling strength selector	0.7
Oscillator	Std. dev. phase observation	0.001
	Std. dev. action	0.005
	Trajectory duration	10
	Trajectory sampling period	0.078125
Reward	Std. dev. Gaussian weight	1.0
	Weight first partial reward	0.0
	Weight second partial reward	1.0
	Weight third partial reward	0.0
	Weight communication reward	0.0

Variations relative to the baseline are introduced in a controlled manner and are explicitly identified within each corresponding subsection. These variations include, for example, changes in reward component weights, network topology, noise intensity, algorithm-specific parameters, or environmental dynamics. Phase change parameters are only considered in the experiments described in Section 5.2.4, as this is the only scenario in which the environment evolves over time and such parameters become relevant.

The results obtained from these experimental configurations are analyzed in Section 5.3, with the objective of identifying performance patterns and validating the theoretical expectations outlined in Section 5.1.

5.2.1. Relative contribution of the reward function components

To evaluate this variable, simulations were carried out in a Moore topology, in order to control the number of neighbors and limit environmental complexity. All the algorithms were compared, testing different reward combinations. Specifically, both the metrics in isolation (r1, r2, and r3) and 50/50 combinations of two of them (r1+r2, r2+r3, and r1+r3) were considered. This experimental design makes it possible to assess not only the effect of each individual component, but also the impact of their interaction on agents' learning.

In total, 180 simulations were conducted, 10 for each reward configuration, whose results allow analysis of the robustness and relative relevance of the metrics that make up the reward function, providing empirical evidence to identify the most critical factors in improving the partner-selection process. The baseline configuration used in these simulations is reported in Table 3, while the parameters that vary in this experiment are summarized in Table 4.

TABLE 4. Variations from the baseline used in the reward simulations

Category	**Parameter**	**Values**
Reward	Weight first partial reward	1.0/0.5/0.0
	Weight second partial reward	1.0/0.5/0.0
	Weight third partial reward	1.0/0.5/0.0

5.2.2. Robustness to different network topologies

To better understand the robustness of the model to different environment topologies, four distinct topologies were considered: *fade, perlin, maze* and *all connected* (presented in Section 4.2). Each of these configurations presents a particular set of challenges. While fade and perlin represent smoother environments, with gradual neighborhood transitions, the maze topology introduces structural restrictions that severely limit interaction options. On the other hand, the fully connected topology gives each agent the possibility of interacting with all others, expanding the decision space and potentially hindering convergence to a stable policy. As in the first set of experiments, UCB, D-UCB and EXP3 algorithms were compared to assess their robustness under different degrees of connectivity.

A total of 120 simulations were carried out, 30 for each topology type, allowing comparative analysis of the algorithms' adaptation to environments with very distinct structural characteristics. The reward values obtained from the optimal configurations identified in Section 5.3.1 were used in these experiments as well as in the subsequent exploratory configurations. The parameters employed in these experiments are listed in Table 3.

5.2.3. Robustness to noise

To evaluate the robustness of the partner-selection algorithms, different experimental conditions with noise levels of 1 and 0.001 were considered. These values were chosen

to represent two contrasting scenarios: one with high perturbations, where synchronization becomes challenging, and another with minimal disturbances, providing a near-ideal environment for coordination. Two types of noise were introduced in the simulations: action noise, which perturbs the perceptions of the decisions made by other agents when selecting partners, and phase noise, which directly affects the oscillators phase dynamics, disrupting synchronization signals. By keeping the partner selection algorithm and reward function constant across both settings, the experiment isolates the specific effect of noise, ensuring that any observed variations in performance can be directly attributed to the intensity of these perturbations.

In total, 30 simulations were conducted for each noise condition, allowing comparison of model performance in noisier and less noisy scenarios, for all the algorithms. The analysis of the results provides empirical evidence of the system's robustness and of its ability to maintain satisfactory levels of synchronization and reward even in adverse environments. The baseline parameter configuration is reported in Table 3, while the parameters varied in the noise robustness experiments are summarized in Table 5.

TABLE 5. Variations from the baseline used in the noise simulations

Category	**Parameter**	**Values**
General	Number iterations	2000
Oscillator	Std. dev. phase observation	0.001/1.0
	Std. dev. action	0.001/1.0

5.2.4. Adaptation to dynamic environments

The experimental design considered different combinations of γ_d, ϵ, and γ_e values to examine how the rate of forgetting past rewards (controlled by γ_d), the level of continuous exploration (controlled by ϵ), and the learning rate in EXP3 (controlled by γ_e) influence the algorithm's responsiveness. This setup enabled comparison between scenarios in which agents emphasize recent neighbor behavior versus scenarios where past information retains high weight, as well as between environments that encourage more or less exploration of new partnership options. In total, 30 simulations were conducted for each parameter combination, allowing identification of adaptation patterns and evaluation of the conditions under which UCB, D-UCB, and EXP3 maintain high performance under gradual environmental changes. The parameterization is identical between altered and constant scenarios, with the external alteration parameter independent of the parameter file. This analysis provides evidence that discounting, exploration, and gamma-weight mechanisms are critical for algorithm robustness in dynamic environments. The baseline parameter configuration is reported in Table 3, while the parameters specific to the dynamic environment simulations, including phase change settings, are summarized in Table 6. For phase change parameters, the first value in each pair corresponds to abrupt changes, and the second value corresponds to gradual drift.

TABLE 6. Variations from the baseline used in the dynamic environments simulations

Category	Parameter	Values
General	Number iterations	20000
	Discount factor(D-UCB)	0.995/0.95
	Exploration factor(UCB)	2.0/1.0/0.5/0.2
	Exploration factor (D-UCB)	1.0/0.1
	Gamma weight (EXP3.P)	0.9/0.5/0.1/0.001
Phase Changes	Replace Sim	{True, False}
	Gradual Drift Sim	{False, True}
	Drift Base	{0, 0.0005}
	Drift Std.	{0, 0.0005}
	Sigma Rw.	{0, 0.0005}
	Wmin, Wmax	$\{0, 0.05\}, \{0, 6.0\}$

The questions presented in Sections 5.1.2.2 and 5.1.2.3 will be addressed according to the results obtained from the simulations presented in Section 5.2.4, since these employed both stationary and non-stationary environments as well as the UCB, D-UCB and EXP3.P algorithms.

5.2.5. Comparison with direct communication

For this purpose, two distinct scenarios were considered: an environment in which agents follow the implicit model, without message exchange and relying solely on observed synchronization rewards, and a reference environment in which agents have access to an "optimal" reward derived from a model with explicit communication, enabling more informed decisions. Both scenarios were tested under all the algorithms, ensuring that the comparison focuses exclusively on the impact of communication while keeping all other parameters constant. It should also be noted that no environmental changes were applied over the iterations in these simulations. Ten simulations were carried out for each configuration, allowing observation of how the absence of explicit communication constrains the learning process and to what extent the emergent behavior approaches ideal performance. The analysis of these results provides relevant evidence for understanding the limitations and potential of implicit communication in rhythmic partner-selection contexts. The baseline parameter configuration is reported in Table 3, while the parameters specific to the communication simulations are summarized in Table 7.

TABLE 7. Variations from the baseline used in the simulations with and without communication

Category	Parameter	Values
Reward	Weight second partial reward	1.0/0.0
	Weight communication reward	0.0/1.0

Finally, all simulations will be used to draw conclusions regarding Section 5.1.3.2, so no specific simulations are associated exclusively with that question.

5.3. Exploratory Simulations Results

In this section, the results of the exploratory simulations are presented, with the primary goal of identifying the most suitable parameter configurations for the final experiments. These simulations focus on the fundamental building blocks of the system, addressing the questions outlined in Section 5.1.1 and Section 5.1.3.1, which explore how reward components, network topology, and noise influence agent behavior and decision-making. The analysis is based primarily on plots of *average reward*, which represent the mean value of the rewards obtained by all agents across iterations, since the rewards are normalized the values are contained in the interval [0,1]. These plots provide a compact measure of overall performance, allowing the comparison of different algorithms and configurations in terms of how effectively they guide agents toward successful partner selections and synchronization.

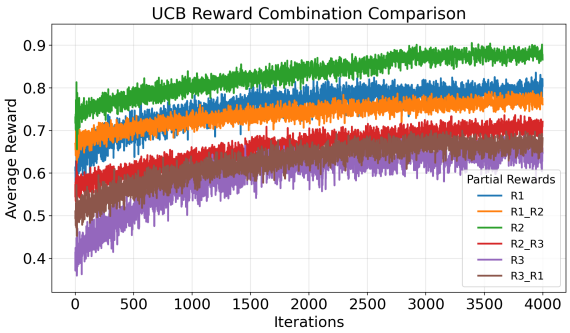
5.3.1. Relative contribution of the reward function components

The results in this section compare different combinations of partial rewards for the three algorithms: UCB (Figure 5.1a), D-UCB (Figure 5.1b), and EXP3.P (Figure 5.1c). In all cases, the second partial reward, R2, consistently produces the highest average reward, confirming it as the most informative component. Adding R1 or R3 does not improve performance and often introduces additional noise in the learning process.

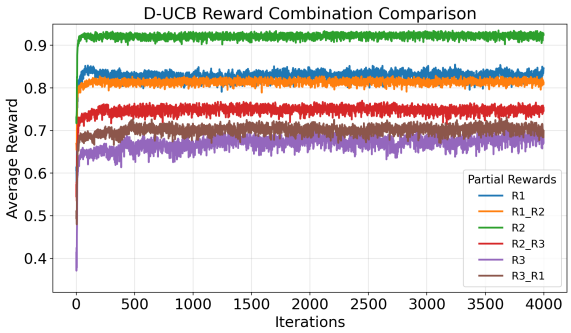
Figure 5.1d provides a concrete example of the outcome of a single simulation in terms of agent choices. In this figure, each ellipse represents an agent and is labeled with its natural frequency. The color of the ellipses encodes these frequencies, following a scale in which darker shades correspond to lower frequencies and lighter blue tones indicate higher frequencies. The arrows represent the agents selected by each agent at the end of the learning process, with arrow colors indicating the frequency of selection: greener arrows correspond to agents that were selected more frequently.

This visualization highlights that agents tend to select partners with similar natural frequencies, as expected from the underlying synchronization objective. The concentration of greener arrows between agents with close frequency values indicates a higher rate of correct selections. Importantly, this behavior is most clearly observed when using R2, which yields both the highest average rewards and the largest number of correct selections. These qualitative observations are consistent with the quantitative results, and are further reinforced by the selection frequency plots presented in Appendix A.1, which confirm the predominance of R2 across simulations.

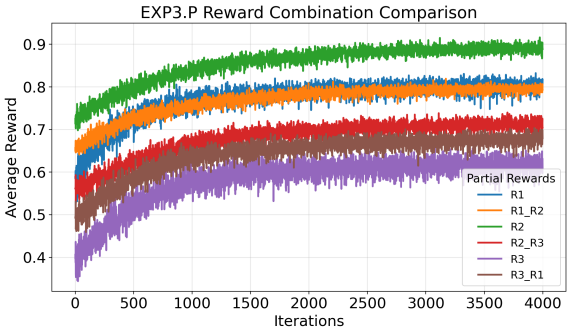
Since high average rewards obtained with R2 correspond directly to correct partner selections, subsequent sections omit detailed selection visualizations and rely exclusively on average reward as a reliable indicator of decision quality. This provides a clear answer to the first research question (Section 5.1.1.1): using R2 alone constitutes the most effective reward configuration for this study, while R1 and R3 offer limited or even detrimental contributions.



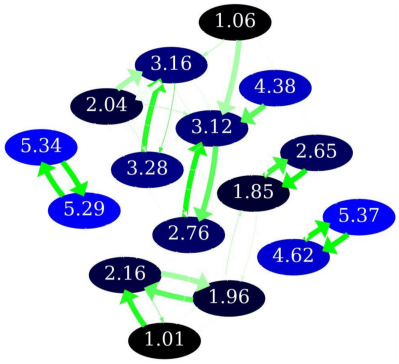
(A) UCB



(B) D-UCB



(C) EXP3.P



(D) Agents Selections with Partial Reward 2

FIGURE 5.1. Comparison of the reward combinations for each algorithm under the same simulation conditions and selections from Partial Reward 2.

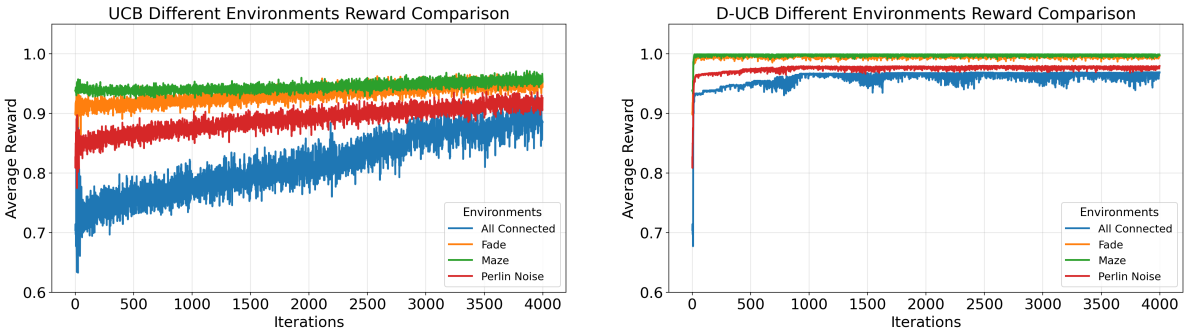
5.3.2. Robustness to different network topologies

For these simulations, the three algorithms were tested across different network topologies to evaluate their adaptability. Results show that all algorithms can ultimately reach high performance, though convergence speed varies with the environment's complexity. In simpler topologies, such as Maze and Fade, where similar-frequency agents are located nearby, convergence is faster as optimal partners are easier to identify. In contrast, more complex topologies, like Perlin Noise and All Connected, slow convergence due to higher variability and less clear interaction patterns.

These findings directly address the question in Section 5.1.1.2, demonstrating that the algorithms are robust to different network structures. While topology affects the rate of convergence, it does not prevent the algorithms from achieving strong long-term synchronization and reward optimization. The plots that present the results are in Figure 5.2:

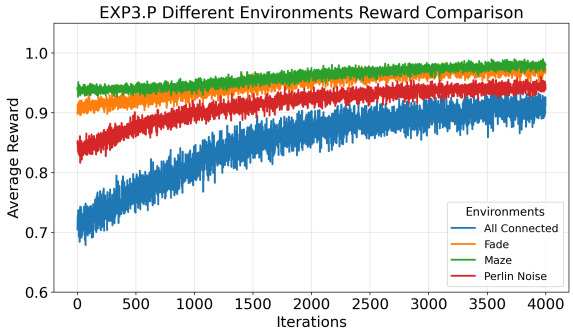
5.3.3. Robustness to noise

In this section the simulations present the results of testing different noise configurations, where noise refers to the Gaussian perturbations added to the agents' phase observations and action signals, controlled by their respective standard deviations. These



(A) UCB

(B) D-UCB



(C) EXP3.P

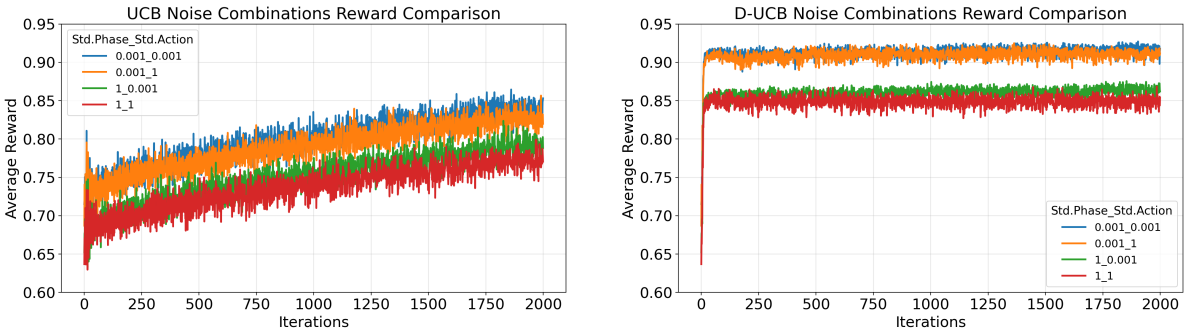
FIGURE 5.2. Comparison of the different algorithms for each environment under the same simulation conditions.

experiments were carried out for each algorithm in environments without frequency alterations. Starting with UCB (Figure 5.3a), the lowest noise setting (0.001, 0.001) achieves the highest final average reward, converging steadily, while configurations with phase deviation set to 1 consistently perform worst, showing reduced stability and slower convergence. In the case of D-UCB (Figure 5.3b), performance remains strong across all settings, but results again drop when the phase deviation is high, although the algorithm still maintains higher stability and reward compared to UCB under the same conditions. Lastly, EXP3.P (Figure 5.3c) shows a similar pattern, with low noise producing the best outcomes and high phase deviation leading to increased variability and lower final rewards, though overall the algorithm continues to reach higher values than UCB.

These results directly address the Question in 5.1.1.3, confirming that the increase of noise negatively affects performance, with phase deviation emerging as the decisive factor that disrupts partner selection and delays convergence. At the same time, the higher rewards achieved by D-UCB and EXP3.P demonstrate their greater resilience to noisy environments, indicating that both algorithms are better suited than UCB when uncertainty in phase observation is significant.

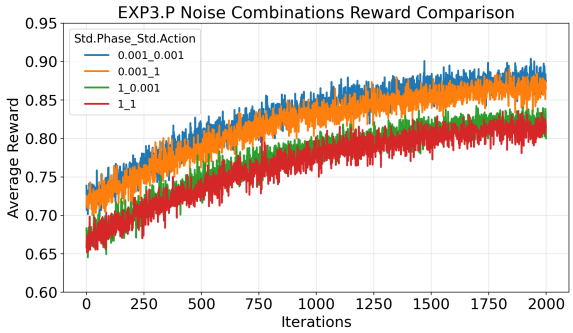
5.3.4. Adaptation in dynamic environments

5.3.4.1. *Stationary Environment*



(A) UCB

(B) D-UCB



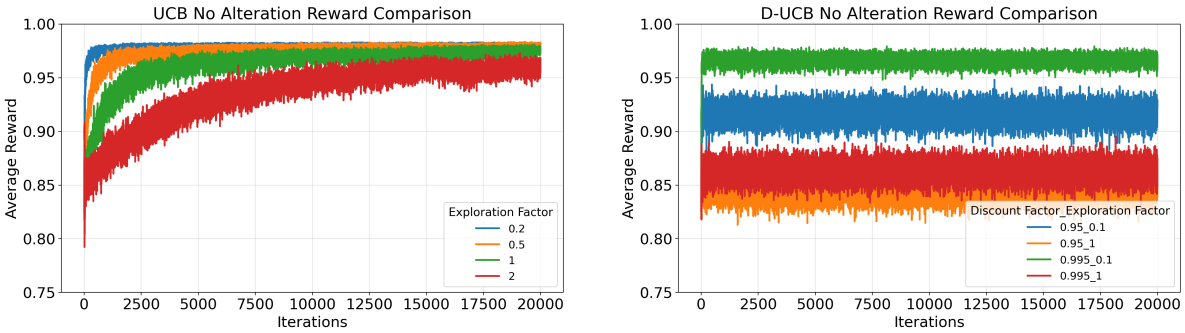
(C) EXP3.P

FIGURE 5.3. Comparison of the different noise values for each algorithm under the same simulation conditions.

In this section the simulations present the results of testing different configurations for each algorithm in environments without frequency alterations. Starting with UCB (Figure 5.4a), the exploration factor of 0.2 achieves the highest final average reward, converging quickly and maintaining stable performance. As the exploration factor increases, performance decreases, with 2.0 being the worst configuration, as it leads to excessive exploration and slower convergence. In the case of D-UCB (Figure 5.4b), the combination of discount factor 0.995 and exploration factor 0.1 delivers the best results, remaining stable throughout the simulation with a consistently high average reward. In contrast, the configuration with discount factor 0.95 and exploration factor 1.0 performs worst, showing lower stability and a reduced final reward. Lastly, EXP3.P (Figure 5.4c), a gamma weight of 0.1 produces the best overall performance, providing a good balance between exploration and exploitation. Higher gamma values, particularly 0.9, result in poorer performance, with the algorithm converging faster to a lower average reward.

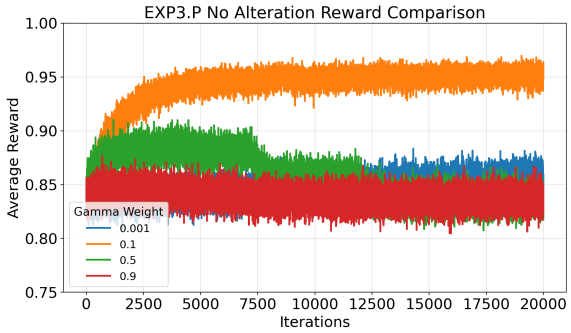
5.3.4.2. *Abrupt Changes*

In this section, the simulations present the results of testing different configurations for each algorithm in environments with abrupt alterations. These alterations occur at predefined moments of the simulation, specifically at half of the run (10,000 iterations) and at three-quarters (15,000 iterations), where the natural frequencies of the agents are reassigned to new values. It is normal for the maximum average reward to decrease after these changes, since the environment is reassigned randomly and the agents' natural



(A) UCB

(B) D-UCB



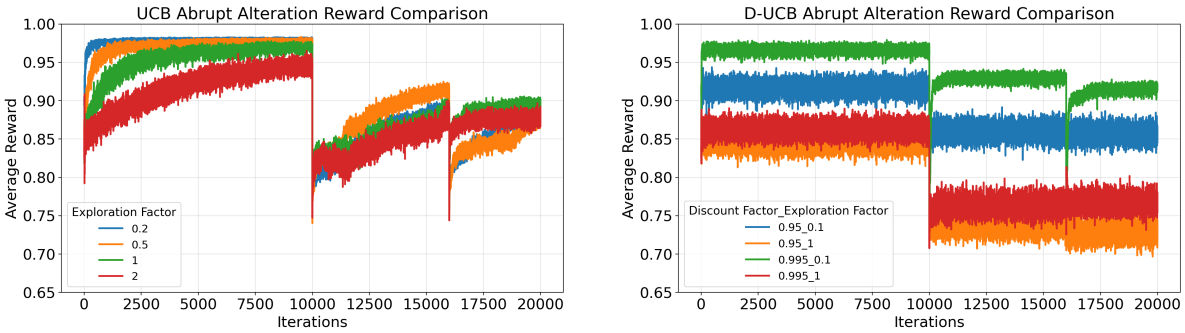
(C) EXP3.P

FIGURE 5.4. Comparison of the rewards in environments without frequency alterations for each algorithm under the same simulation conditions.

frequencies become more dispersed compared to the initial configuration, making the rewards lower. Starting with UCB (Figure 5.5a), exploration factors of 0.2 and 0.5 achieve the highest final average rewards, with both configurations reacting well to the abrupt changes and maintaining strong performance. In contrast, an exploration factor of 2.0 performs worst, as excessive exploration hinders convergence and reduces overall rewards. For D-UCB (Figure 5.5b), the combination of discount factor 0.995 and exploration factor 0.1 delivers the best results, showing fast adaptation to sudden alterations and consistently high average rewards. On the other hand, the configuration with discount factor 0.95 and exploration factor 1.0 performs worst, quickly converging to a poor reward and failing to adapt after the environment changes. Similarly, EXP3.P (Figure 5.5c) shows that gamma value 0.1 yields the best performance, balancing exploration and exploitation while reacting to abrupt changes. Interestingly, the effect of the gamma weight on the average reward is not linear: the value 0.1 produces faster recoveries, while both higher and lower values (e.g., 0.001) lead to slower adaptation. Higher gamma values, particularly 0.9, result in poorer outcomes, with the algorithm converging prematurely to a lower average reward and failing to adjust after alterations.

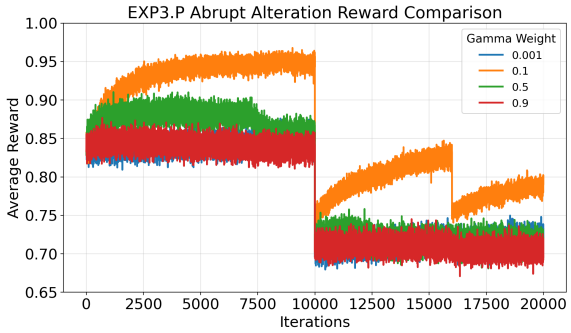
5.3.4.3. *Gradual Changes*

This section presents the results of testing different configurations for each algorithm in environments with gradual alterations. For UCB (Figure 5.6a), an exploration factor of 0.2 achieves the best performance, reaching high rewards and maintaining them over



(A) UCB

(B) D-UCB

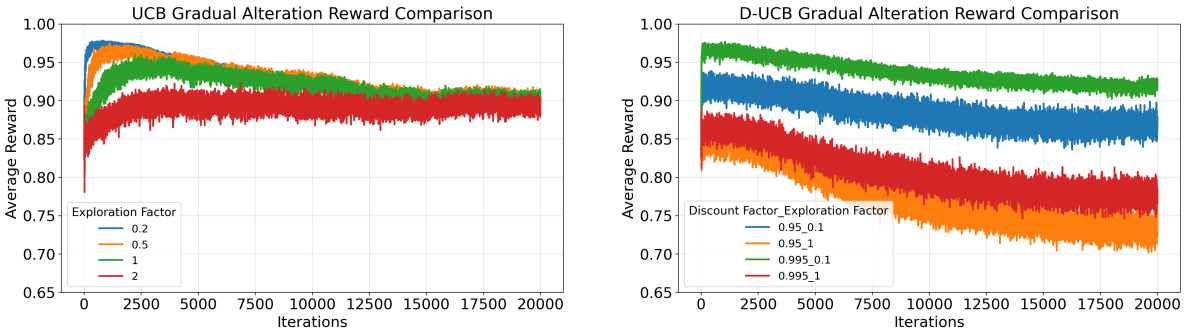


(C) EXP3.P

FIGURE 5.5. Comparison of the rewards in environments with abrupt frequency alterations for each algorithm under the same simulation conditions.

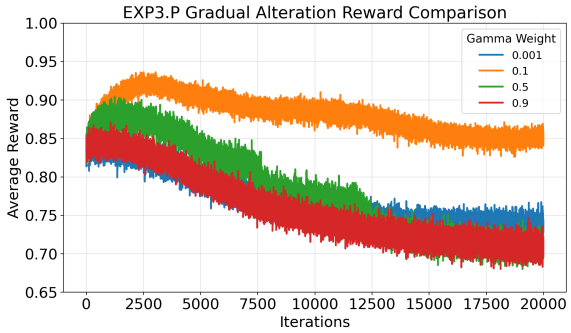
a longer period, while an exploration factor of 2.0 performs worst, failing to reach high rewards initially and exhibiting the worst adaptation as the environment gradually alters. In the case of D-UCB (Figure 5.6b), the combination of discount factor 0.995 and exploration factor 0.1 delivers the best results, showing the smallest drop in rewards over time. Conversely, a discount factor of 0.95 with exploration factor 1.0 performs worst, starting with lower rewards and experiencing a significant decline throughout the simulation. Finally, EXP3.P (Figure 5.6c) shows that a gamma value of 0.1 provides the best overall performance, with higher initial rewards and a relatively stable reward curve that declines minimally under gradual changes. The worst configuration, gamma 0.9, starts with the lowest initial reward and exhibits the largest drop over time.

In summary, across all tested scenarios the best performing configurations consistently emerged as UCB with an exploration factor of 0.2, D-UCB with a discount factor of 0.995 and exploration factor of 0.1, and EXP3.P with a gamma weight of 0.1. These parameter settings effectively balance exploration and exploitation, providing high average rewards while adapting efficiently to both sudden and gradual changes in the environment. Furthermore, these findings directly address the question raised in 5.1.2.1, confirming that discounting, gamma weight, and exploration factors play a decisive role in enabling re-adaptation. The discount factor in D-UCB proved essential for forgetting outdated experiences and prioritizing recent ones, the gamma weight in EXP3.P guided the algorithm's responsiveness to environmental variability, and the exploration factor in UCB



(A) UCB

(B) D-UCB



(C) EXP3.P

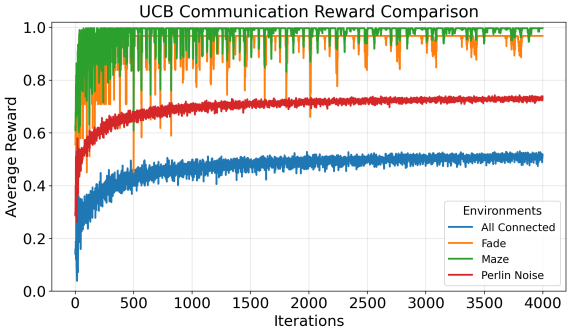
FIGURE 5.6. Comparison of the rewards in environments with gradual frequency alterations for each algorithm under the same simulation conditions.

determined the trade-off between stability and adaptability. Together, the results validate that properly tuned parameters are crucial for maintaining synchronization and ensuring effective partner selection in dynamic environments.

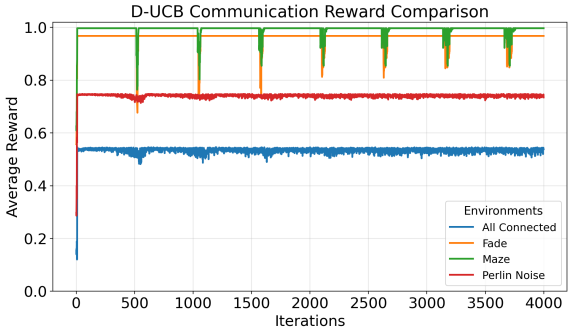
5.3.5. Comparison with direct communication

In this section the simulations evaluate the communication reward as a reference to assess whether the algorithms are performing well, since this reward can be regarded as a standard against which the results from 5.3.2 may be compared (Figure 5.7). For UCB (Figure 5.7a) and D-UCB (Figure 5.7b), the outcomes in the Maze and Fade environments were very close to this standard, showing that both algorithms were able to learn effective partner selection policies that nearly matched the idealized reference. In the case of EXP3.P (Figure 5.7c), the results were not only close to the standard but at certain points even surpassed it, demonstrating that the stochastic exploration strategy can provide additional benefits in less complex settings. However, in the more demanding environments of Perlin Noise and All Connected, comparison with the standard was not possible, as the communication reward consistently underperformed relative to the baseline without communication. This is particularly relevant because these environments are characterized by greater structural complexity and more varied frequency distributions, where the communication method falls short by only accounting for the agents original frequencies and not the changes occurring over time. As a result, the method fails to adapt, leading to poor

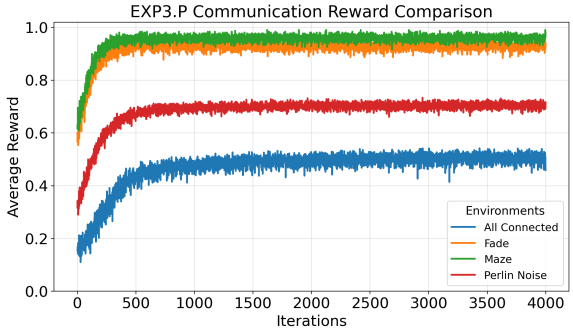
performance and highlighting the advantage of the independence of the MARL system and the type of reward formulation used in the remaining experiments. This limitation becomes even more pronounced in complex environments, where neighbors may start with very different natural frequencies. In such cases, the subsequent adjustments driven by the Kuramoto dynamics amplify these differences, further degrading the performance of communication-based strategies compared to simpler environments in which agents begin with similar or identical frequencies. These findings provide a clear answer to the Question 5.1.3.1: in simpler environments, the rewards obtained without communication can reach values very close to those of the communication-based standard, while in more complex environments the communication reward falls short, confirming the advantage of independent learners in adapting to dynamic conditions.



(A) UCB



(B) D-UCB



(C) EXP3.P

FIGURE 5.7. Comparison of the rewards in environments with direct communication for each algorithm under the same simulation conditions.

5.4. Final Results

In this section, the most relevant simulations are revisited to provide a direct comparison of the three algorithms and assess their performance across different conditions. These final experiments build on the exploratory phase, now relying on the optimal configurations previously identified to ensure a fair and consistent evaluation. The objective is to understand which algorithm performs best in each type of environment, highlighting their respective strengths and limitations. The analysis addresses the exploratory questions outlined in Sections 5.1.2 and 5.1.3, focusing on adaptability in both stationary

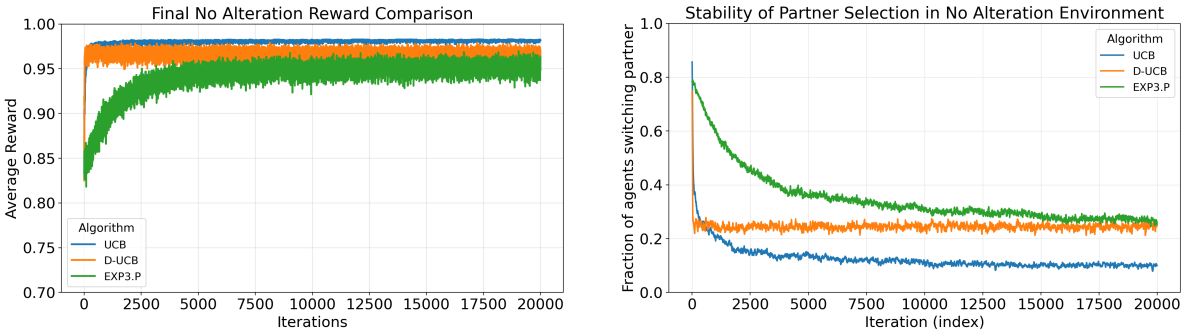
and non-stationary scenarios, as well as on performance relative to idealized and direct-communication models.

The evaluation relies mainly on two types of plots. The first are *average reward plots*, which were also used in the exploratory phase. These plots represent the mean reward obtained by all agents across iterations, offering a compact measure of global performance and allowing a direct comparison between algorithms. The second are *partner selection change plots*, which track how often agents modify their chosen partners over time. A stable system is reflected by fewer changes after an initial learning phase, while frequent changes indicate instability or excessive exploration. This plot takes values in the interval [0, 1], a value of 0 means no agents changed partner between iterations $t-1$ and t, while a value of 1 means all agents changed partner. We then apply a 25 iteration moving average to this per iteration fraction, so higher values indicate more frequent switching over the recent window. Together, these plots are complementary with the average rewards reflecting how effective the learning process is, while switching rates reveal the stability of choices and may help in understanding convergence trends. Additional visualizations are provided in Appendix B.1, including detailed probability evolution and partner selection frequency plots for individual agents. These complementary figures illustrate the temporal dynamics of partner preferences and further highlight the stability patterns observed during training.

5.4.1. Stationary Environment Comparison

The results in Figure 5.8a provide empirical evidence regarding the behavior of the algorithms in stationary environments, as anticipated in Section 5.1.2.2. UCB achieves the highest average rewards, converging rapidly and stabilizing near optimal performance. This aligns with theoretical expectations, as UCB exploits stable reward distributions to accumulate reliable evidence and reduce unnecessary exploration, consistently favoring the best partners. D-UCB also performs competitively, though slightly below UCB in final stability, since the discounting mechanism useful in dynamic scenarios introduces a small loss of efficiency in stationary settings. EXP3.P, in contrast, converges much more slowly and stabilizes at lower reward levels, as its adversarial exploration strategy adds unnecessary variance when rewards are stable.

The second plot in Figure 5.8b further highlights these differences. UCB quickly minimizes partner changes, showing that agents converge early to consistent and reliable selections. D-UCB retains more variability, reflecting the effect of discounting, while EXP3.P maintains the highest number of changes throughout, confirming that its exploration-heavy design prevents agents from settling on stable partnerships. Overall, these results confirm that UCB is the most effective choice for stationary environments, combining both the highest reward levels and the greatest stability in partner selection.



(A) Average Rewards (B) Partner Selection Changes

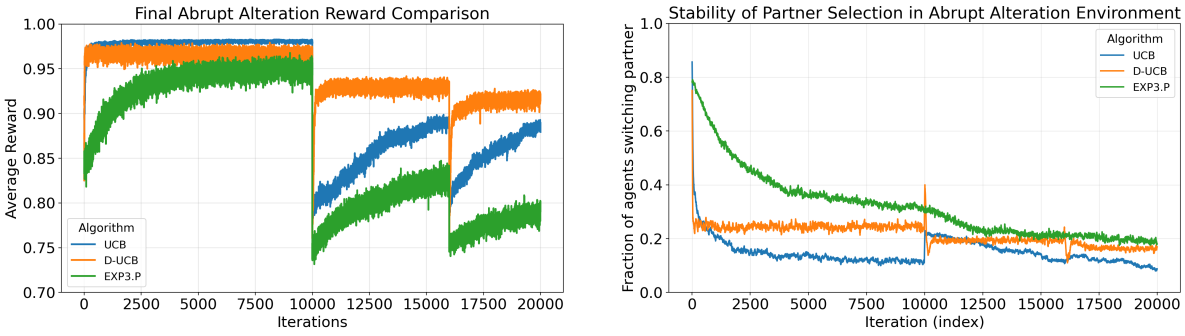
FIGURE 5.8. Comparison of the performance in stationary environment for each algorithm under the same simulation conditions.

5.4.2. Abrupt Alteration Environment Comparison

The results in Figure 5.9a provide clear insights into the adaptability of the algorithms when confronted with abrupt environmental changes, as outlined in Section 5.1.2.3. UCB, which assumes stationarity, initially performs well but reacts poorly to sudden alterations, showing sharp drops followed by slow recoveries that never fully regain previous levels. This reflects the algorithm's reliance on accumulated historical evidence, which becomes a liability once reward distributions shift. D-UCB maintains higher performance overall as expected, since its discounting mechanism rapidly discards outdated information and prioritizes recent rewards, enabling quicker recovery and more stable convergence after each change.

EXP3.P also adapts and its behavior aligns with theoretical expectations. Designed for adversarial environments where rewards are independent of previous actions, its exploration-heavy strategy results in high variance and prevents the consolidation of stable partner selections. As a consequence, recoveries after abrupt changes are slower and noisier, and the algorithm fails to reach the reward levels achieved by UCB. This outcome is consistent with its underlying assumption and it continually spreads probability mass across all partners, EXP3.P maintains robustness in adversarial contexts but performs suboptimally in structured, interactive systems where stability and rapid re-synchronization are more beneficial.

The second plot in Figure 5.9b reinforces these conclusions by illustrating the stability of partner selection. UCB exhibits sudden increases in partner changes after each abrupt shift, followed by slow stabilization, consistent with its reward trajectory. D-UCB, in contrast, adapts more smoothly, with only moderate fluctuations after changes and faster convergence to stable selections. EXP3.P shows the highest variability across the simulation, with partner changes remaining frequent throughout, reflecting its constant exploration strategy. Altogether, these results highlight that D-UCB combines strong adaptability with stable partner choices, making it the most effective algorithm in environments characterized by abrupt alterations.



(A) Average Rewards

(B) Partner Selection Changes

FIGURE 5.9. Comparison of the performance in environments with abrupt changes for each algorithm under the same simulation conditions.

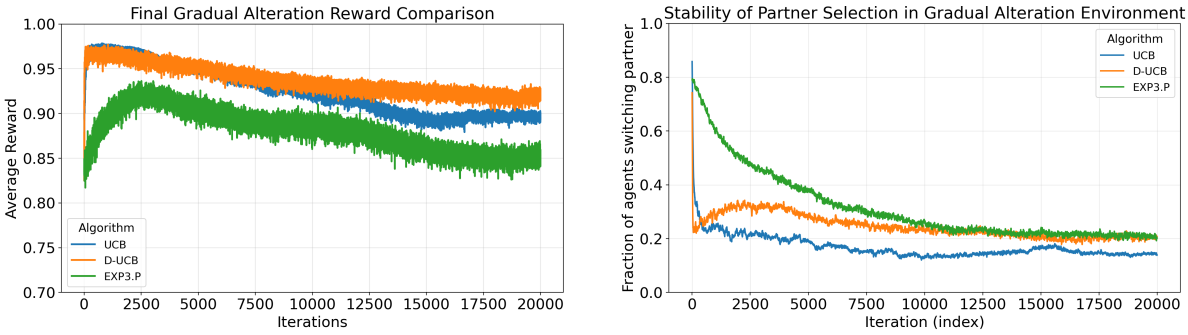
5.4.3. Gradual Alteration Environment Comparison

The results in Figure 5.10a illustrate the behavior of the algorithms under gradual environmental alterations, providing further insight into their adaptability as discussed in Section 5.1.2.3. UCB struggles in this non-stationary context, showing a steady decline in performance as outdated experiences accumulate and prevent effective adjustment to slowly changing rewards. D-UCB adapts more effectively, maintaining higher rewards throughout the simulation and confirming the benefit of discounting past information in dynamic environments.

EXP3.P performs consistently with its theoretical properties but reveals the limitations of adversarial approaches in interactive, non-oblivious systems. Designed for environments where rewards are independent of past actions, EXP3.P faces two distinct challenges in this setting: first, gradual changes in the environment make each reward partially dependent on previous outcomes; second, the rewards themselves are shaped by mutual synchronization among agents, meaning that an agent's own choices influence the future rewards of others and vice versa. These interdependencies violate the independence assumption underlying the algorithm, amplifying its instability. Consequently, its continuous exploration and probability redistribution hinder the consolidation of stable partner selections, leading to steadily decreasing performance well below both UCB and D-UCB. This outcome reinforces that while EXP3.P is robust in truly adversarial conditions, it is poorly suited for coupled rhythmic environments where reward dependencies emerge from dynamic agent interactions.

The second plot in Figure 5.10b further clarifies these dynamics by showing the stability of partner selection. UCB exhibits relatively few partner changes, but this reflects its inability to adapt rather than successful stability, since its reward levels decline over time. D-UCB displays moderate variability in selections, which enables it to track the gradual shifts and sustain higher rewards. By contrast, EXP3.P maintains the highest number of changes throughout, reflecting its constant exploration, yet this instability does not translate into better adaptation, as its rewards remain the lowest. Overall, D-UCB

emerges as the most effective strategy in gradually changing environments, combining adaptability with stable and efficient partner choices.



(A) Average Rewards (B) Partner Selection Changes

FIGURE 5.10. Comparison of the performance in environments with gradual changes for each algorithm under the same simulation conditions.

5.4.4. Close phase partner selection consistency

The overall results provide a comprehensive answer to the investigation question raised in 5.1.3.2, demonstrating that agents consistently tend to select partners with phases close to their own, reinforcing synchronization and yielding higher rewards across all experimental conditions. The reward formulation proved effective in guiding this behavior, with the combination of its components ensuring that interactions based on rhythmic compatibility were reinforced over time. This tendency remained present even when the structure of the interaction network was altered, showing that the model is flexible enough to sustain stable patterns of synchronization under different topological constraints.

When noise was introduced, performance naturally decreased, with phase deviation emerging as the most critical factor negatively affecting stability. High levels of phase noise increased variability in partner selection and delayed convergence, yet the agents were still able to perform well, with D-UCB in particular maintaining resilience and stability across a wide range of conditions.

In dynamic environments, the adaptability of the algorithms became more apparent. UCB, while effective under stable conditions, struggled to recover from abrupt changes and gradually lost performance under continuous alterations. D-UCB, by contrast, adapted more efficiently, discounting outdated information and quickly re-establishing stable synchronization after changes. EXP3.P, while theoretically suited for adversarial non-stationary settings, exhibited high variance and limited stability in this interactive context. Its continuous exploration, appropriate for environments with independent reward sequences, became detrimental here due to the interdependent nature of agent synchronization, preventing consistent partner choices and resulting in lower overall rewards.

The experiments with communication rewards further highlight the robustness of the system. In simpler environments, the performance of the independent learners nearly matched, and in some cases even surpassed, the communication-based benchmark. In

more complex environments, however, the communication reward proved insufficient, while the independent learners continued to achieve strong results. This indicates that the reinforcement dynamics alone are capable of sustaining rhythmic compatibility without the need for explicit information exchange, even under challenging conditions.

Taken together, the findings confirm that agents performed well across all studied environments, consistently converging towards phase-compatible partners. While UCB demonstrates strong performance in stationary contexts and EXP3.P shows potential in simple settings, the results consistently identify D-UCB as the most effective and reliable approach. By combining high stability with strong adaptability to noise and dynamic environments, D-UCB stands out as the most promising algorithm for a definitive application of this system.

5.5. Discussion

The experimental results reveal clear distinctions among the three partner selection strategies. UCB achieved high performance in stationary or slowly changing environments, confirming its effectiveness when reward distributions remain relatively stable. However, its reliance on long-term averages limited responsiveness under abrupt changes, as accumulated knowledge quickly became outdated. The discounted variant, D-UCB, demonstrated consistent adaptability across conditions, successfully tracking gradual environmental drift through its forgetting mechanism. EXP3.P, designed for adversarial settings, exhibited persistent oscillations and lower reward stability, reflecting its assumption of an oblivious environment where rewards are independent of past actions an assumption not fully met in the coupled, interactive system studied.

In relation to RQ1, the results show that agents can autonomously learn to select partners and achieve synchronization through reward-based feedback alone, without explicit communication channels (Section 5.1.3.1). Agents progressively favored partners whose phases were closest to their own (Section 5.1.3.2), demonstrating that implicit coordination is sufficient for adaptive multi-agent synchronization.

Regarding RQ2, not all reward components contributed equally to performance, with the coherence-based term proving most decisive for promoting stable synchronization (Sections 3.4.3.2 and 5.1.1.1). The model also showed robustness across network topologies (Section 5.1.1.2) and resilience under increasing noise (Section 5.1.1.3), confirming that appropriately designed reward functions effectively encode synchronization cues.

For RQ3, the Kuramoto–bandit framework proved effective for partner selection. UCB excelled in stationary environments (Section 5.1.2.2), whereas D-UCB adapted more effectively to dynamic and non-stationary conditions (Sections 5.1.2.1 and 5.1.2.3), sustaining synchronization even after abrupt changes (Section 5.2.4). EXP3.P's slower convergence and persistent fluctuations highlighted the limitations of purely adversarial exploration in interdependent systems. Together, these results underscore the importance of temporal discounting for robust adaptive coordination in multi-agent rhythmic systems.

CHAPTER 6

Conclusions and Future Work

6.1. Conclusion

This dissertation explored rhythmic partner selection in multi-agent systems by integrating Kuramoto oscillators with multi-armed bandit algorithms. The motivation stemmed from the need to understand how artificial agents can autonomously coordinate their actions through synchronization, a mechanism widely observed in biological and social systems. By combining oscillatory dynamics with learning-based decision-making, the study aimed to model how agents form and maintain rhythmic cooperation without explicit communication.

The results demonstrated how agents can autonomously learn to synchronize and select partners through reward-driven adaptation. UCB provided stable performance in stationary environments, efficiently exploiting consistent partners once optimal connections were established. D-UCB extended this capability by incorporating temporal discounting, which allowed agents to adapt to gradual or abrupt environmental changes while maintaining coordination. In contrast, EXP3.P, although robust in adversarial scenarios, showed persistent instability in this interactive context, highlighting the limitations of algorithms that assume independence between actions and future rewards.

The experiments also revealed that coordination quality depends strongly on the reward formulation. The second reward component representing local phase synchronization proved to be the most informative signal for guiding cooperation, whereas other components contributed less consistently. These findings show that effective partner selection emerges when learning mechanisms are directly aligned with the oscillatory dynamics of the system.

Overall, the study provides clear answers to the research questions posed in Section 1.2. In relation to RQ1, the results show that agents are capable of autonomously selecting partners and achieving synchronization solely through implicit, reward-based feedback, without the need for explicit communication. Regarding RQ2, coordination was found to critically depend on the design of the reward function, with the coherence-based term being the most decisive for promoting stable synchronization. With respect to RQ3, among the explored partner selection strategies, UCB excelled in stationary environments, D-UCB provided robust adaptability to dynamic and non-stationary settings, and EXP3.P was less suitable in interdependent systems due to persistent fluctuations. These findings further highlight that temporal discounting is a key mechanism for sustaining coordination in changing conditions.

In conclusion, synchronization-based coordination can be effectively modeled as a sequential decision-making process. The integration of Kuramoto oscillators and multi-armed bandit algorithms provides a quantitative and interpretable approach to evaluating learning strategies in rhythmic environments. These results confirm that autonomous synchronization can emerge from simple, decentralized adaptation rules and that temporal weighting of rewards is crucial for maintaining stability and cooperation among agents.

6.2. Limitations and Challenges

While the proposed framework effectively models rhythmic partner selection, several limitations should be acknowledged. The agents were simplified as Kuramoto oscillators, capturing phase synchronization but omitting perceptual, spatial, and communicative aspects that significantly influence coordination in real-world systems. In natural and human systems, synchronization emerges from complex interactions involving sensory perception, motion dynamics, and feedback delays, which are not captured by phase-only models. Estimating equivalent oscillatory parameters in biological or human agents is particularly challenging, as rhythmic patterns arise from perceptual–motor interactions rather than directly measurable variables. In contrast, such parameters can be more easily controlled or predefined in robotic or mechanical implementations, which may limit the direct transferability of the results to more complex, embodied agents.

The simulations also relied on fixed parameters, such as learning rates, temporal discounting factors, and network connectivity. These choices constrain the range of dynamics observed and may influence both the speed of adaptation and the stability of synchronization. Different parameter values could lead to qualitatively different behaviors, and the fixed settings used here may not fully capture the diversity of possible adaptation strategies or the robustness of coordination under alternative conditions. Exploring a broader parameter space could reveal additional insights about system scalability, resilience, and sensitivity to environmental changes.

In addition, the reward formulation imposed further constraints. The framework relied on fixed synchronization metrics, primarily coherence-based measures, which focus on local phase alignment but may overlook other factors influencing coordination, such as amplitude interactions, frequency variability, or context-dependent preferences in partner selection. As a result, the model may fail to fully capture more nuanced or multi-dimensional drivers of cooperation observed in real-world systems.

Finally, the experiments were conducted on controlled and relatively small populations, limiting conclusions about scalability and robustness in larger, more heterogeneous environments. Larger populations or more complex network structures could introduce emergent behaviors, collective oscillatory patterns, or synchronization breakdowns not observed in the current simulations. Despite these constraints, the framework provides a strong foundation for future research on adaptive and rhythmic multi-agent coordination, offering a baseline from which more complex and realistic systems can be explored.

6.3. Future Work

Future developments could build upon this framework along several complementary directions. One key avenue is to extend the learning mechanisms employed, exploring hybrid strategies that integrate stochastic exploration with adaptive discounting principles. Such approaches could improve responsiveness in non-stationary environments, allowing agents to adjust more efficiently to changing interaction patterns. Mechanisms like periodic resetting or adaptive memory decay could also be investigated to accelerate recovery after abrupt environmental changes, helping to maintain coordination even under highly dynamic conditions. These enhancements would directly address limitations related to fixed learning parameters and the inability of the current framework to quickly adapt to sudden perturbations.

Another important direction concerns the refinement and parameterization of the reward formulation. Adjusting the relative weights of the components $\zeta_1 + \zeta_2 + \zeta_3$ could allow the model to be tuned toward different coordination objectives, such as emphasizing phase stability, responsiveness to changes, or alignment of natural frequencies. From a modeling perspective, these parameters could be optimized using data-driven techniques such as maximum likelihood estimation or Bayesian optimization. Aligning simulated behavior with empirical observations from human or robotic synchronization tasks would improve the ecological validity of the model and facilitate practical applications. Furthermore, exploring multi-objective reward functions could help investigate trade-offs between synchronization quality, energy efficiency, and adaptability in more complex coordination scenarios.

Expanding the experimental scope would further enhance the generality and applicability of the findings. Future studies should investigate larger populations, varied network topologies, and heterogeneous oscillators to assess scalability, emergent collective patterns, and robustness in more realistic environments. Incorporating heterogeneity in agent parameters, such as natural frequencies or learning rates, could reveal richer dynamics, including the emergence of leader–follower relationships, clustering, or synchronization breakdowns. Social simulations provide another promising avenue for exploring how rhythmic coordination operates in human groups or mixed human–robot teams. By simulating interactions in crowds, ensembles, or cooperative tasks, researchers could examine the emergence of leadership, role specialization, and adaptive recovery in dynamic social settings. Such studies could inform practical applications ranging from modeling crowd dynamics in music and dance to improving coordination in team sports or cooperative robotics, and ultimately to designing socially aware robots capable of adaptively synchronizing with multiple humans in collaborative environments. Integrating these social simulations with empirical validation would provide critical insights into the mechanisms of adaptive rhythmic coordination, helping to bridge the gap between computational models and real-world collective behavior.

References

[1] S. H. Strogatz, *Sync: How order emerges from chaos in the universe, nature, and daily life.* Hachette UK, 2012.

[2] A. Baratta, A. Cimino, M. G. Gnoni, and F. Longo, "Human-robot collaboration in industry 4.0: a literature review," in *Procedia Computer Science*. Elsevier, 2022, pp. 1887–1895.

[3] Y. Cao, Z. Wen, B. Kveton, and Y. Xie, "Nearly optimal adaptive procedure with change detection for piecewise-stationary bandit," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 418–427.

[4] P. Santana and J. Moura, "A bayesian multi-armed bandit algorithm for dynamic end-to-end routing in sdn-based networks with piecewise-stationary rewards," *Algorithms*, vol. 16, no. 5, 2023. [Online]. Available: https://www.mdpi.com/1999-4893/16/5/233

[5] M. Wilson and T. P. Wilson, "An oscillator model of the timing of turn-taking," *Psychonomic bulletin & review*, vol. 12, pp. 957–968, 2005.

[6] J. D. Lomas, A. Lin, S. Dikker, D. Forster, M. L. Lupetti, G. Huisman, J. Habekost, C. Beardow, P. Pandey, N. Ahmad *et al.*, "Resonance as a design strategy for ai and social robots," *Frontiers in neurorobotics*, vol. 16, p. 850489, 2022.

[7] S. S. Wiltermuth and C. Heath, "Synchrony and cooperation," *Psychological science*, vol. 20, no. 1, pp. 1–5, 2009.

[8] D. G. Rand and M. A. Nowak, "Human cooperation," *Trends in cognitive sciences*, vol. 17, no. 8, pp. 413–425, 2013.

[9] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[10] T. Iqbal and L. D. Riek, "Temporal anticipation and adaptation methods for fluent human-robot teaming," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3736–3743.

[11] G. Skantze, "Turn-taking in conversational systems and human-robot interaction: a review," *Computer Speech & Language*, vol. 67, p. 101178, 2021.

[12] F. Fu, C. Hauert, M. A. Nowak, and L. Wang, "Reputation-based partner choice promotes cooperation in social networks," *Physical Review E*, vol. 78, no. 2, p. 026117, 2008.

[13] F. Correia, S. Petisca, P. Alves-Oliveira, T. Ribeiro, F. S. Melo, and A. Paiva, ""i choose... you!" membership preferences in human–robot teams," *Autonomous Robots*, vol. 43, pp. 359–373, 2019.

[14] N. Anastassacos, S. Hailes, and M. Musolesi, "Partner selection for the emergence of cooperation in multi-agent systems using reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7047–7054.

[15] A. Dahiya, A. M. Aroyo, K. Dautenhahn, and S. L. Smith, "A survey of multi-agent human–robot interaction systems," *Robotics and Autonomous Systems*, vol. 161, p. 104335, 2023.

[16] S. K. Hasnain, G. Mostafaoui, and P. Gaussier, "A synchrony-based perspective for partner selection and attentional mechanism in human-robot interaction," *Paladyn*, vol. 3, pp. 156–171, 2012.

[17] L. Aubin, G. Mostafaoui, C. Amiel, H. Serré, D. Capdevielle, M. H. de Menibus, J. Boiché, R. Schmidt, S. Raffard, and L. Marin, "Study of coordination between patients with schizophrenia and socially assistive robot during physical activity," *International Journal of Social Robotics*, pp. 1–16, 2021.

[18] B. Heath, R. Hill, and F. Ciarallo, "A survey of agent-based modeling practices (january 1998 to july 2008)," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 4, p. 9, 2009.

[19] F. Bianchi and F. Squazzoni, "Agent-based models in sociology," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 7, no. 4, pp. 284–306, 2015.

[20] R. H. Bemthuis and S. Lazarova-Molnar, "Towards integrating process mining with agent-based modeling and simulation: State of the art and outlook," *Expert Systems with Applications*, p. 127571, 2025.

[21] T. Guo, X. Wang, H. Zhang, Y. Liu, and Z. Li, "Large language model based multi-agents: A survey of progress and challenges," *arXiv preprint arXiv:2402.01680*, 2024. [Online]. Available: https://arxiv.org/abs/2402.01680

[22] A. Slivkins *et al.*, "Introduction to multi-armed bandits," *Foundations and Trends® in Machine Learning*, vol. 12, no. 1-2, pp. 1–286, 2019.

[23] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[24] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[25] D. Bouneffouf, I. Rish, and C. Aggarwal, "Survey on applications of multi-armed and contextual bandits," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.

[26] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, pp. 235–256, 2002.

[27] L. Kocsis and C. Szepesvári, "Discounted ucb," in *2nd PASCAL Challenges Workshop*, vol. 2, 2006, pp. 51–134.

[28] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *Algorithmic Learning Theory: 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011. Proceedings 22*. Springer, 2011, pp. 174–188.

[29] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM journal on computing*, vol. 32, no. 1, pp. 48–77, 2002.

[30] J. Lin and S. Moothedath, "Distributed multi-task learning for stochastic bandits with context distribution and stage-wise constraints," *IEEE Transactions on Signal and Information Processing over Networks*, 2025, early access.

[31] F. A. Rodrigues, T. K. D. Peron, P. Ji, and J. Kurths, "The kuramoto model in complex networks," *Physics Reports*, vol. 610, pp. 1–98, 2016.

[32] J. A. Acebrón, L. L. Bonilla, C. J. P. Vicente, F. Ritort, and R. Spigler, "The kuramoto model: A simple paradigm for synchronization phenomena," *Reviews of modern physics*, vol. 77, no. 1, p. 137, 2005.

[33] Y. Kuramoto, "Self-entrainment of a population of coupled non-linear oscillators," in *International Symposium on Mathematical Problems in Theoretical Physics: January 23–29, 1975, Kyoto University, Kyoto/Japan*. Springer, 1975, pp. 420–422.

[34] A. Pluchino, S. Boccaletti, V. Latora, and A. Rapisarda, "Opinion dynamics and synchronization in a network of scientific collaborations," *Physica A: Statistical Mechanics and its Applications*, vol. 372, no. 2, pp. 316–325, 2006.

[35] A. Kalloniatis and T. McLennan-Smith, "Calibrating coupling for collaboration with kuramoto," *arXiv preprint arXiv:2311.15228*, 2023.

[36] J.-P. Lachaux, E. Rodriguez, J. Martinerie, and F. J. Varela, "Measuring phase synchrony in brain signals," *Human brain mapping*, vol. 8, no. 4, pp. 194–208, 1999.

[37] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the 11th International Conference on Machine Learning (ICML)*, 1994, pp. 157–163.

[38] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[39] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, pp. 750–797, 2019.

[40] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6379–6390.

[41] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016, pp. 2244–2252.

[42] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, J. Pineau, and T. Tsiligkaridis, "Tarmac: Targeted multi-agent communication," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 1538–1546.

[43] S. K. Hasnain, P. Gaussier, and G. Mostafaoui, ""'synchrony' as a way to choose an interacting partner"," in *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, 2012, [Online]. [Online]. Available: http://www.etis.ensea.fr/neurocyber/Videos/synchro/

[44] M. Belkaid, C. Lesueur-Grand, G. Mostafaoui, N. Cuperlier, and P. Gaussier, "Learning sensorimotor navigation using synchrony-based partner selection," in *Proceedings of the ACM International Conference*. ACM, 2016.

[45] Y. Xu, S. C. F. Hendrikse, J. Treur, and P. H. M. P. Roelofsma, "Adaptive network modeling for joint action and memory recall for elderly by detecting interpersonal synchrony," *Cognitive Systems Research*, vol. 88, 2024.

[46] A. list, "Virtual partners improve synchronization in human–machine trios," *Cognitive Science*, 2025, adaptive virtual partners synchronize with human rhythmic action. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/39898831

[47] C. Fourie *et al.*, "Joint action, adaptation, and entrainment in human-robot interaction," in *ACM/IEEE International Conference on Human-Robot Interaction*. IEEE Computer Society, 2022, pp. 1250–1253.

[48] Y. Bouteraa, J. Ghommam, N. Derbel, and G. Poisson, "Adaptive synchronization control of multi-robot teams: Cooperative and coordinated schemes," *International Journal of Modelling, Identification and Control*, 2010.

[49] D. Stoeva, A. Kriegler, and M. Gelautz, "Review paper: Body movement mirroring and synchrony in human-robot interaction," *ACM Transactions on Human-Robot Interaction*, 2024.

[50] M. Agarwal, V. Aggarwal, and K. Azizzadenesheli, "Multi-agent multi-armed bandits with limited communication," *Journal of Machine Learning Research*, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-0138.html

[51] D. Chen, B.-Y. Zhang, X.-H. Li, and Z.-T. Liu, "Human-robot emotional interaction model based on reinforcement learning," *Journal of Electronics & Information Technology*, 2024.

[52] E. Schneiders, C. Fourie, S. Celestin, J. Shah, and M. Jung, "Understanding entrainment in human groups: Optimising human-robot collaboration from lessons learned during human-human collaboration," in *Conference on Human Factors in Computing Systems*. ACM, 2024.

[53] P. Li, Z. Liu, W. Pang, and J. Cao, "Semantic collaboration: A collaborative approach for multi-agent systems based on semantic communication," in *Proceedings of the 2024 5th International Conference on Computing, Networks and Internet of Things (CNIOT)*, 2024.

[54] T. Sakamoto, A. Sudo, and Y. Takeuchi, "Investigation of model for initial phase of communication: Analysis of human interaction by robot," *ACM Transactions on Human-Robot Interaction*, vol. 10, no. 2, 2021.

[55] S. C. F. Hendrikse, S. Kluiver, J. Treur, T. F. Wilderjans, S. Dikker, and S. L. Koole, "How virtual agents can learn to synchronize: An adaptive joint decision-making model of psychotherapy," *Cognitive Systems Research*, vol. 79, pp. 138–155, 2023.

[56] M. Dhanda, B. A. Rogers, S. Hall, E. Dekoninck, and V. Dhokia, "Reviewing human-robot collaboration in manufacturing: Opportunities and challenges in the context of industry 5.0," *Robotics and Computer-Integrated Manufacturing*, vol. 93, p. 102937, 2025.

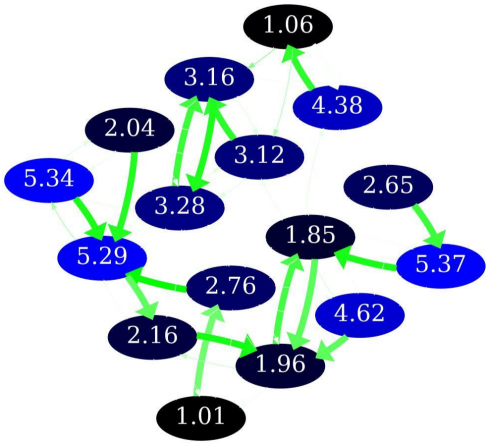
[57] K. Perlin, "An image synthesizer," *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

APPENDIX A

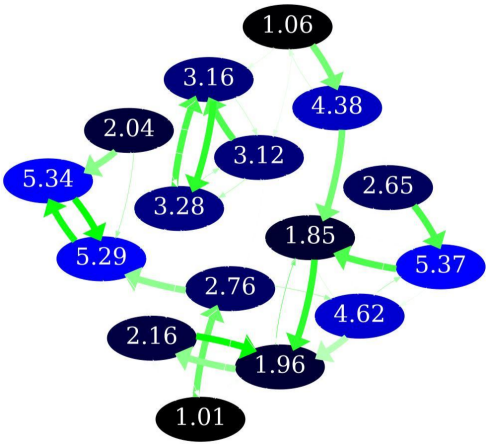
Additional Figures

A.1. Agents' Selections for Remaining Reward Combinations

These figures illustrate the agents' selections at the final iteration for the remaining five reward combinations. Compared with the configuration in Section 5.3.1, the results confirm superior performance, producing more stable and effective partner selections.

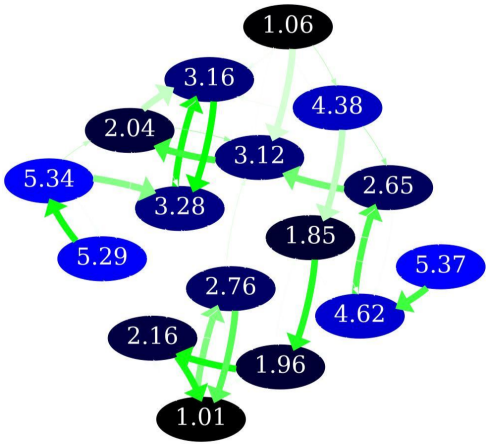


(A) Selections with Partial Reward 1

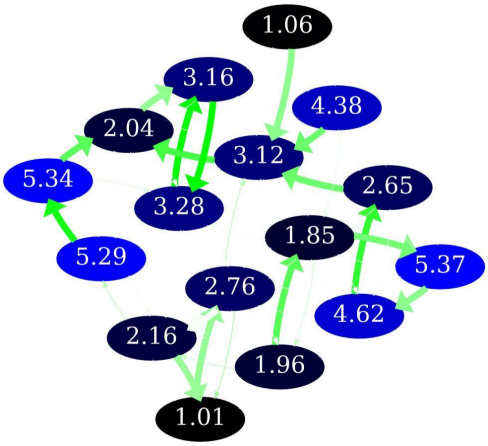


(B) Selections with Partial Rewards 1 and 2

FIGURE A.1. Agent selections for the first two partial reward configurations.



(A) Selections with Partial Rewards 2 and 3



(B) Selections with Partial Reward 3

FIGURE A.2. Agent selections for the next two partial reward configurations.

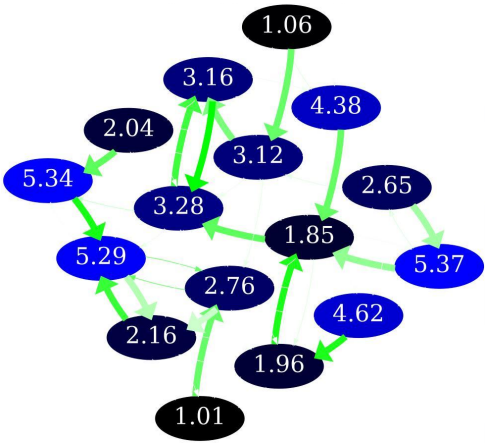


FIGURE A.3. Selections with Partial Rewards 1 and 3.

APPENDIX B

Additional Plots

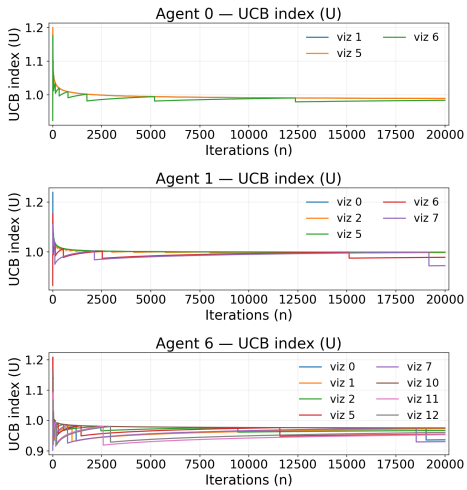
B.1. In-Depth Partner Selection Plots

This appendix presents additional visualizations of the agents' partner selection dynamics across the three simulation conditions (*no alteration*, *abrupt alteration*, and *gradual alteration*). Each set of plots includes the evolution of selection-related variables (left) and the corresponding partner selection frequency heatmaps (right) for selected agents. For the UCB and D-UCB algorithms, the line plots represent the evolution of the upper confidence bounds (*UCB values*) associated with each neighbour over time, while for the EXP3.P algorithm they depict the evolution of selection probabilities. The agents shown were chosen to represent the full range of possible configurations in the Moore neighbourhood, covering distinct positions within the grid and different numbers of available neighbors.

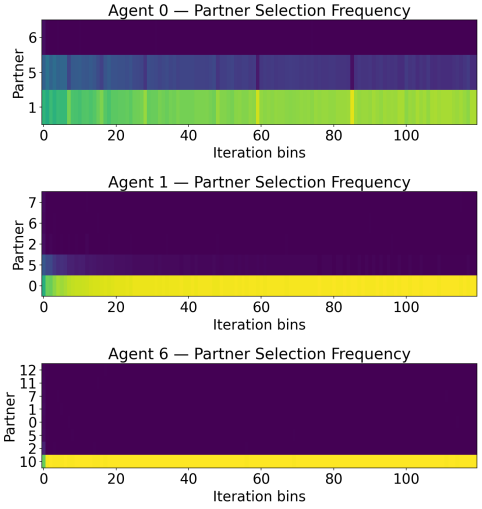
In the heatmaps, colors close to yellow indicate a high frequency of partner selections, whereas colors approaching purple correspond to low selection frequencies. This visual scale provides an intuitive representation of the agents' preference dynamics over time, highlighting dominant and less frequently chosen neighbors.

All results correspond to a single simulation run (not averaged), ensuring that the plots reflect the raw dynamics of one complete execution. All algorithms share the same initial neighbour configuration, allowing direct visual comparison of their behavioural evolution under identical starting conditions. These configurations are the same as those used in the plots presented in Chapter 5.4, ensuring consistency and enabling a direct visual comparison between the main results and the detailed dynamics shown here.

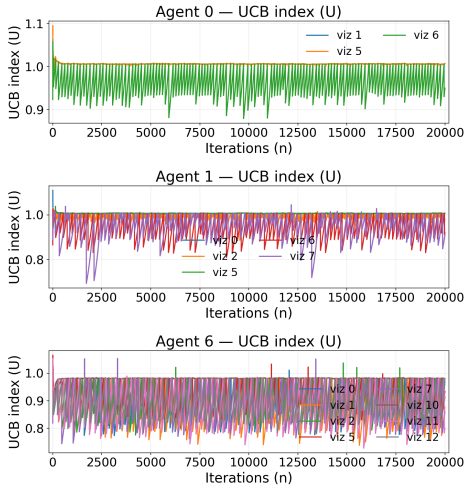
An important observation can be made in these plots, particularly in Figures B.3b and B.3f, where both the UCB and EXP3.P algorithms show agents converging toward similar partner choices. However, since EXP3.P exhibits higher exploration, it disperses its phase by making it closer to non-optimal agents frequently. Consequently, its phase never aligns as closely with that of the best partner, which results in a lower reward compared with UCB, that converges and explores less making its phase closer to the optimal partner, in Figure 5.10a.



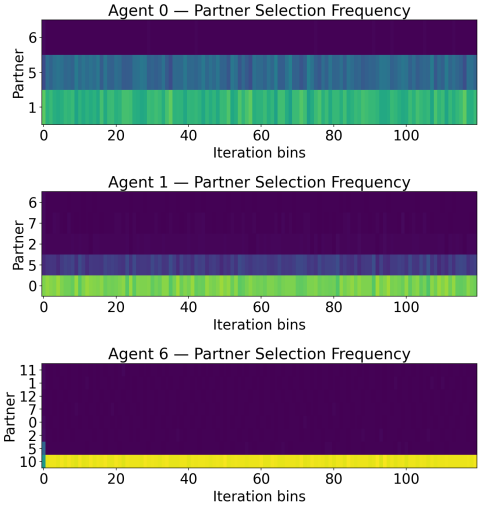
(A) UCB — Selection probabilities.



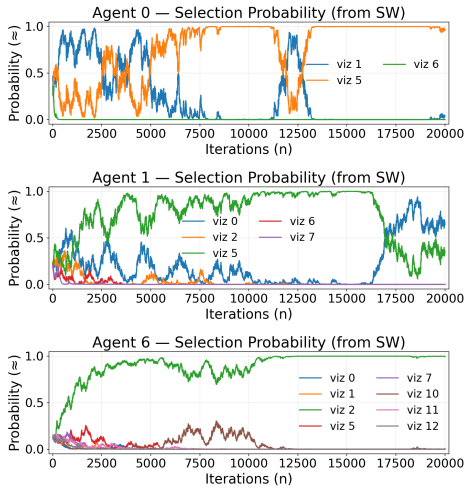
(B) UCB — Selection heatmap.



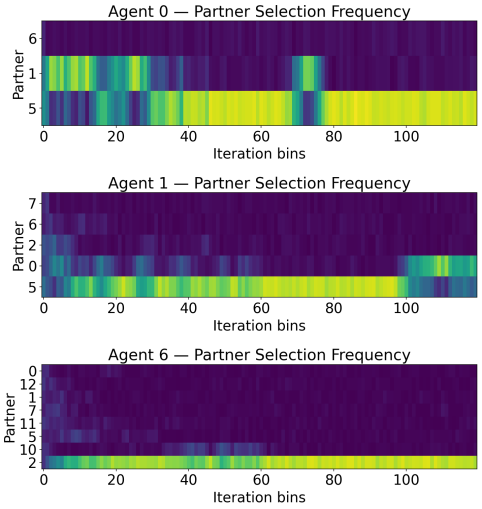
(C) D-UCB — Selection probabilities.



(D) D-UCB — Selection heatmap.

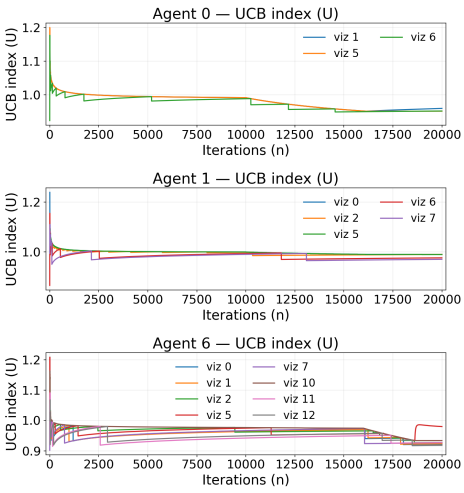


(E) EXP3.P — Selection probabilities.

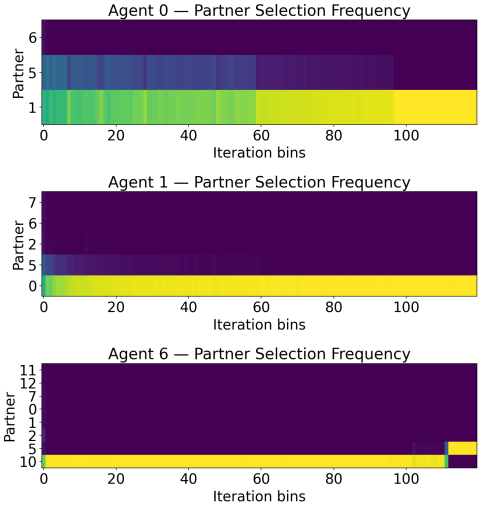


(F) EXP3.P — Selection heatmap.

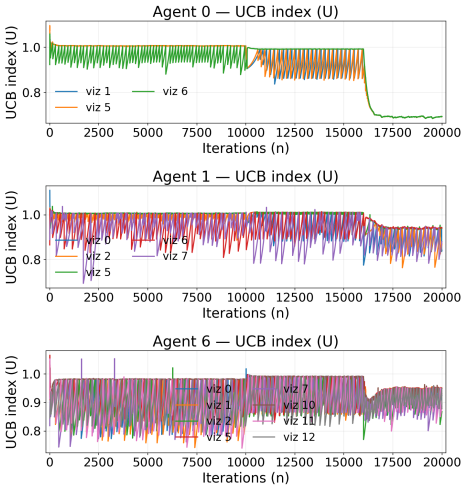
FIGURE B.1. Agents' partner selection dynamics under the *no alteration* condition for different algorithms.



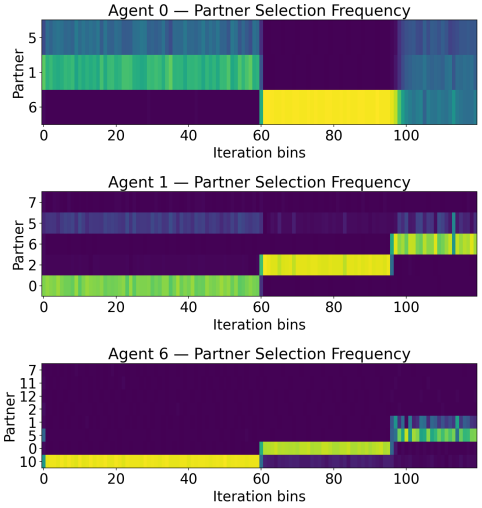
(A) UCB — Selection probabilities.



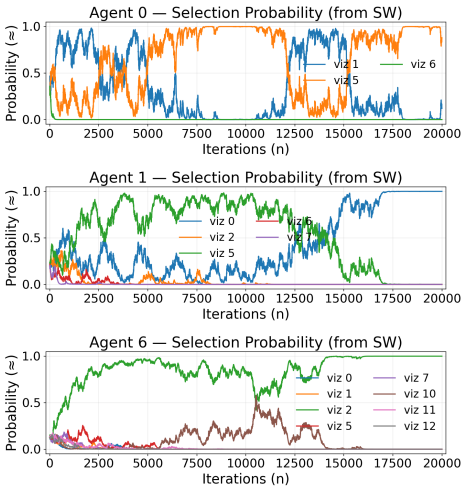
(B) UCB — Selection heatmap.



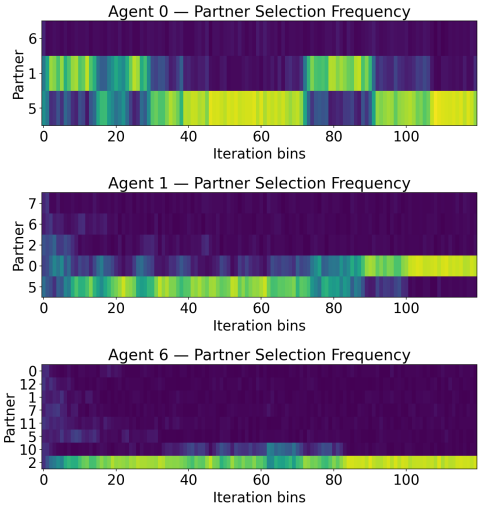
(C) D-UCB — Selection probabilities.



(D) D-UCB — Selection heatmap.

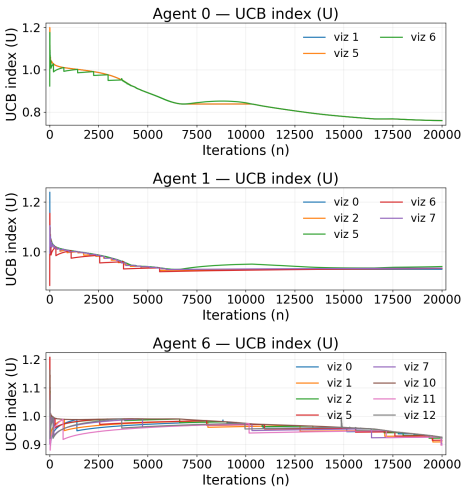


(E) EXP3.P — Selection probabilities.

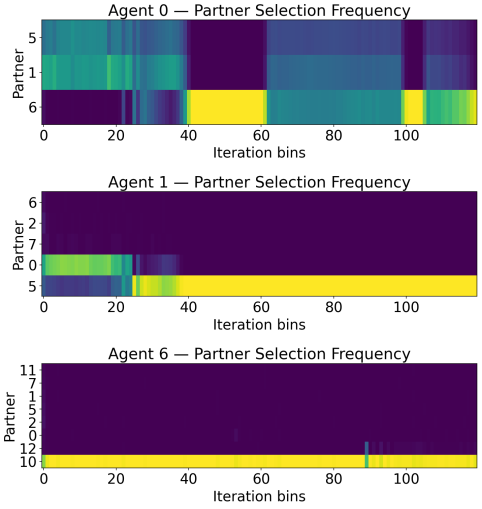


(F) EXP3.P — Selection heatmap.

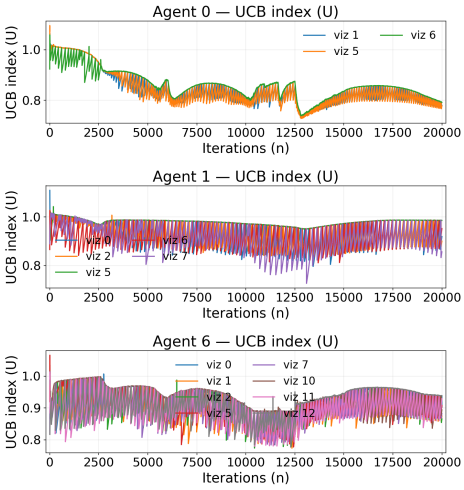
FIGURE B.2. Agents' partner selection dynamics under the *abrupt alteration* condition for different algorithms.



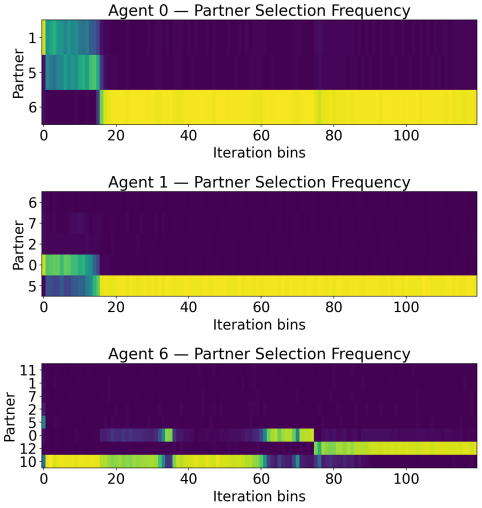
(A) UCB — Selection probabilities.



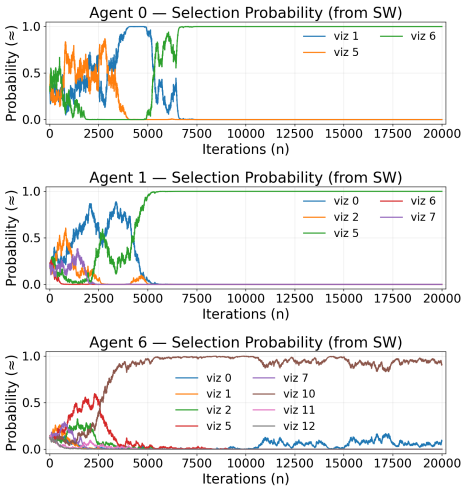
(B) UCB — Selection heatmap.



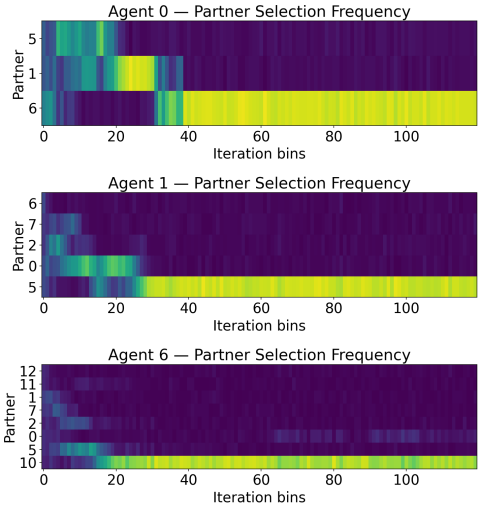
(C) D-UCB — Selection probabilities.



(D) D-UCB — Selection heatmap.



(E) EXP3.P — Selection probabilities.



(F) EXP3.P — Selection heatmap.

FIGURE B.3. Agents' partner selection dynamics under the *gradual alteration* condition for different algorithms.