2023

Multi-agent Learning For Game-theoretical Problems

Kshitija Taywade
University of Kentucky, kshitija.taywade@uky.edu
Author ORCID Identifier:
https://orcid.org/0009-0005-1760-7495
Digital Object Identifier: https://doi.org/10.13023/etd.2023.117

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Kshitija Taywade, Student

Dr. Judy Goldsmith, Major Professor

Dr. Simone Silvestri, Director of Graduate Studies

Multi-agent Learning for Game-theoretical Problems

DISSERTATION

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the College of Engineering at the University of Kentucky

By
Kshitija Taywade
Lexington, Kentucky

Director: Dr. Judy Goldsmith and Dr. Brent Harrison
Professor of Computer Science
Lexington, Kentucky 2023

ABSTRACT OF DISSERTATION

Multi-agent Learning for Game-theoretical Problems

Multi-agent systems are prevalent in the real world in various domains. In many multi-agent systems, interaction among agents is inevitable, and cooperation in some form is needed among agents to deal with the task at hand. We model the type of multi-agent systems where autonomous agents inhabit an environment with no global control or global knowledge, decentralized in the true sense. In particular, we consider game-theoretical problems such as the hedonic coalition formation games, matching problems, and Cournot games. We propose novel decentralized learning and multi-agent reinforcement learning approaches to train agents in learning behaviors and adapting to the environments. We use game-theoretic evaluation criteria such as optimality, stability, and resulting equilibria.

KEYWORDS: Reinforcement Learning, Multi-agent Learning, Decentralized Learning, Multi-armed Bandits, Matching, Cournot Games, Hedonic Coalition Formation Games

Author's signature: Kshitija Taywade

Date: April 27, 2023

Multi-agent Learning for Game-theoretical Problems

By
Kshitija Taywade

Director of Dissertation: Dr. Judy Goldsmith and Dr. Brent Harrison

Director of Graduate Studies: Dr. Simone Silvestri

Date: April 27, 2023

Dedicated to Ms. Manorama Tank, my late grandmother, the source of eternal warmth and strength for me.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my Ph.D. advisors, Dr. Judy Goldsmith and Dr. Brent Harrison. Their immense support helped me pursue and make progress in my research path. They believed in my abilities and supported me in pursuing the research topics of my interest. I am grateful for their patience and constructive feedback on my research work. Their moral support and understanding kept me motivated throughout my Ph.D. journey.

When I first came to the University of Kentucky, I was thrilled to meet Dr. Judy Goldsmith while taking her class. Her genuine interest in uplifting me in my studies and her efforts for inclusion helped me to become a better student and to believe in myself. I am thankful to her for becoming my mentor. She has greatly contributed not only to my academic progress but also to my personal growth. She is an inspiration and a role model to me as a strong woman and caring mentor.

I had the pleasure of collaborating with Dr. Adib Bagh, one of the committee members. I am extremely grateful for his guidance and suggestions on the major part of my Ph.D. thesis. I am thankful for his crucial feedback and moral support.

Lastly, many thanks to all the committee members for being part of my committee.

TABLE OF CONTENTS

Acknowledgments iii

Table of Contents iv

List of Figures v

List of Tables viii

Chapter 1 Introduction 1

Chapter 2 Decentralized Hedonic Coalition Formation 4
- 2.1 Related Work 5
- 2.2 Preliminaries 5
- 2.3 Method 6
- 2.4 Experiments 9
- 2.5 Results 10
- 2.6 Summary 11

Chapter 3 Decentralized Matching 12
- 3.1 Roommate Matching 12
- 3.2 Decentralized Marriage Models 14
- 3.3 Multi-agent Reinforcement Learning for Decentralized Stable Matching 25

Chapter 4 Cournot Games 39
- 4.1 Related work 42
- 4.2 Preliminaries 45
- 4.3 Methods 48
- 4.4 Experiments 51
- 4.5 Summary 66

Chapter 5 Conclusion 68

Bibliography 69

Vita 80
- Education 80
- Professional Positions 80
- Publications & Preprints 80

LIST OF FIGURES

3.1 Marriage Models 17

4.1 Examples of three different demand patterns. X-axis represents time steps in the simulation, and Y-axis represents varying values of parameter u (from the inverse demand function $u - vQ$, presented in eq. 4.1). Here, default value for u, $u_s = 40$. Figs. 4.1a, 4.1b, and 4.1c show demand patterns 1, 2, and 3, resp. 47

4.2 For small-scale simulations, comparisons of different measures obtained by ϵ-greedy, ϵ-greedy+HL, and ϵ-greedy+EL algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $v = 40$, $w = 1$, and $c = 4$. 54

4.3 For simulations with scaled actions, comparisons of different measures obtained by ϵ-greedy, ϵ-greedy+HL, and ϵ-greedy+EL algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms. The size of action space, S, varies from 50 to 500, with $u = S$ and $v = 1$, and $c = 4$. 54

4.4 For simulations with scaled number of agents, comparisons of different measures obtained by ϵ-greedy, ϵ-greedy+HL, and ϵ-greedy+EL algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $u = 1000$ and $v = 1$, and $c = 20$. 55

4.5 Comparison of average total time steps taken by agents to converge . . . 55

4.6 Comparison of variances in actions/production quantities chosen by different firms in an environment. 55

4.7 Comparison of average quantities obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. (a) Cournot model has $n = 3$, $K = 40$, $c = 4$, $u = 40$, and $v = 1$. (b) Cournot model has $n = 15$, $K = 50$, $c = 5$, $u = 300$, and $v = 1$. (c) Cournot model has $n = 30$, $K = 50$, $c = 5$, $u = 300$, and $v = 1$. 57

4.8 Comparison of average profits (corresponding to the quantities shown in the graphs above) obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. (a) Cournot model has $n = 3$, $K = 40$, $c = 4$, $u = 40$, and $v = 1$. (b) Cournot model has $n = 15$, $K = 50$, $c = 5$, $u = 300$, and $v = 1$. (c) Cournot model has $n = 30$, $K = 50$, $c = 5$, $u = 300$, and $v = 1$. 57

4.9 Comparison of average quantities and corresponding average profits obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. Cournot model has firms with $n = 10$, $K = 50$, $c = 5$, $u = 200$, and $v = 1$. 58

4.10 Comparison of average quantities and corresponding average profits obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. Cournot model has firms with $n = 10$, $K = 40$, $c = 5$, $u = 200$, and $v = 1$. 58
4.11 Comparison of average quantities and corresponding average profits obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. Cournot model has firms with $n = 10$, $K = 40$, $c = 5$, $u = 200$, and $v = 1$. 58
4.12 For Cournot duopoly model with 2 firms: comparison of joint quantity obtained by AWE ϵ-greedy and adaptive ϵ-greedy algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 2$, $K = 40$, $c = 4$, $u_s = 40$, and $v = 1$. Fig. 4.13a, 4.13b, and 4.13c show results for demand patterns 1, 2, and 3, resp. 61
4.13 For Cournot duopoly model with 2 firms: comparison of joint profit obtained by AWE ϵ-greedy and adaptive ϵ-greedy algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 2$, $K = 40$, $c = 4$, $u_s = 40$, and $v = 1$. Fig. 4.13a, 4.13b, and 4.13c show results for demand patterns 1, 2, and 3, resp. 62
4.14 For Cournot model with 10 firms: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 10$, $K = 50$, $c = 10$, $u_s = 500$, and $v = 1$. Fig. 4.14a, 4.14b, and 4.14c show results for demand patterns 1, 2, and 3, resp. 62
4.15 For Cournot model with 50 firms: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 50$, $K = 50$, $c = 20$, $u_s = 1000$, and $v = 1$. Fig. 4.15a, 4.15b, and 4.15c show results for demand patterns 1, 2, and 3, resp. 62
4.16 For Cournot model with 100 firms: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 100$, $K = 50$, $c = 20$, $u_s = 1000$, and $v = 1$. Fig. 4.16a, 4.16b, and 4.16c show results for demand patterns 1, 2, and 3, resp. 63
4.17 For Cournot model with scaled actions: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Corunot model has symmetric firms with $n = 2$, $K = 500$, $c = 4$, $u_s = 500$, and $v = 1$. Fig. 4.17a, 4.17b, and 4.17c show results for demand patterns 1, 2, and 3, resp. 63
4.18 For Cournot model with scaled actions: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 10$, $K = 500$, $c = 4$, $u_s = 500$, and $v = 1$. Fig. 4.18a, 4.18b, and 4.18c show results for demand patterns 1, 2, and 3, resp. 63

4.19 For Cournot duopoly model with 2 asymmetric firms: comparison of quantity obtained by an individual agent using AWE ϵ-greedy algorithm with respective Nash outputs. Corunot model has asymmetric firms with $n = 2$, $K = 40$, $c_0 = 1$, $c_1 = 3$, $u_s = 40$, and $v = 1$. Fig. 4.19a, 4.19b, and 4.19c show results for demand patterns 1, 2, and 3, resp. (AWE) in graphs refer to AWE ϵ-Greedy. 64
4.20 For Cournot duopoly model with 2 asymmetric firms: comparison of quantity obtained by an individual agent using AWE ϵ-greedy algorithm with respective Nash outputs. Corunot model has asymmetric firms with $n = 2$, $K = 40$, $c_0 = 1$, $c_1 = 5$, $u_s = 40$, and $v = 1$. Fig. 4.20a, 4.20b, and 4.20c show results for demand patterns 1, 2, and 3, resp. 64
4.21 For Cournot duopoly model with 2 asymmetric firms: comparison of quantity obtained by an individual agent using AWE ϵ-greedy algorithm with respective Nash outputs. Corunot model has asymmetric firms with $n = 2$, $K = 40$, $c_0 = 2$, $c_1 = 6$, $u_s = 40$, and $v = 1$. Fig. 4.21a, 4.21b, and 4.21c show results for demand patterns 1, 2, and 3, resp. 64

2.1 Comparison of Decentralized Approach with Random partitions for both Asymmetric and Symmetric cases 11
2.2 Comparison of decentralized approach with and without Budding heuristics for the 10-agent environment and the 20-agent environment. 11

3.1 Comparison of average total utility over 10 instances obtained by adaption of our approach for Roommate Matching with the utilitarian optimal matching 13
3.2 **Upper:** Results of asymmetric preference experiments in terms of average total utility averaged over 10 runs. We also list the percentage of the optimal utility obtained. The highest performers among the three models for each set of experiments are listed in bold. **Lower:** Results of symmetric preference experiments in terms of average total utility averaged over 10 runs. We also list the percentage of the optimal utility. The highest performers among the three models for each set of experiments are listed in bold. 22
3.3 For SM (asymmetric) case, MARL results on stability (%), instability measures $(Avg \pm Std)$ and median matches (%). 35
3.4 For SM (asymmetric) case, comparison of set-equality cost and regret cost in $(Avg \pm Std)$ format; results for 8 agents on 3×3 grid not included due to limited space. 36
3.5 For SM (asymmetric) case, comparison of egalitarian cost in $(Avg \pm Std)$ format. 36

4.1 Results of computer simulations for asymmetric Cournot duopoly model with firms having different marginal costs $C1$ and $C2$; $u = 1000$, $v = 1$. Q=Quantity; P=Profit. Results are in format $[q_1, q_2]$ for quantities and $[p_1, p_2]$ for profits; q_i and p_i represents quantity and profit for firm i, respectively. 65

Chapter 1 Introduction

Humans are social beings and naturally tend to interact and, in some situations, affect each other through their decisions, indirectly or otherwise. This interrelation can also be seen in terms of different organizations, communities, and even countries. Needless to say that this is also reflected in many scenarios involving robots and online bots. Thus, there has been much study on the interaction between entities, humans or otherwise. The interactions between two entities can fall anywhere in the collaborative-competitive spectrum. There could be full collaboration, for example, robots in rescue missions, or there could be full competition, for example, playing chess; however, many times, real-world situations require some collaboration along with some competition among participating entities. Usually, the participants do not have all the relevant information about the situation. Participants need to implement strategies to navigate through such interactions and gain something out of them. Figuring out such strategic approaches per the requirements of various real-world situations is crucial.

Social or economic interactions are modeled as games in game theory, giving those interactions some rules and structures that are supposed to provide insight into real-world situations. Following are the three main variants of games that we focus on:

1. Hedonic coalition formation games
2. Matching problems
3. Cournot games

A coalition formation game is a cooperative game in which agents must come together to form groups or coalitions. It can be defined as two or more parties who agree to cooperate (pool their resources) in order to obtain some mutually desired outcome. The parties involved may be individuals, groups, or collectivities of any size. Similarly, the outcomes may be anything humans desire (money, status, power, etc.), and the resources that are pooled may be whatever is needed to obtain the desired outcome (skills, abilities, money, etc.) Komorita and Kravitz [1983]. Hedonic games have applications in team formation and social group formation, where agents can have preferences over coalitions. Matching, which is a variant of hedonic games, finds application in problems like matching between workers-employers, students-colleges, residents-hospitals, etc. Cournot games mostly pertain to economic markets. In a standard Cournot game Cournot [1838], firms compete over the production of identical goods. Production or services in various real-world markets can be modeled as Cournot games; for example, energy systems Kirschen and Strbac [2018], transportation networks Bimpikis et al. [2019], and healthcare systems Chletsos and Saiti [2019]. While in the first two games, i.e., hedonic coalition formation games and matching problems, the agents can cooperate explicitly with each other either by forming coalitions or matches, in Cournot games, cooperation (also known as collusion) cannot happen explicitly; however, there is still a possibility of indirect cooperation. These games/problems can also be considered as multi-agent systems.

A multi-agent system (MAS) is a computer-based environment made of multiple interacting intelligent agents. There is no categorical definition of MAS. It can be seen as a loosely coupled network of problem-solving entities/agents that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity/agent Stone and Veloso [2000]. Multi-agent systems have applications in a wide variety of domains, including robotic teams, grid computing, electronic business, the semantic web, bioinformatics, computational biology, monitoring and control, resource management, education, space, military, and manufacturing applications Oprea [2004], Luck et al. [2003] and also collaborative decision support systems, computer networks, traffic and distributed control, in general Bu et al. [2008].

Most of the previous approaches for the types of multi-agent systems that we have considered have been centralized, assuming the presence of a central agency to control agents. However, this assumption is infeasible for many real-world scenarios. In the centralized approaches, typically, a centralized agency requires perfect knowledge about the system and every agent, and it determines the action course for every agent. In real-world situations, the applications of centralized techniques are very limited due to the fact that the agents operating in the environment are often concerned about their privacy and are reluctant to share any information about themselves. We consider decentralized multi-agent systems and train autonomous agents to learn behavior strategies using novel decentralized learning and multi-agent reinforcement learning approaches. No centralized agency is present in our models to direct the agents, and moreover, all the agents are autonomous, very much like most real-world scenarios.

As we model the problems as decentralized multi-agent systems with fully autonomous agents, challenges arise due to the dynamicity of the environment, defined by the uncertainty and non-stationary behavior of the agents. Because of these complex challenges, solving them with pre-programmed agents is hard. Instead, agents must look for solutions on their own, using some sort of learning approach. It is often important for agents to learn new behaviors online to gradually adapt to a system to improve their performance as well as that of the whole system Stone and Veloso [2000], Sen and Weiss [1999]. This is usually necessary for complex environments where making an apriori design of good agent behavior is difficult and sometimes impossible.

Additively separable hedonic games (ASHGs) are one of the simplest formulations of the coalition formation problem, and Aziz et al. Aziz et al. [2011a] show that finding optimal partitions for ASHGs is NP-hard in the strong sense. Therefore, we propose two approaches: decentralized learning and multi-agent reinforcement learning. In decentralized learning, autonomous agents explore the unknown environment and collect useful information about the environment, i.e., other agents present in the system, obtainable utility values, locations of other agents/coalitions, etc. Agents consistently update their information while making rule-based decisions based on current knowledge. Our motivation behind using decentralized learning is that we wanted to know its effectiveness before proceeding to use reinforcement learning, as we considered decentralized learning simpler and computationally easier than reinforcement learning.

A significant part of the research on multi-agent learning concerns reinforcement learning techniques. As using those techniques has been proven successful, we want to contribute by exploring new multi-agent reinforcement learning approaches for solving a subset of problems in multi-agent settings, as discussed above. Reinforcement learning (RL) is learning by interacting with an environment. An RL agent learns from the consequences of its actions rather than from being explicitly taught, and it selects its actions on the basis of its past experiences (exploitation) and also by new choices (exploration), which is essentially trial and error learning Sutton and Barto [2018]. The simplicity and generality of the setting make reinforcement learning attractive for multi-agent learning, but with added challenges of non-stationarity of environment, scalability in terms of the number of agents, and defining proper learning goals. In multi-agent systems, it is important to learn strategies/behaviors. Unlike single-agent environments, multi-agent systems are inherently non-stationary.

Multi-armed bandits (MAB) framework is a simpler version of reinforcement learning. It is also called a k-armed bandit problem where an agent repeatedly faces a choice among k different actions. After each choice, it receives a numerical reward depending on the selected action. It is named by analogy to a slot machine except that it has k levers instead of one Sutton and Barto [2018]. We model repeated Cournot games using a MAB framework, where firms learn independently and are autonomous. Each agent deals with its own multi-armed bandit problem separately. This setting provides a practical framework to model the Cournot game when there are assumptions of low cognitivity for the firms involved. Here, information on competitors is not available to the firms, and they cannot deduce the market demand function. Moreover, firms do not necessarily even need to know their own cost function. In any MAB problem, dealing with the exploration-exploitation trade-off is a crucial task. Hence, the MAB framework facilitates the use of exploration-exploitation approaches.

We simulated the above-mentioned problems in such a way that the simulations closely reflect real-world scenarios. We investigated the results according to the conventional game-theory metrics, such as the resulting equilibria (for Cournot games), optimality, or stability (for matching and coalition formation problems).

Chapter 2 Decentralized Hedonic Coalition Formation[1]

A coalition formation game is a cooperative game in which agents must come together to form groups, or *coalitions*. A *hedonic game* is a coalition formation game in which agents have preferences over the coalitions they can be a part of.

Many real-world problems can be modeled as hedonic games. Examples include team formation and social group formation. Typically these problems are solved using centralized approaches in which one controlling agent contains perfect knowledge about each agent's preferences and determines how coalitions should be formed. While these techniques have been shown to be effective, it is unlikely that all preferences will be known a priori in most real-world situations, limiting their effectiveness. In this work, we address these issues by introducing a decentralized algorithm for solving hedonic games.

For simplicity, we focus on solving *additively separable* hedonic games (ASHGs). In an additively separable hedonic game, an agent's utility for a coalition is equal to the sum of its utility for every other agent in the coalition. While this may be the simple formulation of the problem, Aziz et al. Aziz et al. [2011a] show that finding optimal partitions for ASHGs is NP-hard in the strong sense. Therefore, finding optimal coalition structures requires a heuristic approach. To this end, we model coalition formation as autonomous agents exploring a grid world and forming coalitions with other agents they meet. Unlike a centralized approach, in this formulation, each agent is responsible for finding its own coalition and does not initially know the preferences of other agents.

We also propose an extension to our decentralized technique inspired by Roth and Vate [1990] and Diamantoudi et al. [2004]. This extension, which we refer to as *budding*, is an additional search heuristic in which new coalitions are formed by breaking apart previously formed coalitions in a locally centralized way. We hypothesize that this will lead to higher utility coalitions for all agents involved. We introduce three budding heuristics in Section 2.3.

We explore how our decentralized approach performs on unconstrained coalition formation. In the unconstrained case, there are no restrictions on how coalitions can be formed. By exploring how our decentralized approach performs, we aim to show that our technique is applicable to a variety of potential problems. In experiments, we model all types of environments as grid worlds. The motivation behind using grid-world is that it can better represent real-world problems where agents must expend some effort to seek out information about other agents or form coalitions with others. For example, consider a situation in which students are asked to form teams for an assignment on the first day of class in school. Lacking any other information on their fellow students, they will physically explore around and inquire about forming teams with other students. This process is better modeled using a grid world so that the effort spent navigating is taken into account.

[1]The work in this chapter was published in EUMAS 2018 Taywade et al. [2018].

2.1 Related Work

Much of the past work on hedonic games has focused on centralized approaches where a single controller contains knowledge of agent preferences and determines how each agent should act in order to get a stable outcome Aziz et al. [2011b, 2013], Gairing and Savani [2010] or optimal coalition structure which improves the performance of overall system Rahwan et al. [2009], Rahwan and Jennings [2008], Michalak et al. [2010], Sandholm et al. [1998]. There have been extensions made to these techniques recently, however, in which each agent is responsible for forming its own coalitions Janovsky and DeLoach [2016a,b]. In each of these types of algorithms, a centralized is used to determine which solution is the best overall solution. This is possible because it knows the preferences of all agents. Our technique is truly decentralized since there is no controller that contains this additional information.

In this work, we develop a decentralized approach for solving hedonic coalition formation games. The idea of using decentralized approaches to solving this problem is not new. One common approach used to solve this problem in a decentralized way is to use reinforcement learning Abdallah and Lesser [2004], Jiang et al. [2008]. In these approaches, each agent explores its environment and learns how to form coalitions based on its own reward signal. Researchers have also explored the idea of augmenting reinforcement learning-based approaches by using additional heuristics Li and Soh [2004] or through the use of Bayesian reinforcement learning Chalkiadakis and Boutilier [2004]. The primary limitation of these approaches is that one must be able to model the hedonic game as a Markov decision process (MDP) in order to make use of reinforcement learning, which can be difficult to do depending on the nature of the hedonic game. In addition, reinforcement learning-based approaches also rely on reward functions that can be difficult for humans to define. Our decentralized algorithm does not rely on the environment having an MDP structure or on external reward functions and, thus, is more generally applicable than reinforcement learning-based approaches.

In this work, we also propose a novel heuristic for generating new candidate coalitions, which we call budding. This technique is inspired by the work done by Roth and Vate Roth and Vate [1990] and by the work of Diamantoudi, et al. Diamantoudi et al. [2004] which involve using sequences of blocking pairs as a means to find stable coalitions in matching problems.

2.2 Preliminaries

Definition 1. *Banerjee et al. [2001], Bogomolnaia and Jackson [2002] A* hedonic coalition formation game *(also just "hedonic game") G is defined by a set of agents, and the utilities each agent, a, holds for each coalition C containing a. A* coalition structure *π for G is a partition of the agents into coalitions. The goal of a hedonic game may be to find a utility-maximizing coalition structure or a stable one.*

There are many notions of *stability* for coalition structures for hedonic games.

In this work, we use a myopic notion of stability within a coalition, called *proximal stability* Schlueter and Goldsmith [2018]. This is related to "budding," introduced in Section 2.3, and is based on the standard hedonic games notion of core stability.

Definition 2. *Richardson [1956] Given a hedonic game G and a coalition structure π, a* blocking coalition *$C \notin \pi$ is a set of agents that have higher or equal utility for being in C as for being in their assigned coalition in π, and where at least one agent has strictly higher utility in C.*

We have decided to model problems as an additively separable hedonic game which is defined as follows:

Definition 3. *Banerjee et al. [2001], Bogomolnaia and Jackson [2002] In an* Additively Separable Hedonic Game (ASHG)*, each agent a_i has a utility, $p_{i,j}$, for each other agent a_j. The total utility $u_i(C)$ of agent a_i from the coalition C is $\Sigma_{j \neq i} p_{i,j}$.*

In hedonic games, preferences can be either *symmetric* or *asymmetric*. When preferences are symmetric, $p_{i,j} = p_{j,i}$ for all pairs i, j. In the asymmetric case, *i*'s utility for *j* might differ from *j*'s utility for *i* (so it's possible that $p_{i,j} \neq p_{j,i}$).

2.3 Method

The core element of our decentralized algorithm for hedonic games is that we model the problem as a multi-agent system where agents explore a grid-world environment. Each agent is initialized with no prior knowledge except for some basic information about the environment (such as its dimensions). As agents explore the environment, they will encounter other agents by moving into the same grid position as other agents and, in doing so, obtain preference information about them. Coalitions are formed when agents occupy the same grid cell and then choose to enter into a coalition. We go into more detail on this algorithm and the budding extension to this algorithm below.

Decentralized Algorithm for Hedonic Games

The core of our approach is that we model a hedonic game as a multi-agent system in which each agent navigates a grid-based environment and builds up knowledge of its own preferences. The agents themselves are autonomous, selfish, and myopic. They are selfish in that they are only concerned with their own utility, and they are myopic in that the only part of the environment they can see is their current grid cell location. In other words, each agent is unable to comprehend the whole environment. After each agent is initialized, it will explore the environment for a given number of time steps. This constitutes one *episode* of learning, and our algorithm will return the set of discovered coalitions after a set number of learning episodes. The number of time steps in an episode and the number of learning episodes can be customized for a specific hedonic game.

We will now discuss two key elements of our approach: what each agent can remember and recognize in the environment and how agents operate in the grid world environment.

What Agents Remember:

Agents store limited knowledge of the environment. Specifically, agents in our environment remember the following information:

1. After seeing another agent for the first time, an agent assigns a utility value to that agent and stores it for future reference. Agents only know their utility for other agents after they interact with them. Note that agents can only see other agents when they are in the same cell location on the grid.
2. Agents also remember the location where they last saw their *friends*. In this algorithm, we define an agent's friends to be other agents for whom it has a positive utility. This encourages agents to revisit these locations while exploring, as there is an increased likelihood of it being able to form a coalition with a friendly agent in that location.
3. Each agent also contains information about the highest utility values it has received from joining coalitions in the past. In our experiments, we store 10 values, but this parameter can be tuned to the problem being solved. This helps the agent evaluate the quality of its current coalition in terms of its own past experience. This further encourages the agent to seek out higher utility coalitions. One important thing to note is that this does not guarantee that an agent will always discover the coalition that results in its overall highest utility. This only encourages the agent to form coalitions that improve upon its past coalitions.

Operating in grid-world:

At the beginning of each episode, each agent is placed at a random position on the grid. At each time step, all agents simultaneously act in the grid world environment. The most common action for an agent is to *explore,* i.e., move to a different location in the grid world. The agent can also take the following actions depending on the specific situation it finds itself in. If an agent occupies the same grid position as another agent, these two can decide to form a coalition. The decision to form a coalition must be mutual, so it will never be the case that a coalition will be formed with only one willing party. The agents will enter into a coalition if the resultant coalition results in positive utility for both agents. If a coalition is formed, then that coalition will remain in that location on subsequent time steps. After entering into a coalition, agents will remain in that coalition for several time steps waiting for others to join. After a sufficient amount of time has passed, the agents in this coalition can consider leaving to form better ones or explore the environment.

If an agent moves into a square occupied by a pre-existing coalition, it can lobby to join the coalition. If the agent wishes to join this coalition, then each coalition member casts a vote to either let the new agent in or reject them. Agents will vote to accept the candidate if the resulting coalition has a higher utility than their current one, or they will vote to reject the candidate if the resulting coalition has a lower utility. If more than half of the agents in the coalition vote to let the new agent into the coalition, then the new agent becomes a member of the coalition.

If an agent is already in a coalition, it is possible that changes in coalition membership could cause the agent to want to leave. Since an agent's utility towards its coalition is affected by its preference over coalition members, new members being added or old members leaving will greatly affect how an individual agent views the coalition. If an agent's utility towards its own coalition becomes sufficiently small, then it may decide to leave the coalition to seek out better ones. This decision is influenced by utility values and the amount of time remaining in the episode. If there is still sufficient time remaining in the episode, it may choose to explore more to find higher-scoring coalitions. However, if a small amount of time is left, agents may choose to exploit the knowledge and remain in high-scoring coalitions. Note that just after joining the coalition, for a few steps, agents remain in the coalition (while still having positive utility) and wait to see if changes in the coalition caused by leaving or joining of other agents can improve its utility. Agents follow the following mechanism to maximize their utility and settle down in coalitions:

- For the first $(1/3)^{rd}$ of the total steps, an agent compares their current utility from the current coalition with the average of the top 10 highest utilities it has got so far. If its current utility is greater or equal to that average utility, then it stays in the coalition.
- For the second $(1/3)^{rd}$ steps, an agent compares their current utility with the average value of the top 10 utilities seen multiplied by 0.75. It stays in the coalition if its current utility is greater or equal to that value. This means an agent is more likely to stay in a coalition rather than further explore.
- For the next $(1/6)^{th}$ steps, an agent compares their current utility with the average value of the top 10 utilities seen multiplied by 0.5. It stays in the coalition if its current utility is greater or equal to that value.
- For the last $(1/6)^{th}$ steps, agents only leave a coalition if they have a negative utility for that coalition.

In this way, agents set high expectations and then gradually lower them as the clock runs down. The coalition structure at the end of the last episode is considered the solution coalition structure.

Budding

In the base form of our approach, coalitions can only be formed when multiple agents occupy the same location, and all agree to enter into a coalition. To further improve our approach, we have proposed an extension to our baseline approach, which introduces a new way to form coalitions. This technique, called budding, allows large coalitions to break apart into smaller ones. This helps avoid the situation in which many agents keep joining a pre-existing coalition when one or more sub-coalitions could be formed from this one, resulting in higher overall utility for the agents involved. Formally, we define budding as follows:

Definition 4. *Consider a set of agents $A = \{a_1, a_2, ..., a_n\}$, among which some agents form coalition C. Budding is when a sub-coalition, B, is formed from C. This happens when the number of agents in C reaches a certain threshold T, and forming a sub-coalition produces better total utility for agents in B than their total utility from the original coalition C.*

We have the following three budding heuristics:

1. Random: When a coalition, C, reaches a sufficient size, a random sub-coalition, B, is formed and the total utility of agents in B which they can get from being part of B is compared with their total utility from being part of C. This process is repeated until a B with higher total utility is found, or until the specified number of iterations of trying new sub-coalitions is reached.

2. Greedy: A random agent, a, is chosen from C to "seed" B. While it is possible to increase the total utility of the agents in B by adding an agent from C to B, we do so.

3. Clique Detection: We define a graph on C where vertices are agents in C, and edges are between friends. We set B as the first clique we find where the total utility of agents in B, which they are getting from being part of B, is higher than their total utility from being part of C.

We consider these budding strategies to be only locally centralized in that they only occur on a subset of agents that happen to be in a coalition. Since this requires only minimal information sharing between agents that are in a coalition, we still consider this approach to be decentralized in general.

2.4 Experiments

we initialize the grid world by placing agents in random positions on the grid. We then run the algorithm for 100 learning episodes of 4000 time steps each. Since our algorithm is non-deterministic, we run 10 instances with different preference profiles and report the average total utility of all the agents, averaging over those 10 instances.

For evaluating our approach for unconstrained hedonic games, we explored how it would perform in an environment with 10 agents exploring a 10×10 grid. One aspect of the environment that can have a large effect on the outcome is the range of expected utility values. In this test case, we experimented with two possible ranges of utility values: the range $(-5, 10)$ and the range $(-10, 10)$. When one agent encounters another agent for the first time, both agents will uniformly sample a preference from these ranges and assign it to the other agent. We evaluate both asymmetric and symmetric preferences. For this evaluation, we compare the results of our approach with the results of an algorithm that generates random partitions. We do not compare our results against the optimal solution as finding the optimal involves evaluating the utility of every possible partition, which is an NP-hard problem Aziz et al. [2011a].

We also evaluate the effect that our budding technique has on the quality of discovered coalitions. To do this, we performed experiments where we compared versions of our decentralized algorithm with budding implemented using each heuristic introduced earlier against a version of our algorithm that did not have budding implemented. We tested these methods in a 10×10 grid world with 10 agents, and also with 20 agents. For this study, we only consider a utility value range of $(-10, 10)$, and we assume that agents have asymmetric preferences. Since we are using budding, we must define a threshold value that determines when the budding process starts. We set this value to 9 when there are 20 agents on the grid (meaning that budding will begin when a coalition has 9 members), and we set this value to 6 when we examined 10 agents on the grid.

2.5 Results

The performance of our decentralized algorithm compared to random partitioning can be found in Table 2.1. As you can see in the table, our decentralized approach was able to outperform our random baseline significantly. Notably, this behavior is consistent across all utility ranges regardless of whether preferences were symmetric or asymmetric. While this baseline is not very sophisticated, this shows that our approach can drastically outperform this worst-case scenario.

The results of the comparison between versions of our decentralized algorithm with and without budding can be seen in Table 2.2. The results of this comparison show that the versions of our decentralized algorithm that had the ability to produce new coalitions through budding consistently outperformed the base version of our technique. This shows that our intuition about the potential benefits of budding for coalition formation was well founded. Of the three heuristics, it appears as though the clique-based approach performed better than both random and greedy heuristics. A more rigorous evaluation is required, however, before more definitive conclusions can be drawn.

Preference Type	Utility Range	Decentralized Approach	Random Partitions
Asymmetric	(-5,10)	188.3	57.8
	(-10,10)	67.3	-9.1
Symmetric	(-5,10)	224.2	37.2
	(-10,10)	134.7	19.6

Table 2.1: Comparison of Decentralized Approach with Random partitions for both Asymmetric and Symmetric cases

Number of Agents	Without Budding	Random	Greedy	Clique
10	69.55	72.44	74.55	75.88
20	229.6	235.8	235.2	241.1

Table 2.2: Comparison of decentralized approach with and without Budding heuristics for the 10-agent environment and the 20-agent environment.

2.6 Summary

We proposed a decentralized approach for solving hedonic games based on modeling the problem as a grid-world exploration problem. We feel that decentralized approaches better simulate how these problems work in a real-world environment and are, thus, more generally applicable than more common, centralized approaches. We also introduced a novel coalition discovery technique called budding, in which large coalitions spawn smaller sub-coalitions if they would increase the total utility of agents in the new sub-coalition. Our experiments showed promising results as our techniques both with and without budding performed well on a variety of hedonic games. We believe that our experiments provide strong evidence of the quality of our approach.

Chapter 3 Decentralized Matching

3.1 Roommate Matching [1]

In chapter 2, we have seen unconstrained coalition formation. In the unconstrained case, there are no restrictions on how coalitions can be formed, whereas in constrained coalition formation, explicit restrictions, such as limiting the total number of agents allowed in a coalition or limiting the number of total coalitions allowed, are present. Roommate matching is one such example of constrained coalition formation hedonic games. Roommate matching places constraints on the number of agents that can be in a coalition and the number of coalitions that can be formed. We formulated this problem on the grid. To implement our decentralized algorithm in this scenario, we must take these constraints into account when agents form coalitions. To do this, we make the following alteration to our algorithms proposed in chapter 2:

- Because the number of roommates per room is limited to C, if more agents converge on a cell, we choose the set of C that has the highest utility in a centralized manner. All other agents must leave the cell and explore further.
- Since the number of coalitions must not exceed R, our approach has certain pre-designated time steps on which the total number of coalitions on the grid is checked. If there are more than R coalitions, then the coalition with the lowest total utility across all its agents will be forced to dissolve and agents in that coalition will search for other coalitions to join.

Here, we make the assumption that agents do not have preferences over specific rooms, only over their roommates.

Related Work

Irving proposed an algorithm to find the complete stable roommate matching or report if it does not exist Irving [1985]. Tan proposed the modified version of the algorithm proposed by Irving to find a maximum stable matching, i.e., a maximum number of disjoint pairs that are stable among themselves Tan [1990]. Irving and Manlove further proposed an algorithm to find stable roommate matching with ties in the preferences or the incomplete list of preferences Irving and Manlove [2002]. Fleiner et al. also proposed methods for handling generalized stable roommate matching in which preference lists may be partially ordered and forbidden pairs may be present Fleiner et al. [2007]. However, all of these works assume the centralized framework and are limited to finding the pairs of roommates, i.e., matching only up to two roommates.

[1] The work in this section appeared in our EUMAS 2018 paper Taywade et al. [2018].

Preliminaries

Definition 5. *An instance of* Roommate Matching *consists of a set of agents* $A = a_1, a_2, .., a_n$ *and R, the available number of rooms, and a uniform maximum capacity per room of c. The goal is to form a partition P of agents where each coalition in P has at most C agents, and there are at most R coalitions.*

Experiments

For roommate matching, we consider a 10x10 grid with 9 agents. In this environment, a "room" can contain 3 agents, and there are 3 rooms available. In other words, it is possible for each agent to be a member of a coalition as long as each coalition has 3 members. In these experiments, we compare the results of our decentralized algorithm against the utility associated with optimal matching. We consider 3 different utility ranges in these experiments: a range of $(1, 10)$, a range of $(-5, 10)$, and a range of $(-10, 10)$. In this problem, we assume that agents have asymmetric preferences.

Results

The results of the experiments we conducted on the roommate matching problem can be seen in Table 3.1. In these experiments, we were able to compare against the optimal matching. The first thing to note is that our technique is not able to reproduce the optimal matchings. That being said, we feel that our performance was comparable to this upper bound. We can see from Table 3.1 that our approach gives results close to the optimal matching for all three utility ranges.

Utility Range	Decentralized Approach	Optimal Matching
(1,10)	102.3	129.6
(-5,10)	65.8	91.3
(-10,10)	42.9	64.6

Table 3.1: Comparison of average total utility over 10 instances obtained by adaption of our approach for Roommate Matching with the utilitarian optimal matching

Summary

We showed that our decentralized learning approach could be adapted for roommate matching, and we can get near-optimal results. Our proposed algorithm can work for an arbitrary number of rooms (coalitions to form) and capacity of rooms (limit on the number of agents per coalition).

3.2 Decentralized Marriage Models [2]

Finding a good partner has always been one of the important things in people's lives. The problem of determining who should marry whom is referred to as the *marriage problem*, and it is frequently modeled as a *two-sided matching* problem. A two-sided matching problem consists of two disjoint sets of agents where agents are matched to members of the other set. In this problem, we assume that agents have preferences over the types of matches that can be made.

There are several personal, social, and cultural factors that influence how people find potential mates and, thus, which people will end up together. There are many social structures that control what potential matches can be made.

We explore the problem of finding optimal partner matchings under different assumptions about how people are introduced. Specifically, we model the marriage problem using three different environments: a grid-world environment, a small-world network graph, and an affiliation network graph. Each of these environments represents a practically grounded way that people meet each other in the real world. In the grid-world models, the agents that actively seek out potential partners on their own have little to no prior information about the other agents they will meet. In the small-world network environment, agents are presented with potential matches based on their degrees of separation. In other words, agents that are closer to each other in terms of their social network are more likely to be considered as candidates for a match. Affiliation networks model the situation where agents are registered to matrimonial agencies and get potential matches suggested by those agencies. In many real-world scenarios, it is not feasible for a centralized agent or agency to optimally select partners for agents. Typically, agents — people — act to find partners autonomously without the need for a centralized agency. Thus, to better simulate real-world scenarios, we introduce a heuristic-based, decentralized approach for solving the marriage problem and show how it can be used, with slight modifications, to help agents find matches in each of the three environments described above. We show the effectiveness of our approach by demonstrating its ability to find good matchings in each of our three test models, where good matchings are defined as those that maximize the utility of all agents. We then compare the matches produced by our decentralized algorithms with several baselines. As an upper bound, we compare our matches against the matches found using the (centralized) Hungarian algorithm Kuhn [1955], which produces optimal matches. We also compare against matches produced from the Gale-Shapley algorithm Gale and Shapley [1962] and a bidirectional local search algorithm Viet et al. [2016] as a more informed set of baselines. Note that all of these comparisons are with centralized algorithms. We include them to get some sense of the range of possible total utilities. We also compare with Hoepman distributed matching algorithm Hoepman [2004]. We also show that our methods scale for large instances of up to 500 agents. As our method produces outcomes in polynomial time, it is fitting for environments/models where large numbers of agents need to

[2] The work in this section was published in the proceedings of FLAIRS 2020 Taywade et al. [2020].

make quick decisions.

Preliminaries

Definition 6. *An instance of* Bipartite Matching *is a bipartite graph* $G = (S_1 \cup S_2, E)$ *with* $E \subseteq U \times V$*. A solution is maximal matching. We consider the problem of* Weighted Bipartite Matching, *where the edge weights correspond to preferences.*

Definition 7. *An instance of the* optimal marriage problem *is an instance of the weighted bipartite matching problem, where U and V correspond to the categories of agents,* $p_{i,j}$ *is the preference of i for j, i.e., the weight of edge* (i, j)*, and the goal is to find a maximum weight matching. When preferences are asymmetric, the weight of a single pair is the sum of the weights of the two directed edges between those nodes.*

Definition 8. *A social network is a graph, where nodes are individuals and edges are relationships. The edges can be annotated with the type of relationship, and nodes may be annotated, for instance, to indicate membership in a node set in a bipartite graph for matching purposes.*

One measure of interest in social network graphs is the clustering coefficient, which is the average "cliquishness' of nodes in the graph. Note that, for a node with k neighbors, those neighbors may or may not connect with each other, for a maximum of $\binom{k}{2} = k(k-1)/2$ edges.

Definition 9. *For a given graph G and node n with k neighbors, we define the* cliquishness *of n to be the ratio of the number of edges connecting n's neighbors to* $\binom{k}{2}$*. The* clustering coefficient *of G is the average of* C_n *over all nodes n Watts and Strogatz [1998].*

We now define the network of interest, inspired by the folkloric "six degrees of separation."

Definition 10. *A small-world network is a network where the expected distance D between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes N in the network (*$D \propto \log N$*), while the clustering coefficient C is not small Watts and Strogatz [1998].*

Definition 11. *The Erdös-Renyi graph generation model is specified by two parameters: the number of vertices in the graph n, and the probability of an edge p. Given n and p, we choose a graph on n vertices by including an edge between each pair of vertices with probability p, independently for each pair. Mathematically,* $G \Leftarrow G'(n, p)$*, which indicates that G is a random graph chosen from this distribution* G'*.*

Our affiliation/bipartite graph is a bipartite version of the Erdös-Renyi graph. The bipartite random graph generator chooses each of the $n \times m$ *(undirected) or* $2 \times n \times m$ *(directed) possible edges with probability p and each node has an attribute* BIPARTITE *with value 0 or 1 to indicate which bipartite set the node belongs to.*

Related Work

One popular approach to solving marriage market problems and job matching problems is to use decentralized approaches. These methods explore how random meetings can result in stable matches since matches are generated by the agents themselves rather than a centralized controller agent. Most decentralized approaches for solving marriage problems are focused on finding stable matches Lauermann and Nöldeke [2014], Wu [2015]. Many of these techniques have focused on using preferences or interests as the basis for their matchings Vanzin and Barber [2006], Roth and Xing [1997], Foner [1997] and we also do the same in our work. Since there may be many potential stable matchings in marriage problems, there has also been work that seeks to identify the stable matchings that are most likely to prevail Boudreau [2011]. Contrary to much of this work, our decentralized approach focuses on generating optimal matchings instead of stable matches. Some earlier work has looked at optimal matchings in the centralized setting Irving et al. [1987], Nemhauser and Weber [1979]. While these algorithms are elegant solutions to the problem of assigning individuals to pairs, they presuppose the central authority with access to all individuals' preferences. Our work avoids this assumption.

Distributed algorithms for weighted matching mainly include algorithms that are distributed in terms of agents acting on their own either synchronously or asynchronously Hoepman [2004], Wattenhofer and Wattenhofer [2004], Khan et al. [2016], and algorithms that are distributed using parallel programming to find approximate maximum matchings (references omitted due to space constraints). In our work, we explore how social networks and pre-existing preferences affect the pairings produced by a decentralized matching algorithm, which has not, to our knowledge, been examined in previous research.

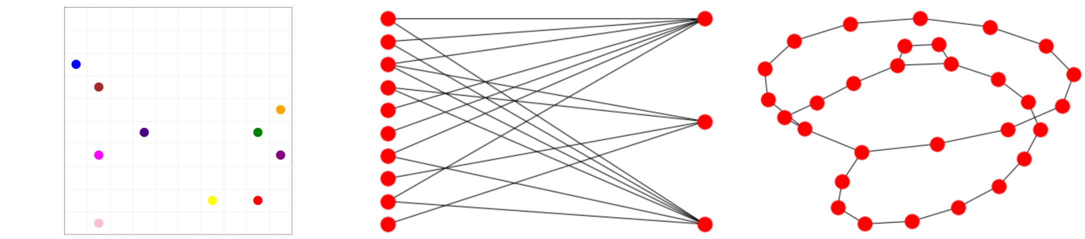
Methods

As mentioned before, we model the marriage problem in three types of models: grid-world, affiliation network, and small-world network. In this section, we introduce our three models thoroughly. We assume that agents start off agnostic about the world and about their potential mates in every model. We propose a decentralized approach to help autonomous agents find their partners in all three models. The models differ in how, and how often, agents meet.

Grid-world Environment

Model The grid-world model is a spatial environment where agents can freely navigate the grid and encounter each other. We deploy agents at random cell locations on the grid. Agents can only see the contents of their current grid location.

Algorithm The grid-world environment contains S_1 and S_2 agents in equal numbers. At the beginning of each episode, agents are initialized at random starting locations. From here they start randomly exploring the grid environment. When an agent



(a) Grid-world with 10 agents on 10×10 grid

(b) Affiliation Network with 10 agents and 3 agencies

(c) Small-world graph with 30 nodes, average degree of separation ≈ 6

Figure 3.1: Marriage Models

encounters another agent of the opposite category, it discovers how much utility it can get if it is matched with that agent and it stores that utility value if it is positive (which indicates that it can be a potential match). It does not store the identity of the other agent. If two agents from the same category encounter each other, they ignore each other. Each agent also has basic information about the total number of agents of the opposite category present in the environment. Using this, they can calculate how many potential matches could be formed and can, from that, reason about how many candidate agents they have not encountered yet. As explained below, this helps determine if an agent forms a match and stays there, or if the agent chooses to explore to form a potentially better match.

We define the exploration rate r to be the ratio of steps so far to the total number of steps. For each individual, we define h to be the average of all positive utilities an agent has seen so far, c is the highest possible utility for an agent in the environment and u is the utility agent is getting from a match if it is part of the one. An agent decides to form a match with another agent when they encounter each other in the same cell location if they each get positive utility from the match. If an agent encounters more than one agent of the opposite category at the same cell location then it chooses the best among them. If two opposite-category agents are willing, then the match is formed (they begin dating). Agents only stay in the match when they find themselves in one of the following situations:

- If $u \geq 0.75 \times c$;
- If $r \leq 0.6, u \geq h$;
- If $0.6 \leq r \leq 0.8, u \geq 0.5 \times h$;
- If $r \geq 0.8, u \geq 0$.

In this way, agents first set high expectations and then gradually lower their expectations as time passes. Note that, if an agent encounters a better agent at the same cell location, while in a match, then it may leave the current match and form a new one with a better agent if that new agent is also interested. The matching at the end of the last episode is the resulting matching (the marriage). The values used for each of these situations can be tuned to specific tasks or environments. We use these values because they worked well for us in practice.

Affiliation Network/Bipartite Network

Model An affiliation network is usually used to represent common membership of groups, and in our case refers to membership in matrimonial agencies or dating websites. In this affiliation network, actors are connected by common membership. We represent such a network as a bipartite graph, with nodes being either individuals or matrimonial agencies. Edges represent individuals' memberships in the agency. In our affiliation network model, actors are people who are looking for partners and collectives are agencies to which actors are registered. These matrimonial agencies suggest matches to their members. Note that an actor can be a member of more than one agency.

Algorithm As mentioned before, we create two types of agents, matrimonial agencies and people, as nodes in the affiliation graph; people are randomly assigned to agencies. At each step, agencies randomly suggest a match to everyone who is registered with them. Note that these suggestions are not necessarily symmetric: an agent may be suggested to another, without receiving the other as a suggestion. The one receiving a suggestion might express interest in the other. When individuals accept each other as a match, they are removed from the agencies' registered candidate lists.

Each individual has a list of possible matches, including others they are interested in, and those who are interested in them. They may select one match to propose to, and be accepted or not. However, if they receive a better proposal in that time step, they retract their own proposal. If a proposal is offered and accepted and not retracted in that time step, then the marriage happens. If no marriage happens for an individual in a particular time step, then the individual's list is updated at the next step when they get new suggestions from agencies, until they get married. People are removed from others' lists when they get married. To recap: when an agent expresses interest in another agent, the couple is married unless the agent receives interest from a better candidate at the same time step, or the recipient refuses them. The factors that affect an agent's decisions on whether to propose, or to accept a proposal, are the utilities the agent has seen so far, and the time left.

We define the exploration rate r to be the ratio of steps so far to the total number of steps. For each individual, we define h to be the average of the five highest utilities an agent has seen so far (Note that h is particular to an agent and their history so far). Finally, c is the utility of the candidate on that individual's current list. We denote by max the maximum possible utility. There are four possibilities listed below when agents decide to take action (proposing to someone or accepting a proposal).
• If $r < 0.4$ and $c \geq 0.75 \times max$;
• If $0.4 \leq r \leq 0.6$ and $c \geq h$;
• If $0.6 \leq r \leq 0.8$ and $c \geq 0.75 \times h$;
• If $0.8 \leq r$ and $c \geq 0.5 \times h$.

If any of these conditions hold for an individual with respect to someone who was suggested, then they propose. If a proposal has been received, and the recipient finds themselves in one of those conditions, then they will accept the proposal. Note that, in the fourth step, expectations have been lowered, similar to what happens as time runs out in the grid-world environment. As before, each of these conditions can be tuned for specific environments. We selected these because they performed well for us in practice.

Small-World Network

Model The final model we consider is a social network graph. This model is meant to formalize the process of finding matches through one's social network. In practice, people are often introduced by friends, or through chains of acquaintances. We randomly generate suggested matches, with probability inversely proportional to their distance in the graph, as a proxy for such introductions. Every node represents someone in search of a mate, and also a conduit for introductions. We use a small-world network, where the average distance between nodes is somewhere between 6 and 7, based on the folkloric "six degrees of separation" introduced by Travers et al. Travers and Milgram [1967, 1977] and further confirmed by Dodds et al. Dodds et al. [2003]. We use the Watts-Strogatz model Watts and Strogatz [1998] of small-world graph construction to build our model.

Algorithm This is a variant of the Affiliation Network algorithm, where suggestions come not from agencies, but from the individuals' social network (small-world network). The likelihood of two individuals i and j meeting is a function of their distance $d(i, j)$ in the network. At each stage, individuals receive at most 3 suggestions/introductions. To generate the list of suggestions for i, the algorithm first samples (uniformly at random) one from each distance from i. The sampled individuals are added to the list with probability $\frac{1}{2d(i,j)}$. (Note that closer individuals are more likely to be added.) Finally, if the list has more than 3 suggestions, individuals on the list are chosen uniformly at random for culling from the list for this stage. We have chosen these parameters as they keep the number of introductions per agent balanced (not too high and not too low) and realistic while comparing with that of other models.

Note that, unlike the grid world, in the affiliation network and social network algorithms, once agents become paired with someone they are considered married forever.

Experiments

In experiments, we evaluate our algorithms on three criteria: 1) The quality, measured as total utility; (2) the scalability with respect to the number of agents, and (3) robustness with respect to changes in the range of utilities. To test quality, we calculate the total utility of all matchings found by our algorithm in each of the three environments. Note that in our decentralized models, agents selfishly try to increase their own utility; their aim does not include optimizing the total utility of the whole system. We also examine the total number of matches found by each algorithm. We run each algorithm on each environment 10 times using different randomly assigned preferences between agents each time.

The performance of our method is the average cumulative utility over these trials. To evaluate scalability, we perform these trials with varying numbers of agents. Specifically, we consider trials with 100 agents in each environment, and trials with 500 agents. To evaluate the robustness of each algorithm with respect to observed utility values, we also perform trials with different utility ranges. We consider the following two ranges: $(1, 10)$ and $(-10, 10)$. Thus, for each environment model, we perform 4 sets of experiments with different parameters: 100 agents, $(1, 10)$ utility range and $(-10, 10)$ utility range; 500 agents, $(1, 10)$ utility range and $(-10, 10)$ utility range. Recall that preferences can be symmetric or asymmetric. For completeness, our experiments consider both symmetric preferences and asymmetric preferences for each set of trials run. To contextualize these results, we compare them against five baseline algorithms discussed in the next section.

Baselines

For this evaluation, we compare our decentralized algorithms against four baselines: Gale-Shapley algorithm Gale and Shapley [1962], bidirectional local search Viet et al. [2016], Hoepman algorithm (decentralized) Hoepman [2004] and optimal matchings found using the Hungarian algorithm Kuhn [1955]. The random matching algorithm provides a lower bound on the expected quality of matches found, whereas the matches found by the Hungarian algorithm are optimal. The Gale-Shapley and bidirectional local search algorithms serve as a more informed baseline to compare against. Note that the goal of each of these algorithms is to find stable matches, which is different than our goal.

The Hungarian algorithm is optimal in a utilitarian sense, meaning that individual agents may get low-utility matches in order to maximize the total utility. This happens particularly in the asymmetric-preference case. Therefore, comparison with distributed Hoepman algorithm is important because there, agents also act selfishly, as in our approaches.

We discuss each of these baselines in greater detail below.

Gale-Shapley: We assume familiarity with this algorithm.

Bidirectional Local Search: This algorithm also attempts to find optimal stable matchings but via bidirectional local search. It simultaneously searches forward from the man-optimal stable matching and backward from the woman-optimal stable matching until the search frontiers meet.

Hoepman Algorithm: This is a decentralized variant of the sequential greedy algorithm Preis [1999] which computes a weighted matching that has utility at least half that of the maximum-utility matching. Agents act independently and communicate via asynchronous messages. Unlike our setting, agents all know each other a priori.

Hungarian Algorithm: The Hungarian algorithm finds optimal weighted bipartite matchings.

Decentralized Algorithm Parameters

In the following sections, we discuss the parameters that we used for designing our decentralized models and algorithms.

It is difficult to get impartial real-world data for these models, so we have chosen parameters according to our intuition. We will discuss the parameters we have chosen in greater detail below.

Grid-world Model: In the grid-world environment, learning occurs in episodes in which agents are allowed to take a fixed number of actions before the learning for that episode concludes. For our experiments, we only use two episodes to train each agent. Between episodes, agents remember the positive utilities they have received so far by being matched with other agents. The number of training steps in each episode changes based on whether there are 100 or 500 agents in the environment as well as the size of the grid: • 100 agents: (20×20) grid, 1000 steps • 500 agents: (45×45) grid, 30,000 steps. After exhausting the specified number of time steps, agents are reset to random starting locations and learning resumes.

Affiliation Network Model: Our affiliation network model is a bipartite version of the binomial (Erdös-Renyi) graph model Erdos and Renyi [1959], Bollobás [2001], Gilbert [1959]. For the experiments, we use the following parameter combinations: • n=100, m=5, p=0.5 • n=500, m=10, p=0.5.

Thus, for experiments with 100 agents, we have 5 agencies that agents can be members of and there is a 50% chance that an agent belongs to a given agency. For experiments with 500 agents, we increase the number of agencies to 10. Using these parameters, we create affiliation networks in which each agent is affiliated with 4–5 agencies on average. We chose these parameters by conducting an informal social survey among people that are registered with matrimonial agencies. We found that people that are registered to these agencies are typically registered to 4–5 on average. As in the grid world, we use 2 learning episodes for each set of experiments. For the experiments involving 100 agents, an episode lasts for 1000 time steps. For experiments involving 500 agents, an episode lasts for $30,000$ time steps.

Small-world Network Model: For constructing a small-world graph, the following parameters are required: the number of nodes, n; the number of nearest neighbors that each node is joined with, k; and the probability of rewiring each edge, p. Depending on how these values are assigned, the average shortest path between any two nodes will change. We have chosen parameters such that the average shortest path between any two nodes is about 6. This is inspired by the *six degrees of separation* phenomenon Travers and Milgram [1967], Dodds et al. [2003]. To achieve this average distance between nodes, we set the parameters for a network with 100 agents as follows: n=100, k=5, and p=0.05. This results in a network where the average shortest path between two nodes is approximately 6.2. For a network with 500 agents, we assign the following values: n=500, k=4, and p=0.15. This results in a network where the average shortest path between two nodes is approximately 6.6. For the social network graph model, we also use learning episodes. The number of

Agents; Utility Range	Gale-Shapley	Bidirectional Local Search	Hoepman Algorithm	Grid-world	Affiliation Network	Small-world Network	Optimal (Hungarian)
100; (1,10)	608.5 (65.1%)	915.2 (98.05%)	742.2 (79.51%)	**790.5 (84.69%)**	776.2 (83.14%)	753.3 (80.7%)	933.4
100; (-10,10)	134.6 (16.4%)	779 (95.10%)	676.9 (82.63%)	**678.1 (82.78%)**	592.5 (72.34%)	606.7 (74.06%)	819.1
500; (1,10)	2841.5 (57.30 %)	4881.7 (98.45 %)	3740.7 (75.44 %)	4018 (81.03 %)	**4368.1 (88.09 %)**	3924.2 (79.14 %)	4958.4
500; (-10,10)	461.5 (9.8%)	4577.3 (97.41 %)	3667 (78.04 %)	3821.9 (81.34%)	**3935.4 (83.76%)**	3626.8 (77.18%)	4698.6
100; (1,10)	943.4 (94.4%)	988 (98.89%)	738.8 (73.95%)	852.9 (85.37%)	843.2 (84.40%)	**875 (87.58%)**	999
100; (-10,10)	856.6 (87.8%)	937.4 (96.08 %)	729.6 (74.78 %)	799.3 (81.92%)	**817 (83.74%)**	814.2 (83.45%)	975.6
500; (1,10)	4943.6 (98.9%)	4995 (99.9 %)	3739 (74.78 %)	4332.1 (86.64%)	**4666.8 (93.33%)**	4553.6 (91.07%)	5000
500; (-10,10)	4827.2 (96.5%)	4974.6 (99.49 %)	3687.6 (73.75 %)	4237.6 (84.75%)	**4582.8 (91.65%)**	4486 (89.72%)	5000

Table 3.2: **Upper:** Results of asymmetric preference experiments in terms of average total utility averaged over 10 runs. We also list the percentage of the optimal utility obtained. The highest performers among the three models for each set of experiments are listed in bold. **Lower:** Results of symmetric preference experiments in terms of average total utility averaged over 10 runs. We also list the percentage of the optimal utility. The highest performers among the three models for each set of experiments are listed in bold.

episodes and number of time steps per episode are defined the same as they were for the grid-world environment model and the affiliation model network.

Results

We have compared our methods for the three marriage models to several centralized algorithms and to the decentralized Hoepman algorithm. These results are summarized in Table 3.2 for both asymmetric and symmetric preferences. Note that our approaches significantly outperform the Hoepman algorithm in most cases.

For asymmetric preferences, all three of our decentralized algorithms outperform the Gale-Shapley algorithm in average total utility but bidirectional local search algorithm performs better than our decentralized algorithms. For 100 agents with a utility range of (1,10), all three of our decentralized algorithms achieved roughly 80% of the performance of the Hungarian algorithm. When utility ranges were expanded to include values from (-10,10), the performance of our algorithms with respect to the optimal matchings decreased to about 70% for affiliation networks and small-world networks. The performance of our algorithm in the grid-world environment remained consistent by achieving around 83% of the performance of the Hungarian algorithm. When the number of agents increased to 500 with asymmetric preferences, the performance of our algorithms with respect to the Gale-Shapley algorithm and bidirectional local search algorithm remains consistent. In both utility ranges (1,10) and (-10,10), our algorithms' performance relative to the optimal matching improves compared to the experiments with fewer agents, and also, the difference between results for two different utility ranges goes down. Each of our algorithms is able to achieve roughly 80% of the performance of the optimal matchings in this case. Interestingly, the affiliation network setting performs better than the other two settings in both types of utility ranges.

In the set of experiments with symmetric preferences, the matchings produced by the Gale-Shapley and bidirectional local search algorithm outperformed the matchings produced by our decentralized algorithms in all environment models in terms of average total utility. For experiments using 100 agents, our decentralized algorithms achieved about 80–85% of the performance of the Hungarian algorithm across both utility ranges. When the number of agents was increased, the performance of our decentralized approaches increased to 85–90% of the performance of the utility of the optimal matchings. The affiliation network setting performs better than the other two settings except for the case of 100 agents with a $(1, 10)$ utility range. And the overall performance of small-world and affiliation network settings improves over the grid-world setting in the symmetric case.

The performance of our methods compared to the Hoepman algorithm highlights the power of our decentralized approach for solving matching problems when using a variety of representations. In addition, the fact that our method produced matchings with higher total utility despite each agent being selfish further shows the power of our algorithm.

Another thing to note is that our decentralized algorithms drastically outperformed the Gale-Shapley algorithm and gave quite close results to the bidirectional local search algorithm in the set of experiments with asymmetric preferences. This shows that our decentralized approaches are very effective in these situations. This is especially notable because our decentralized approaches are more generally applicable since they do not rely on a centralized agent that contains perfect knowledge of agent preferences. Despite their performance in asymmetric environments, our decentralized approaches were not able to outperform the Gale-Shapley algorithm in environments with symmetric preferences. This is likely due to how the Gale-Shapley algorithm finds matches. When preferences are symmetric, it is likely that the matchings produced by the algorithm are not so bad for the second group. The explanation for the bidirectional local search algorithm performing better than our approach is that, even if it attempts to optimize a solution like our approach, it does it in a centralized manner which is a big advantage that our approach does not have.

Summary

When we compare our decentralized methods for the three different marriage models against each other, we find that they give somewhat similar results. This roughly indicates that different environment settings do not significantly change the quality or the quantity of matches found. When we compare results for affiliation network and small-world network settings as both of them are social networks, we noticed that in 8 out of 10 different experiment settings, the affiliation network performs better than the small-world network setting. This may suggest that when agents are affiliated with agencies, there is a slightly better chance of finding a good partner than just depending on interpersonal networks, but it certainly needs more investigation as some of our parameters were based on our discretion and not on real-world data.

Recall that one of the motivations for using a decentralized approach is that it better simulates how people find matches in the real world. The fact that each of our decentralized algorithms performed well across all environments and test cases indicates that they can be useful across different strategies for finding partners.

We proposed the use of three decentralized models for matching in marriage problems and proposed algorithms for those models. In each case, the overall utility of matches found was, on average, 84% of the utility achieved by the optimal (Hungarian) centralized algorithm. Since we argue that the social problem is inherently decentralized, agents have limited information, and preferences are typically asymmetric, so optimal sets of matchings are quite rare; our results are quite strong. Our approaches outperform the distributed Hoepman algorithm, which supports our claim of good results. We also compared our results to those of the Gale-Shapley algorithm for a stable marriage and the bidirectional local search algorithm. Gale-Shapley is optimal for one side and not the other; as expected, when preferences were asymmetric, our algorithms performed significantly better than Gale-Shapley. However, when preferences are symmetric, the Gale-Shapley algorithm produced nearly optimal utility matchings better than the decentralized algorithms. Millennia of literature assures us that human romantic preferences, however, are not symmetric. As expected, the bidirectional local search algorithm performed better than our approach.

One can view the models in this work as reflecting three different approaches to find a partner. Our work, as presented here, offers no insight into which of these approaches is better. Nonetheless, we successfully provided simple, scalable, and easily adaptable methods for utilizing individually acting agents in three different decentralized environments with a large number of agents.

3.3 Multi-agent Reinforcement Learning for Decentralized Stable Matching[3]

Matching markets are prevalent in the real world, for example, matching of students to colleges, doctors to hospitals, employees to employers, men and women, etc. A two-sided market consists of two disjoint sets of agents. In a two-sided stable matching problem, each participant has preferences over the participants on the other side. A matching is stable if it does not contain a blocking pair. A blocking pair is formed if two agents from disjoint sets prefer each other rather than their current partner. Although, most of the prior literature focuses on centralized algorithms where the entire set of preferences is known to some central agency, having such a central clearinghouse is not always feasible. Therefore, we consider a decentralized matching market with independent and autonomous agents.

There have been several decentralized matching methods proposed in recent years. However, many of them assume that the agents have knowledge of one another's preferences and can easily approach/contact each other, i.e., negligible search friction. In reality, it takes time to meet a partner and to learn the value of said partnership. Furthermore, there is seldom a scope for knowing the preferences of other agents. Also, it is a crucial task to locate and approach a potential match, either by navigating physically or virtually. Several decentralized matching markets, such as worker-employer markets and buyer-seller trading markets, consist of locations at which matching agents may meet, be it physically or online. The level of information, search cost, medium of interaction, and commitment laws can vary across markets. Nonetheless, these are the important features of decentralized markets. Some research works study the impact of these features on the final outcomes for certain types of markets Echenique and Yariv [2012], Pais et al. [2012, 2017]. To better represent these features, we propose a generalized matching problem in which agents are placed in a grid world environment and must learn to navigate it in order to form matches. We see this as a generalized case for matching problems. While it contains the features described above, it does not conform to the standards set by any individual market type.

There are multiple factors involved in deciding a preferred match in real-world situations, and having a score for each match is more expressive. Thus, we consider weighted preferences for a stable matching problem, which is discussed in Gusfield [1987], Irving et al. [1987], Pini et al. [2013]. The weighted preference is used as the utility value (or *reward*) for being in the match. These scores reflect the underlying preference order. Agents are initially unaware of others' preferences as well as of their own. In many matching markets, knowledge acquisition is important: in labor markets, employers interview workers; in matching markets, men and women date; and in real estate markets, buyers attend open houses. We have taken this into account, so an agent gets to know a noisy version of its utility for a match only after being part of it. Noise represents uncertainty in the value of a partnership, e.g., the uncertain nature of human behavior in relationships.

Finding a long-term match in this scenario is quite a complex task. Therefore,

[3] The work in this section was published in the proceedings of ADT 2021 Taywade et al. [2021].

we propose multi-agent reinforcement learning (MARL) as an alternative paradigm where agents must learn how to find a match based on their experiences interacting with others. We equip each agent with its own reinforcement learning (RL) module. We use *SARSA*, a model-free, online RL algorithm. Instead of a common reward signal, each agent has a separate intrinsic reward signal. Therefore, we model this problem as a stochastic/Markov game Littman [1994], which is useful in modeling multi-agent decentralized control where the reward function is separate for each agent. Agents learn to operate in the environment with the goal of increasing the expected total reward by getting into a long-term, stable, or close-to-stable and fair match. We impose search cost as a small negative reward (-1) for each step whenever an agent is not in a match.

We investigate the applicability of the MARL approach to the conventional *stable matching (SM)* problem, as well as its extensions, such as *stable matchings with incomplete lists (SMI),* where agents are allowed to declare one or more partners unacceptable Gusfield and Irving [1989], and *stable matching with ties (SMT),* where agents have the same preference for more than one agent Gusfield and Irving [1989], Irving [1994]. Moreover, we study both the cases of symmetric and asymmetric preferences of agents towards each other. Stability is one of the main measures in our investigation. We check whether our method yields stable results and then to which stable matching the method will converge if there are multiple stable matchings. As we have a dynamic decentralized system with agents having incomplete information, it is hard to guarantee stability for every instance. For unstable outcomes, we check instability with three measures: the degree of instability calculates the number of blocking agents, i.e., those agents who are part of blocking pairs Roth and Xing [1997]; the ratio of instability gives the proportion of blocking pairs out of all possible pairs Eriksson and Häggström [2008]; and maximum dissatisfaction, which is the maximum difference between an agent's current utility and their obtainable utility by being part of the blocking pair. Overall, we found that many of our outcomes are stable, or if not, they are close to stable. Also, it is easy to get stable outcomes for instances with symmetric preferences and harder for asymmetric ones.

It is important for the outcome to be fair to all the agents, as the goal of the agents is to increase only their own happiness. Therefore, we use three measures of fairness: set-equality cost, regret cost, and egalitarian cost Gusfield and Irving [1989]. We compare the fairness of our results to those of bidirectional local search, a centralized approach, and two decentralized approaches: Hoepman's algorithm Hoepman [2004], and a decentralized algorithm by Comola and Fafchamps. Note that these algorithms solve the much easier, non-spatial problems, usually with the assumptions of complete information on the part of agents. Nonetheless, our approach performs competitively in terms of fairness. Lastly, similar to Echenique and Yariv [2012], we check the proportion of overall median stable matchings, as well as individual median matchings in our results, which are other important measures of fairness.

Related Work

Reinforcement Learning has not been used for the decentralized two-sided stable matching problem, but researchers have applied both RL and Deep RL mechanisms to solve coalition formation problems. This is closest to our work as matching problems are a special case of coalition formation problems. In Bachrach et al. [2020], Bachrach et al. proposed a framework for training agents to negotiate and form teams using deep reinforcement learning. They have also formulated the problem spatially. Bayesian reinforcement learning has also been used for coalition formation problems Chalkiadakis and Boutilier [2004], Matthews et al. [2012]. Unlike most of the work in MARL approaches for coalition formation and task allocation, our agents cannot communicate with each other (although they can observe other agents in the same cell). Nonetheless, their utilities get affected by the actions of other agents.

Researchers have studied several decentralized matching markets, and have proposed frameworks for modeling them and techniques for solving them, and have also analyzed different factors that affect the results Niederle and Yariv [2007, 2009], Satterthwaite and Shneyerov [2007], Haeringer and Wooders [2011], Echenique and Yariv [2012], Pais et al. [2012, 2017], Diamantoudi et al. [2015]. Most of these works focus on job markets. Echenique and Yariv, in their study of one-to-one matching markets, proposed a decentralized approach for which stable outcomes are prevalent, but unlike our formulation of a problem, agents have complete information of everyone's preferences Echenique and Yariv [2012]. Unlike our work, none of these works have formulated the problem spatially, and also, they have used different methods than RL. Some distributed algorithms for weighted matching include algorithms that are distributed in terms of agents acting on their own either synchronously or asynchronously Hoepman [2004], Wattenhofer and Wattenhofer [2004], Khan et al. [2016]. The crucial assumption in these works is that agents already know their preferences over the members of the opposite set and can directly contact other agents to propose matches.

Most of the decentralized methods mentioned here allow agents to make matching offers and accept/reject such offers. While in our approach, an agent shows interest in pairing with an agent from the other set that is present at the same location by selecting a relevant action. The agent's state space represents those agents from the opposite set that are present at the same cell location and also which ones among them are interested in pairing. Agents get matched only when both agents select an action for pairing with each other. While this may seem similar to making, accepting, or rejecting offers, it is not exactly the same.

Preliminaries

Our two-sided stable matching problem consists of n agents divided equally into two disjoint sets S_1 and S_2. These agents are placed randomly on the grid with dimensions $H \times L$. We investigate if agents can learn good matching policies in a decentralized, spatial setting.

Definition 12. *In the classical two-sided* ***stable matching problem (SM)****, each agent has a strict preference order p over the members of the other set. Given matching M, the pair (i, j) with an agent $i \in S_1$ and an agent $j \in S_2$ is a blocking pair for M, if i prefers j and j prefers i to their respective partners in M. A matching is said to be stable if it does not contain any blocking pairs.*

The preferences are expressed as weights and hence referred to as weighted preferences. A weight/score represents the true utility value an agent may receive by being in a particular match. These weights still correspond to a strict preference order for each agent. An agent only gets to know the utility from a match when it is in that particular match. Even then, it only receives a noisy utility value for that match rather than the true, underlying utility value. It can be formally written as: for $i \in S_1$ and $j \in S_2$, agent i receives the utility $U_{ij} \cdot C$ for being in a match with j, where C is the noise, sampled from a normal distribution with mean $\mu = 1$ and standard deviation $\sigma = 0.1$ and U_{ij} is the true utility value that agent i can get from a match with agent j. Agents still have a strict preference order p over the agents on the other side. U_{ij} is picked uniformly from range $[k, l] \in \mathbb{Z}$, while maintaining the strict preference order.

We also consider the following two extensions of the SM problem.

Definition 13. *The* ***stable matching problem with incomplete preference lists (SMI)*** *may have incomplete preference lists for those involved. In this case, the members of the opposite set who are unacceptable to an agent simply do not appear in their preference list Gusfield and Irving [1989].*

As we have a score-based formulation, an agent has negative scores for unacceptable agents of the other set.

Definition 14. *Some agents may be indifferent (i.e., have the same utility) between two or more members of the opposite set. This is called the* ***stable matching problem with ties (SMT)*** *Gusfield and Irving [1989], Irving [1994].*

We consider two types of preferences among agents: symmetric and asymmetric.

For $i \in S_1$ and $j \in S_2$, let $p_i(j)$ ($p_j(i)$, respectively) denote the position of i in j's preference list (the position of j in i's preference list, respectively). In **symmetric preferences**, $p_i(j) = p_j(i)$ (in our case, $U_{ij} = U_{ji}$ as well), which is not guaranteed to be the case in **asymmetric preferences**. With asymmetric preferences (similar to random preferences in literature), there can be many different stable matchings in a market. However, in the case of symmetric preferences, there can be only one stable matching where each agent gets their best choice. Our environment is dynamic and uncertain, and also, due to the narrow difference between noisy utilities, it can be hard for agents to discriminate between their choices efficiently. This can cause unstable outcomes, especially for asymmetric preferences. Therefore, if a stable outcome does not emerge, then we investigate the nature of instability with the following three measures.

Definition 15. *The **degree of instability (DoI)** of the matching is the number of blocking agents, i.e., the agents that belong to some blocking pair Roth and Xing [1997].*

Eriksson and Häggström pointed out that, rather than only looking for the number of blocking agents, it can also be helpful to look at the number of blocking partners of an agent, as it gives insight into how likely the agent will exploit instability Eriksson and Häggström [2008]. Their notion of instability is defined as follows.

Definition 16. *For any matching M under preference structure $P^{(m)}$ on a set of m agents, let $B_P^{(m)}(M)$ denote the number of blocking pairs. Let $\hat{B}_P^{(m)}(M)$ denote the proportion of blocking pairs: $\hat{B}_P^{(m)}(M) = B_P^{(m)}(M)\ /\ m^2$ Eriksson and Häggström [2008].*

While Eriksson and Häggström call this measure the 'instability' of the matching M, we call it the **ratio of instability (RoI)**. We also use a third measure, **maximum dissatisfaction (MD)**. It is inspired by the notion of α-stability in Pini et al. [2013] which is specific to SM with weighted preferences.

Definition 17. *In matching M, for every blocking agent x, let y be their current match and v be their partner in some blocking pair, then*

$$MD(M) = \max_{(x,v)}\{U_{xv} - U_{xy}\}.$$

An increase in this number may lead to the exploitation of instability by agents in the market. Stability in the outcomes does not guarantee fairness. We consider three measures of fairness to check the quality of matchings as given in Gusfield and Irving [1989].

Definition 18. *The **regret cost**, $r(M) = \max_{(i,j)\in M} \max\ \{p_i(j), p_j(i)\}$.*

Definition 19. *The **egalitarian cost**, $c(M) = \sum_{(i,j)\in M} p_i(j) + \sum_{(i,j)\in M} p_j(i)$.*

Definition 20. *The **set-equality cost**, $d(M) = \sum_{(i,j)\in M} p_i(j) - \sum_{(i,j)\in M} p_j(i)$.*

Lower values for these measures indicate better quality of the matchings. Especially, low regret cost and set-equality cost indicate fairness among agents. It is well known that the Gale-Shapley algorithm provides a matching that is optimal for only one side, over all possible stable matchings. Thus, one notion of fairness is to consider the median of the set of stable matchings, so as to privilege neither set over the other. Thus, similar to Echenique and Yariv [2012], we check whether the final matchings are median stable matchings (MSM), and overall, what proportion of individual matches are median matches (MM). The well-known median property is first discovered by Conway Gusfield and Irving [1989]. A median matching exists whenever there is an odd number of stable outcomes. It is the matching that is in the middle of the two sides' orders of preference. Thus, the median stable matching represents some sense of fairness as it balances the interests of both sides.

Definition 21. *Let P be a preference profile with the set of stable matchings $S(P)$. If $K = |S(P)|$ is odd, the* ***median stable matching (MSM)*** *is a matching $M \in S(P)$ such that for all agents $a \in S_1 \cup S_2$, $M(a)$ occupies the $\frac{K+1}{2}$th place in a's preference among the agents in $\{M'(a)|M' \in S(P)\}$. $M(a)$ is a's median partner among a's stable-matching partners Echenique and Yariv [2012]. While MSM is for overall matching,* ***median match (MM)*** *refers to individual matches between the agents, i.e., an individual agent being matched to its median stable match partner.*

Multi-agent Reinforcement Learning

As mentioned earlier, we propose a multi-agent reinforcement learning (MARL) approach that enables each agent to learn independently to find a good match for itself. A reinforcement learning agent learns by interacting with its environment. The agent perceives the state of the environment and takes an action, which causes the environment to transition into a new state at each time step. The agent receives a reward reflecting the quality of each transition. The agent's goal is to maximize the expected cumulative reward over time Sutton and Barto [2018]. In our system, although agents learn independently and separately, their actions affect the environment and in turn affect the learning process of other agents as well. As agents receive separate intrinsic rewards, we modeled our problem as a Markov game. Stochastic/Markov games Littman [1994] are used to model multi-agent decentralized control where the reward function is separate for each agent, as each agent works only towards maximizing its own total reward.

A *Markov game* with n players specifies how the state of an environment changes as the result of the joint actions of n players. The game has a finite set of states S. The observation function $O : S \times \{1, \ldots, n\} \rightarrow R_d$ specifies a d-dimensional view of the state space for each player. We write $O_i = \{o_i | s \in S, o_i = O(s, i)\}$ to denote the observation space of player i. From each state, players take actions from the set $\{A_1, \ldots, A_n\}$ (one per player). The state changes as a result of the joint action $\langle a_1, ..., a_n \rangle \in \langle A_1, ..., A_n \rangle$, according to a stochastic transition function $T : S \times A_1 \times ... \times A_n \rightarrow \Delta(S)$, where $\Delta(S)$ denotes the set of probability distributions over S. Each player receives an individual reward defined as $r_i : S \times A_1 \times ... \times A_n \rightarrow \mathbb{R}$ for player i. In our multi-agent reinforcement learning approach, each agent learns independently, through its own experience, a behavior policy π_i: $O_i \rightarrow \Delta(A_i)$ (denoted $\pi(a_i|o_i)$) based on its observation o_i and reward r_i. Each agent's goal is to find policy π_i which maximizes a long-term discounted reward Sutton and Barto [2018].

Method

We propose a MARL approach for decentralized two-sided stable matching problems that are formulated spatially on a grid. For each agent, the starting location is picked uniformly randomly from the grid cells. As agents go through episodic training, they start in this same cell location in each episode and explore the environment. Agents must first find each other before they can potentially form matches.

This approximates the spatial reality of meeting with individuals (at, e.g., bars or parties) or organizations (at, e.g., job fairs).

We believe that finding a partner for oneself is an independent task, where agents do not necessarily need to compete or even cooperate. Agents only need to learn to find a suitable partner. Each agent independently learns a policy using the RL algorithm, SARSA Rummery and Niranjan [1994], Sutton and Barto [2018], with a multi-layer perceptron as a function approximator to learn the set of Q-values. An agent's learning is independent of other agents' learning as all the agents have separate learning modules (neural networks). We use SARSA because it is an on-policy algorithm in which agents improve on the current policy. Unlike off-policy algorithms like deep Q-learning where agents' behavior while learning can be erratic due to inconsistencies in the policy, on-policy algorithms follow the same policy and improve on it, which is useful when the agent's exploratory behavior matters. In real-world matching markets, there is value to the path of finding a final match. SARSA is also a model-free algorithm, so that agents directly learn policies, without having to learn the model.

While exploring, agents cannot perceive any part of the environment other than their cell location. If an agent encounters another agent from the opposite set in the same cell and both agents show interest in matching with each other, at the same time step, then they get matched. As agents can only view their current grid cell, agents can only match with one another if they are in the same cell. As long as agents are matched, they receive a noisy reward as a utility value at each time step. This noise is sampled from the normal distribution and the true utility value is multiplied by this noise. Note that our environment is deterministic. We now describe the agents' observation space, action space, and reward function.

Observation space: An observation O_i for an agent i at time step t, let's say $O_i[t]$, consists of three one-hot vectors. The first one represents an agent's position on the grid, the second vector represents which members of the opposite set are present in the current cell, and the third one shows if any of those agents are interested in forming a match. The size of O_i is $R \times C + 2 \cdot m$, where R and C are the number of rows and columns in the grid and m is the total number of members of the opposite set. The size of the first hot vector is equal to the total number of grid cells, and the size of the second and third vectors is equal to the size of the opposite set. Thus, an agent initially starts out knowing only the dimensions of the grid and the total number of agents in the opposite set.

Action space: There are two types of actions available to an agent: navigating the grid and expressing an interest in matching with an agent from the opposite set. The action space is of size $m + 4$, where m is the size of the opposite set and each member has an action associated with it for showing an interest in matching with that member. There are 4 additional actions for navigating the grid by moving up, down, left, and right. There is no specific action for staying in the same grid cell because whenever an agent is interested in forming a match with another agent, it automatically stays in the same cell. When two collocated agents show an interest in forming a match with one another, then the match is considered to be formed.

Note that once a match is formed, the agents must continue to express interest in each other at each time step in order to maintain the match. If at some point, one ceases to express interest, the match is dissolved.

Reward Function: We have a noisy reward function described as: (1) -1 reward for not being in a match. (2) The immediate reward received by an agent i for the matching of agents i and j is $R_{ij} = U_{ij} \cdot C$, where C is the noise, sampled from a normal distribution with mean $\mu = 1$ and standard deviation $\sigma = 0.1$ and U_{ij} is the true utility value that agent i can get from a match with agent j.

Agents have prior knowledge of the grid size and the total number of agents in the opposite set because of the way the states are constructed. However, they completely lack knowledge of the weighted preferences/utility values of other agents. Furthermore, agents only get to know their own utility for an agent on the other side when they get into a match with it, and that utility value is noisy. In our setup, individuals may choose to be in a match until someone better comes along or may choose to leave a match in order to explore further and look for someone better. Thus, a time step in which all agents are paired is not necessarily *stable*, because agents may break off a partnership to explore, or another, more appealing agent may be willing to partner with them.

Experiments

In this section, we present the ways we tested our approach on stable matching problems. Our main focus is on investigating the applicability of our MARL approach. Along with the classical stable matching case (SM), we examine how MARL performs on variations such as stable matching with incomplete lists (SMI) and ties (SMT). We consider two types of preferences among agents: symmetric and asymmetric. As mentioned earlier, agents have weighted preferences over agents on the other side. For an agent $i \in S_1$, it can be seen as the utility value U_{ij} that it gets while in a match with agent $j \in S_2$. In the case of SM and SMT problems, these weights are generated from a uniform random distribution in the range $[1, 10]$; for SMI, the weights are generated from a uniform random distribution in the range $[-10, 10]$ (negative weights indicate how much one agent dislikes the other). For SM and SMI problems, the instances where agents have weights reflecting the strictly ordered preferences are chosen for the experiments. This constraint is removed while choosing SMT instances.

As we formulate the problems on a grid, we investigate results for increasingly complex environments. This complexity is in terms of grid size and the number of agents. We use grid sizes 3×3, 4×4, and 5×5 in combination with 8, 10, 12, and 14 agents as follows: (1) Grid: 3×3 ; Agents: 8; (2) Grid: 4×4 ; Agents: 8, 10, 12, 14; (3) Grid: 5×5 ; Agents: 8, 10, 12, 14. We do not place more than 8 agents on a 3×3 grid to keep a reasonable density of the population. We chose grid sizes such that agents find other agents easily accessible. This is motivated by real-world places like bars, parties, job fairs, etc. We think that our choices of grid size and the number of agents are sufficient to get the essence of realistic situations.

Starting cell locations of agents are chosen uniformly randomly from the grid cells, and agents are placed back to these locations at the start of each episode. We run experiments for every possible combination of matching problem variation (SM, SMI, and SMT), preference type (symmetric and asymmetric), grid size, and total agents. We implement 10 different instances of each of these combinations. Each instance is generated by assigning weights between the agents uniformly randomly while still maintaining the preference order if needed.

Parameter Settings: Each agent independently learns a policy using SARSA Rummery and Niranjan [1994], Sutton and Barto [2018] with a multi-layer perceptron as a function approximator to learn a set of Q-values. Each network consists of 2 hidden layers with 50 and 25 hidden units, respectively. We trained models using the Adam optimizer Kingma and Ba [2014] with a learning rate 10^{-4} to minimize TD-control loss. We used the discount factor, $\gamma = 0.9$. We have combined SARSA with experience replay for better results. The use of experience replay along with SARSA has been proposed by Zhao et al. Zhao et al. [2016]. As SARSA is an on-policy algorithm, we only used data from recent (last 10) episodes in our experience replay buffer, which increased our performance over not using a replay buffer. The number of training episodes and steps varies based on grid size and the total number of agents in an instance. The number of steps per episode varies between 300–700 and training can take between 100k to 400k episodes to converge. When there are multiple suitable matches available in the environment for an agent, a proper exploration strategy is needed to find the best among them. Therefore, we used an exploration rate with non-linear decay, such that it is high in the beginning but decays later (with a minimum exploration rate, $\epsilon = 0.05$). The learning rate and discount factor are fine-tuned as the outcomes are slightly sensitive to these hyper-parameters; however, results are robust to the changes in other hyper-parameters.

We investigate the stability and fairness of the outcomes. Roth hypothesized that the success of a centralized labor market depends on whether the matchmaking mechanism generates a stable matching Roth [1991]. Although we have a decentralized matching market, we think that stability is still an important measure of success. For the *SM* problem, stable matchings always exist, and for the *SMI* and *SMT* problems, at least a weakly stable matching exists Iwama and Miyazaki [2008]. In weak stability, a blocking pair is defined as (i, j) such that $M(i) \neq j, j \succ_i M(i)$, and $i \succ_j M(j)$ Iwama and Miyazaki [2008]. Note that in *SMI* instances, agents can end up without a partner as incomplete lists make some potential matches unacceptable.

As we have a dynamic and uncertain environment and agents with incomplete knowledge, there is a scope for the rise of instability. Economic experiments on decentralized matching markets with incomplete information Ünver [2005], Niederle and Roth [2006] have yielded outcomes with considerable instability. We use three more measures to study instability: the degree of instability (DoI), the ratio of instability (RoI), and maximum dissatisfaction (MD) (details in Preliminaries). Stable or close-to-stable solutions do not guarantee fairness, specifically for asymmetric preference cases. As agents are independent and autonomous, we need to check the efficacy of our approach from an individual agent's point of view. Therefore, we use three fairness measures: set-equality cost, regret cost, and egalitarian cost. Additionally, we check the proportion of both median stable matchings as well as individual median matches. We compare our results with both centralized and decentralized algorithms. The comparison baselines are detailed below.

Bidirectional Local Search Algorithm (BLS) Viet et al. [2016] is a centralized local search algorithm for stable matching with set equality. It uses the Gale-Shapley algorithm Gale and Shapley [1962] to compute S_1-optimal and S_2-optimal stable matchings and executes a bi-directional search from those matchings until the search frontiers meet.

Hoepman's Algorithm (HA) Hoepman [2004] is a variant of the sequential greedy algorithm Preis [1999] which computes a weighted matching at most a factor of two away from the maximum. It is a distributed algorithm in which agents asynchronously message each other.

Decentralized Algorithm by Comola and Fafchamps (D-CF) Comola and Fafchamps [2018] is designed to compute a matching in a decentralized market with deferred acceptance. Deferred acceptance means an agent can be paired with several other partners in the process of reaching their final match. This algorithm includes a sequence of rounds in which agents take turns in making proposals to other agents, who can accept or reject them. While Comola and Fafchamps focused on many-to-many matching, the method can be easily adapted for one-to-one matching.

Note that not only BLS but also HA and D-CF are non-spatial algorithms where agents already have knowledge of every other agent present in the system. This gives them a significant advantage over the agents in our system, both because the agents know whom they prefer and because they have instantaneous contact, rather than having to wander around in a grid world. Both of the decentralized algorithms use randomness while forming their final matching, giving different results each time. Therefore, we run each instance 5 times and compare to the average of those runs. We also run our MARL approach 5 times for each instance. We discovered that if a stable outcome is found, the same one is found consistently, but if not, then the outcomes vary.

Results

We evaluate the results for stability, as well as the level of instability for unstable outcomes. We use three measures to evaluate instability: the degree of instability (DoI), the ratio of instability (RoI), and maximum dissatisfaction (MD). As fairness in the outcomes is also important, we use three fairness measures: set equality cost, regret cost, and egalitarian cost. In addition to this, we check what percent of the stable matchings are median stable matchings, as well as what percent of the individual matches are median matches. Results for SM and SMT problems with symmetric preferences and for SMI problem with both symmetric and asymmetric preferences are straightforward, therefore, are mentioned in the text. However, the results for SM and SMT problems with asymmetric preferences needed more

Grid	3×3	4×4				5×5			
Agents	8	8	10	12	14	8	10	12	14
Stability(%)	100	92.0	82.0	68.0	56.0	80.0	74.0	54.0	46.0
DoI	0	2 ± 0.0	2 ± 0.0	3.3 ± 1.3	3.2 ± 1.7	2 ± 0.0	2.5 ± 1.1	2.8 ± 0.9	3.5 ± 1.6
RoI	0	0.04 ± 0.0	0.04 ± 0.0	0.06 ± 0.01	0.07 ± 0.02	0.04 ± 0.0	0.04 ± 0.01	0.05 ± 0.02	0.07 ± 0.03
MD	0	1.75 ± 0.9	2.89 ± 1.7	3.13 ± 1.9	3.89 ± 1.8	2.33 ± 1.5	2.77 ± 1.4	3.25 ± 1.9	4.44 ± 2.3
MM(%)	83.1	73.2	65.3	63.4	52.4	75.0	67.1	58.9	48.7

Table 3.3: For SM (asymmetric) case, MARL results on stability (%), instability measures $(Avg \pm Std)$ and median matches (%).

analysis. We elaborate on the results of SM problem with asymmetric preferences in Tables 3.3–3.5, as we think that this is the most relevant and adverse case. Due to lack of space, we omitted a similar analysis of the results for the SMT-asymmetric case; however, those results are very similar to the ones presented for the SM-asymmetric case.

Many of our outcomes are stable or close to stable. For SM problem with symmetric preferences, there is only one possible stable matching, and all the outcomes converge to that. However, for asymmetric preferences, more than one stable matching is possible. The instances with symmetric preferences converge faster than the asymmetric ones. The instances of SM and SMT with asymmetric preferences take longer to converge, with lower rates of convergence to stability. The results of SM and SMT are similar. Additionally, for SM asymmetric instances, we have observed that the agents disliked by everyone in the opposite set (low utility associated with them by everyone) find it difficult to get a long-term match. Similarly, unsurprisingly, the most-liked agent (high utility associated with them by everyone) easily settles with its ideal match. We also noticed that the noise in utilities adversely affects convergence to stable outcomes.

When it comes to SMI, our results are always stable. The number of agents that are matched is the maximum possible. This is important because when agents have incomplete lists (negative utilities for matches), it is hard to get a match for everyone, even though it is easier for some agents to find a stable partner due to fewer choices. Here, the final outcome always has the lowest regret cost. Importantly, between the agents in the matched pair, there can be an agent having zero utility towards its match, while the other agent still has positive utility for the same match. As the agent with positive utility tries to get in a match, having noise in the reward causes the agent with zero utility to stick to the match. Note that this does not happen

N	Set-equality Cost; $d(M)$					Regret Cost; $r(M)$				
	MARL (4×4)	MARL (5×5)	BLS	HA	D-CF	MARL (4×4)	MARL (5×5)	BLS	HA	D-CF
8	3.1 ± 2.4	3.9 ± 2.5	2.9 ± 2.1	2.6 ± 1.8	3.1 ± 1.7	3.6 ± 0.8	3.5 ± 0.8	3.5 ± 0.8	3.7 ± 0.7	3.5 ± 0.8
10	2.9 ± 2.1	3.2 ± 2.8	3 ± 2.8	3.5 ± 1.6	4 ± 2.7	4.3 ± 0.8	4.1 ± 0.7	4 ± 0.8	4.6 ± 0.5	4.2 ± 0.9
12	4.6 ± 3.5	6.6 ± 3.8	5 ± 4.2	4 ± 4.2	4 ± 4.2	5.4 ± 0.8	5.3 ± 0.9	5.1 ± 0.9	5.3 ± 1.1	5.1 ± 0.9
14	7 ± 9.2	7.4 ± 7.7	7.5 ± 4.9	7.2 ± 4.4	7.5 ± 4.9	6.7 ± 0.7	5.9 ± 1.3	5.7 ± 1.1	5.6 ± 1.0	5.7 ± 1.1

Table 3.4: For SM (asymmetric) case, comparison of set-equality cost and regret cost in $(Avg \pm Std)$ format; results for 8 agents on 3×3 grid not included due to limited space.

N	Egalitarian Cost; $c(M)$				
	MARL (4×4)	MARL (5×5)	BLS	HA	D-CF
8	15.3 ± 2.8	15.1 ± 2.3	15.5 ± 2.7	16.6 ± 2.8	15.5 ± 2.7
10	20.3 ± 2.7	20.4 ± 3.4	19.8 ± 2.8	25.1 ± 3.7	20.4 ± 2.9
12	31 ± 5.9	28.4 ± 4.6	27.6 ± 3.4	32.6 ± 5.1	27.8 ± 3.3
14	41.4 ± 6.1	39.4 ± 5.6	34.9 ± 3.5	41.2 ± 6.9	34.9 ± 3.5

Table 3.5: For SM (asymmetric) case, comparison of egalitarian cost in $(Avg \pm Std)$ format.

when both the agents in the match have zero utility for the match, as neither of them tries to stick with the match.

From Table 3.3, which elaborates on the results of the SM-asymmetric case, we can see the curse of dimensionality in how the number of agents affects stability. Although the grid size also affects stability, its impact is much less. Both of these factors affect the convergence rate as well: more complex environments take longer to converge. The environment with 8 agents on a 3×3 grid is the easiest one for training agents, and 100% of the outcomes are stable, while the one with 14 agents on a 5×5 grid is the hardest to train and the stability of the final outcomes declined significantly to 46%. Nonetheless, we can also see from the measures of instability that the outcomes are close-to-stable. Note that in Table 3.3 the values associated with these measures are averaged over only unstable outcomes. The average number of blocking agents (DoI) is low in all cases. We also checked the proportion of blocking pairs (RoI), as the greater this number, the more likely that blocking agents will discover and exploit the instability at some point Eriksson and Häggström [2008]. Our approach does well for this measure. This follows the suggestion by Eriksson and Häggström that if agents increase the search effort rather than picking random partners, then we can expect outcomes to have a very small proportion of blocking pairs Eriksson and Häggström [2008].

Furthermore, we look at the maximum dissatisfaction (MD) that an agent can have for an outcome, as great dissatisfaction may also lead to exploiting instability. This number is also low, which assures that there is a low likelihood of blocking agents exploiting unstable outcomes in the market. We think that the dynamic and uncertain environment, incomplete information, noisy utilities, and the narrow differences in the utilities between matches found for an individual over different episodes are potential reasons behind the emergence of instability in the outcomes. Especially in the case of asymmetric preferences, it is unlikely that an agent's ideal partner also best prefers that agent.

In Tables 3.4 and 3.5, we compare fairness in the outcomes with three other algorithms. Here, we can see that MARL performs competitively, and there is no significant difference between the fairness results. The regret cost of MARL is slightly, but not significantly, higher for all the types of instances. Hoepman's algorithm (HA) and the decentralized algorithm by Comola and Fafchamps (D-CF) are decentralized approaches. While D-CF always produces stable outcomes, that may not be the case with HA. Our approach performed better than HA in almost all cases and was very similar to the D-CF algorithm. Again, our approach performs well despite being implemented on a fundamentally more complex formulation of the problem than the ones for HA and D-CF. Further, when our outcomes are stable, they usually match with those found by BLS. It shows that despite the decentralization, our MARL approach is capable of producing outcomes as good as those found by a central agency. This is further supported by the fact that a good proportion of individual matches are median matches (shown in Table 3.3). Also, approximately half of the stable matchings are median stable matchings. We think that fairness is achieved because agents are self-interested and independent, and stability is achieved as agents learn to find their best viable matches.

The learned policies include agents moving to a fixed location from their starting point and getting into a match corresponding to the final outcome. It is possible that more than one pair is formed at the same location, but it is rare. The location where agents in a pair move to form the match is not necessarily the mid-point of the distance between starting points of two agents, nor is it guaranteed to be close to either starting point. Centralized algorithms do not work on a grid; they produce matchings but not the learned policies. This shows that real-world entities can benefit from using our MARL approach to learn to efficiently navigate the environment in finding and maintaining a good match.

Summary

We have shown that the MARL paradigm can be successfully used for decentralized stable matching problems that are formulated spatially in a dynamic and uncertain environment, with independent and autonomous agents having minimum initial knowledge. Our MARL approach is also applicable for variations such as SM with incomplete lists and ties. Agents tend to be happy with their final matches, as outcomes are stable or close-to-stable and fair for everyone. Even with unstable outcomes, agents are less likely to exploit instability.

Chapter 4 Cournot Games[1]

We investigate using a multi-armed bandit (MAB) setting for modeling repeated Cournot oligopoly games. The firms acting as agents choose from the set of arms representing production quantity. Agents interact with separate bandit problems. An agent can choose from a set of arms/actions representing discrete production quantities; here, the action space is ordered. Agents are independent and autonomous and cannot observe anything from the environment; they can only see their own rewards after taking action and only work towards maximizing these rewards. We first study Cournot models with stationary market demand where random entry or exit from the market is not allowed. Given these assumptions, we found that an ϵ-greedy approach offers a more viable learning mechanism than other traditional MAB approaches, as it does not require any additional knowledge of the system to operate. We also propose two novel approaches that take advantage of the ordered action space: ϵ-greedy+HL and ϵ-greedy+EL. These new approaches help firms focus on more profitable actions by eliminating less profitable choices and are designed to optimize the exploration. However, in real-world scenarios, market demands evolve over a product's lifetime for a myriad of reasons. Therefore, we investigate repeated Cournot games with non-stationary demand such that firms/agents face independent instances of the non-stationary multi-armed bandit problem. We propose a novel algorithm *Adaptive with Weighted Exploration (AWE) ϵ-greedy* which is loosely inspired by the ϵ-greedy approach. We use computer simulations to study the emergence of various equilibria in the outcomes and empirically analyze joint cumulative regrets. Using our proposed method, agents are able to swiftly change their course of action according to the changes in demand and produce collusive outcomes without communicating with each other.

The Cournot oligopoly model is a well-known model in economic game theory. In a standard Cournot game Cournot [1838], firms compete over the production of identical goods. Production or services in various real-world markets can be modeled as Cournot games; for example, energy systems Kirschen and Strbac [2018], transportation networks Bimpikis et al. [2019], and healthcare systems Chletsos and Saiti [2019]. There are three main types of equilibria associated with this model: Walrasian equilibrium, Cournot/Nash equilibrium, and collusive equilibrium (described in Section 4.2). Cournot games have been modeled in several ways in the literature with various assumptions on the firms' cognitive capabilities and rationalities. Different learning mechanisms have been analyzed in order to understand the conditions required to arrive at a specific equilibrium.

We study Cournot models with both stationary and non-stationary demands. Most of the literature on Cournot games assumes stationary demand functions. However, it is also necessary to consider non-stationary demand patterns since, in many real-world market settings, the demand is mostly non-stationary due to

[1]Part of the work in this chapter was published in the proceedings of FLAIRS 2022 Taywade et al. [2022].

factors such as changing seasons, trends, and, as seen most recently, global health crises. For example, the decision problem of fresh food production, where products have limited shelf life, after which the products cannot be sold. The demand for such products is constantly fluctuating for various reasons, such as promotional offers by retailers. We consider three different types of non-stationarities in the market demand. Demand patterns are inspired by the works based on economic markets Berry [1972], Tunc et al. [2011]. We cover both sudden and incremental changes, as well as reoccurring and uncertain changes.

Moreover, in real-world markets, firms often do not have access to knowledge about market demands or their competitors' behavior. This also contradicts many of the assumptions made by prior works. To better model these situations, we consider Cournot games such that the firms/agents are not able to directly observe information about other agents in the environment or the demand function. The only information available to an individual agent is the profit obtained after selecting a production quantity, which is a single play of a Cournot game where firms produce simultaneously.

To incorporate the assumption of low cognitivity, we model repeated Cournot games using a MAB framework, where firms learn independently and are autonomous. Each agent deals with its own multi-armed bandit problem separately. This setting provides a practical framework to model the Cournot game with the assumptions of low cognitivity. Here, information on competitors is not available to the firms, and they cannot deduce the demand function. Moreover, firms do not necessarily even need to know their own cost function. In multi-armed bandits, there is only one state, or we can say there is no context, and the Cournot game is a one-step game that is played multiple times sequentially; therefore, we model it using a multi-armed bandit framework. The MAB framework facilitates the use of exploration-exploitation approaches. In any MAB problem, dealing with the exploration-exploitation trade-off is crucial. Because of the lack of market knowledge, firms are naturally bound first to explore and learn. In our multi-agent setting, agents learn simultaneously. As learning involves exploration, high uncertainty in the rewards can be seen from the perspective of a single agent having no knowledge of the environment, regardless of the demand being stationary or otherwise.

As multiple agents learn simultaneously, where learning involves exploration, high uncertainty in the rewards can be seen from the perspective of a single agent having no knowledge of the environment. This uncertainty can vary based on agents' exploration-exploitation strategies. However, for models with stationary demand, it is guaranteed that the rewards are generated using the same mechanism every time. We also assume that agents learn simultaneously using the same learning mechanism, and as they learn, their strategies converge. Because of this, we consider these bandits to be stochastic for models with stationary demand. However, for models with non-stationary demand, we consider a non-stationary bandits problem.

One potential issue with using MABs for this problem is that many popular learning algorithms, such as UCB and Thompson sampling, rely on additional knowledge from the environment about reward distributions of the arms, i.e., those methods require the magnitude of the rewards. As mentioned previously, this information is unavailable to our agents while they are learning. While these methods are unsuitable for our goals, ϵ-greedy approaches do not require any additional knowledge about the problem. Therefore, all our approaches are inspired by the ϵ-greedy strategy. However, one potential disadvantage of using traditional ϵ-greedy approaches is that the learning process can slow down as the exploration space increases, limiting how well these algorithms scale with larger exploration spaces. To address this limitation, our proposed methods take advantage of the ordered nature of the action space to optimize the exploration. For models with stationary demand, we propose two novel approaches. The first approach is similar to hierarchical learning; we call it ϵ-greedy+HL. It is based on partitioning the action space into several buckets and choosing the best bucket. The second method resorts to action/arm elimination to reduce the action space, and hence is called ϵ-greedy+EL; EL stands for elimination. In real-world scenarios, the exploration can be costly, and the operating mechanism of our proposed approaches can significantly reduce the exploration cost. Therefore, these newly proposed approaches are more adaptable for real-world usage and also applicable to other similar problems with ordered action spaces, such as dynamic pricing.

In Cournot games with non-stationary demand, the reward distribution changes with time. With changing reward distribution, the optimal action also changes, and it is necessary to manage exploration-exploitation properly to discover the new optimal action. In this case, it is not practical to completely eliminate actions as we did in two proposed approaches for Cournot games with stationary demand. Therefore, for models with non-stationary demand, we propose a novel approach called *Adaptive with Weighted Exploration (AWE) ϵ-Greedy*. It is also remotely based on the ϵ-greedy strategy. Here, each agent quantifies changes in the reward and uses them to adjust both exploration and learning rates. This enables the agent to quickly discover new optimal arms in response to changes in perceived reward distributions. Furthermore, this approach utilizes a weighted sampling method that allows it to reduce the search space quickly. This mechanism also can be used for problems with ordered action space, such as the one considered in this work. It can also deal with the uncertainty in rewards caused by the activities of other agents operating in the system.

We evaluate our proposed approaches empirically by running different kinds of simulations. We study the resulting equilibria, which signify the degree of success achieved by agents in learning to navigate through a multi-agent system. We also study the effects of scaling the models on the resulting equilibria; we scale the models in terms of the number of agents in the environment and the action space. Furthermore, we study the sensitivity of our proposed methods to different hyperparameter values. In addition to Cournot models with symmetric firms (same marginal cost), we also study models with asymmetric firms (different marginal costs). With Cournot models with stationary demand, we consider the deterministic demand function. In experiments, we start with simulations the same as that used in Waltman and Kaymak [2008], Xu [2020]; these are small-scale simulations containing between two and six firms. We further scale the models in terms of the number of firms and the action space size. We investigate our outcomes concerning three types of equilibria associated with Cournot games: Walrasian equilibrium, Cournot/Nash equilibrium, and collusive equilibrium. For models with stationary demand, we compare joint quantity, joint profit, and joint cumulative regret obtained by ϵ-greedy and our proposed methods. Joint cumulative regret is the difference between the joint profit obtained by agents over the course of learning and the joint profit that could have been obtained if agents acted as a cartel throughout the process. For models with non-stationary demand, we use the adaptive ϵ-greedy algorithm dos Santos Mignon and da Rocha [2017] as a baseline to compare our results; this algorithm is somewhat similar to our approach as it is also based on the ϵ-greedy strategy.

For simulations with stationary demand, collusion in various degrees can be seen, in small-scale settings, with a limited number of firms and action choices. Here, in general, outcomes obtained by our proposed methods are more collusive than baseline ϵ-greedy, although full collusion usually does not emerge. By collusive behavior, we mean outputs anywhere between the Nash and collusive equilibria. Although the real-world scenarios are more complex than our simulations, the results still indicate the possibility of the emergence of collusive behavior, even without explicit instructions to collude.

We also evaluate the quality of our outcomes using joint cumulative regret, a metric not actually available to the individual agents. The joint cumulative regret obtained by ϵ-greedy+HL is the lowest, followed by that obtained by ϵ-greedy+EL. Moreover, our proposed approaches converge much faster than the baseline ϵ-greedy method. In models with symmetric firms (firms with the same marginal cost), individual agents produce similar quantities; however, asymmetry in firms can cause them to produce outcomes with substantial differences. As expected, due to asymmetric costs, firms with lower production costs make higher profits. We further our investigation by running simulations of Cournot models where agents do not use the same learning algorithm but instead are assigned learning algorithms from ϵ-greedy+HL, ϵ-greedy+EL, and baseline ϵ-greedy. In contrast to the models where all agents use the same method, we found that the agents using ϵ-greedy+HL yield lower profits than the agents using the other two learning algorithms. For simulations with non-stationary demand, we found that by using our approach, AWE ϵ-greedy, agents can readily change their action course according to the changing demand. Similar to models with stationary demand, agents show collusive behavior, although full collusion usually does not emerge. However, the collusive behavior declines with an increasing number of firms in the system.

4.1 Related work

In this section, we first discuss the literature on various learning methods for Cournot games. Furthermore, we look at non-stationarity in Cournot games and then the MAB frameworks for modeling non-stationarity. We then look at reinforcement learning approaches for Cournot games and MAB algorithms for dynamic pricing.

Learning algorithms for Cournot Games

Cournot oligopoly games, and their convergence to various equilibria, have been widely studied in past research. The literature establishes that the long-term outcome of a Cournot oligopoly model depends on the underlying learning mechanism, firms' rationality, and the memory size of the firms. The learning of economic agents can be widely divided into two categories: individual learning and social learning Vriend [2000]. In individual learning models, an agent learns exclusively from its own experience, whereas in social learning models, an agent also learns from the experience of other agents. Learning frameworks also reflect the rationality of agents. Several works have studied individual learning mechanisms Vriend [2000], Arifovic and Maschek [2006], Vallée and Yıldızoğlu [2009], Fudenberg et al. [1998], Riechmann [2006]. Some works claim that while social learning leads to the Walrasian equilibrium, individual learning schemes result in convergence to the Nash equilibrium Vriend [2000], Vega-Redondo [1997], Franke [1998], Dawid [2011], Bischi et al. [2015]. In this work, we use MAB problem Thompson [1933] as a learning framework. Firms make decisions based on the profit/reward they are getting, mostly unaware of the impact of other firms' actions on the market price; this can be described as implicit individual learning.

Non-stationarity in Cournot Games

Literature based on learning methods in Cournot games mostly assumes stationary demand. However, in real-world scenarios, non-stationarity is integral to economic markets. Depending on the market type, demand changes follow various patterns, i.e., small and slow changes to large and sharp changes. Several works study the inventory management model where companies have to decide efficiently about the production or stocking of goods [Graves and Willems, 2008, Silver, 2008, Tunc et al., 2011, Yue et al., 2010, Mohebbi and Choobineh, 2005]. These inventory management models can be seen as a non-stationary version of Cournot games.

Multi-armed Bandit Frameworks for Modelling Non-stationarity

Non-stationary multi-armed bandits are mainly divided into two categories: rested bandits Gittins and Jones [1979], Bouneffouf et al. [2014], Bouneffouf and Féraud [2016], Levine et al. [2017], Seznec et al. [2020] and restless bandits Gafni and Cohen [2018], Liu et al. [2012], Meshram et al. [2018], Besson and Kaufmann [2018], Cheung et al. [2019], Russac et al. [2019], Wei et al. [2016], Seznec et al. [2020]. In the rested bandit case, the underlying distribution changes only when the arm is played. While in the case of restless bandits, the underlying distribution of all the arms changes at every time step according to a known but arbitrary stochastic transition function. Additionally, some problems are categorized as piece-wise stationary Garivier and Moulines [2011], Yu and Mannor [2009], Cao et al. [2019]

in which the reward distributions for the arms are piece-wise stationary and will shift at some unknown time steps. Several approaches have been proposed to deal with different kinds of non-stationarities. Garivier and Moulines proposed sliding window UCB and discounted UCB algorithms. We use these algorithms in our comparison baselines as these can work with normally distributed reward functions. Raj and Kalyani and Cavenaghi et al. proposed discounted Thompson sampling and f-Discounted-Sliding-Window Thompson Sampling, respectively, but these methods assume that the rewards follow the beta distribution. Ghatak and Ghatak et al. proposed Thompson sampling with change-detection and Kolmogorov-Smirnov test based Thompson Sampling algorithms, respectively. While both methods assume that rewards follow Gaussian distribution, those are restricted to two-armed bandit settings only and can not be extended to multi-armed bandit settings. Mellor and Shapiro proposed a family of algorithms, collectively termed as Change-Point Thompson Sampling; however, they also assume that the rewards follow the beta distribution. Liu et al. proposed Change Detection UCB algorithm for piecewise-stationary MAB problems; in contrast, we consider different types of non-stationarities.

Our approach is remotely based on the ϵ-greedy algorithm. Closely related to our work is the work by dos Santos Mignon and da Rocha in which authors propose the adaptive ϵ-greedy algorithm, which modifies the exploration rate based on changes detected in rewards [dos Santos Mignon and da Rocha, 2017]. Our algorithm employs a different mechanism to quantify changes than the ones used in this work. Our algorithm also uses the weighted exploration technique; this technique is specific to Cournot games where the action set is ordered.

Reinforcement Learning for Cournot Games

Researchers have used RL (not MAB) methods to facilitate learning in repeated Cournot games Kimbrough and Lu [2003], Waltman and Kaymak [2008], Xu [2020], but as per our knowledge, all of them consider stationary demand. Waltman and Kaymak analyze the results of Q-learning in a Cournot game with a discrete action space and explain the emergence of collusive behavior. They focus on three types of firms; firms in our experiments are similar to those without memory in their paper. Kimbrough and Lu [2003] also reported results on Q-learning behavior in a Cournot oligopoly game where they found a slight tendency towards collusive behavior in their simulation study. Xu incorporated memory and imitation with RL. In their model, firms do not have any information about market demand, but they can observe quantities produced by the other firms and their profits. They use a continuous action space and parametric function approximation. They use three different settings, the first of which (Treatment 1 in their paper) resembles our models. They have used the same experimental setup as in Waltman and Kaymak [2008]. Unlike our results, they observe convergence to the Nash equilibrium for these settings. Both of these papers evaluate scalability up to a limited range. In contrast, our work incorporates a more thorough investigation of scalability in terms of the number of firms and the size of the action space. Huck et al. studied another trial-and-error method. In their work, firms do not have information about their rivals and the payoff function of the game. However, their method allows agents only to decrease or increase the production level; therefore, firms have a great deal of responsibility in choosing the starting quantity. They showed that full collusion could be achieved with their approach; however, they did not study the scalability of their approach.

Multi-armed Bandit Algorithms for Dynamic Pricing

We think that the dynamic pricing problem is similar to Cournot games. There is a wide variety of work that uses RL for dynamic pricing problems. The market models considered in these works are diverse Den Boer [2015]. Kephart and Tesauro [2000] use Q-learning in one of such settings, and Könönen [2006] use Q-learning with function approximation as well as policy gradient method. Misra et al. [2019] proposed a multi-armed bandit based algorithm for a multi-period dynamic pricing problem where firms face ambiguity. Hansen et al. [2020], Trovo et al. [2015] studied the applicability of MAB algorithms on dynamic pricing problems and proposed variants of the UCB algorithm. Hansen et al. found that with static demand having low noise, it is possible to get collusive behavior in firms, but as the noise increases, the results become indistinguishable from Nash equilibrium. In contrast, we did not notice any significant changes in our outcomes due to the increase in noise.

4.2 Preliminaries

In this section, we have included the formal definition of the Cournot game and described the stationary and non-stationary variants of the market demand. We have also described three equilibria associated with Cournot games, along with the description of the MAB framework.

Cournot Games

We consider a standard Cournot model with n firms offering an identical product. Firms produce independently and simultaneously, and compete on the quantity they produce. In our model, the product quantity is a discrete number. Firm i produces quantity q_i at each time step t. Firm i's total cost is $C_i = cq_i$, where c is the constant marginal cost. The linear demand function with an inverse demand equation is

$$p = max(u - v\sum_{i=1}^{n} q_i, 0) \tag{4.1}$$

where $u > 0$ and $v > 0$ denote two parameters. This is the stationary demand function when parameters u and v are constant. In our models, we still use the same function to calculate the demand; however, parameter u is no longer a constant value. We change it throughout the game to add non-stationarity to the market demand.

The profit of a firm i is calculated as

$$\pi_i = pq_i - C_i \; for \; i = 1, \ldots, n. \quad (4.2)$$

Non-stationarity in Demand

In our second set of experiments, we add three different types of non-stationarities in the underlying demand presented in eq. 4.1. As mentioned above, we make changes in the parameter u to make the demand non-stationary. Parameter v is always constant ($v = 1$) in our models; therefore, u dictates the changes in demand. The patterns emerging from these changes are shown in fig. 4.1. These patterns are inspired by works based on economic markets Berry [1972], Tunc et al. [2011]. We have chosen to explore these patterns as they incorporate some of the common non-stationary behaviors of demand seen in real-world industrial settings, i.e., sudden changes (shocks), slow changes, and more erratic changes in demand. We refer to these demand patterns as patterns 1, 2, and 3, respectively. Unlike other works, we derive the demand pattern solely by changing u. At the start of the game, u is initialized with some value, u_s, and the changes are made to this initial value to change the demand. Fig. 4.1a represents sudden and reoccurring changes in market demand; u_s changes according to following rules:

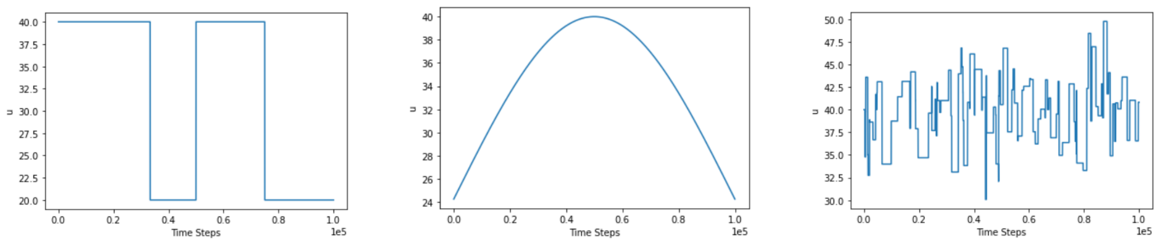
$$u_t = \begin{cases} u_s/2, & \text{when t=T/3} \\ u_s, & \text{when t=T/2} \\ u_s/2, & \text{when t=3T/4} \end{cases}$$

where T is the total number of time steps in a simulation of a repeated Cournot game. For the time steps other than the ones mentioned above, the u_ts remain stationary, i.e., $u_t = u_{t-1}$. Fig. 4.1b represents a sinusoidal pattern with demand changing continuously; there are no sudden changes; instead, the demand changes incrementally and then gradually decreases with small variations. Here, u_t is calculated as follows:

$$u_t = \frac{u_s}{f(T/2|T/2, T/2)} \cdot f(t|T/2, T/2)$$

where f represents the normal probability density function. Both the demand patterns discussed above are dynamic but deterministic. Unlike these patterns, the third pattern, presented in fig. 4.1c, is stochastic. It is an example of an erratic demand pattern which is an extreme case of non-stationarity. Here, u_t is calculated as follows:

$$u_t = \begin{cases} u_{t-1} \cdot |X|, & \text{if } z_t < \gamma \\ u_{t-1}, & \text{otherwise} \end{cases}$$



(a) (b) (c)

Figure 4.1: Examples of three different demand patterns. X-axis represents time steps in the simulation, and Y-axis represents varying values of parameter u (from the inverse demand function $u - vQ$, presented in eq. 4.1). Here, default value for u, $u_s = 40$. Figs. 4.1a, 4.1b, and 4.1c show demand patterns 1, 2, and 3, resp.

where $X \sim \mathcal{N}(\mu, \sigma^2)$, with $\mu = 1$ and $\sigma = 0.2$; $z_t \sim \mathcal{U}(0, 1)$. $\gamma = 0.01$; it dictates the probability of change in demand at any time step t.

There are three main equilibria associated with Cournot games. The following definitions are for models with symmetric firms.

Definition 22. *The* ***Cournot (Nash) Equilibrium*** *is obtained if each firm chooses the production level that maximizes its profit, given the production levels of its competitors. No firm wishes to unilaterally change its output level when the other firms produce the output levels assigned to them in the equilibrium. Firms individually maximize their profit; they do not maximize their joint profit. The resulting joint production level is* $\frac{(u-c)n}{v(n+1)}$.

Definition 23. *The* ***Walrasian Equilibrium*** *is obtained if firms are not aware of their influence on the market price and therefore behave as price takers. They adopt Walrasian rule and produce Walrasian quantity. The Walrasian rule is based on the assumption that a firm acting as a price taker decides next-period output to maximize its profit. The resulting quantity dynamics lead to a dynamic equation that allows the Walrasian equilibrium output as the unique steady state Radi [2017]. The resulting joint production level is* $\frac{(u-c)}{v}$.

Definition 24. *In a* ***Collusive Equilibrium****, firms form a cartel and maximize the joint profit. They produce a smaller quantity than the quantity that maximizes their individual profit. Hence, they have the incentive to increase their production levels. The resulting joint production level is* $\frac{(u-c)}{2v}$.

Multi-armed Bandit Framework

We consider n firms in a repeated Cournot game. Let K_i be the set of arms/actions faced by agent i; arms represent different production choices. The action space is discrete and ordered; every action represents a production level choice, and the action set consists of integer values in a specific range. Agents do not have any knowledge of the environment; agents only have knowledge about their own payoffs/rewards after taking action. As agents are independent and autonomous, they each face a separate non-stationary MAB problem. Each agent's goal is to maximize their individual reward/profit over a long (possibly infinite) time horizon T. At time step t, the reward is calculated using eq. 4.2. From eq. 4.1 and 4.2, we can see that the reward not only depends on demand (or parameter u) but also depends on individual agents' production as well as total industry output at time step t. This adds more uncertainty to the rewards.

4.3 Methods

In our formulation of Cournot games, agents do not have any information about their opponents and environment. They can only see their own profits/rewards. ϵ-Greedy approaches do not require any knowledge about possible reward distributions or the priors of reward distributions, while other MAB approaches usually have this requirement. Therefore, our proposed methods are remotely based on the ϵ-Greedy strategy.

ϵ-Greedy

The ϵ-Greedy method Sutton and Barto [1998] is a common method for balancing exploration and exploitation trade-offs. At each time step $t = 1, 2, ..$, an agent chooses a random action with probability ϵ or otherwise chooses the action with the highest empirical mean. The empirical mean of rewards after taking action a is often referred to as that action's Q-value, denoted as $Q_t(a)$ for time step t. However, in our formulation of the proposed method, $Q_t(a)$ is calculated using learning rate α (inspired by Q-learning algorithm Sutton and Barto [1998]). It is calculated as follows:

$$Q_t(a) = \alpha R_t + (1 - \alpha) Q_{t-1}(a)$$

where R_t is the reward obtained at time step t. A linear bound on the expected regret can be achieved with constant ϵ. For a variant of the algorithm where ϵ decreases with time, Cesa-Bianchi and Fischer [1998] proved poly-logarithmic bounds. However, Vermorel and Mohri [2005] did not find any practical advantage to using these methods in their empirical study. Here, we are referring to regret as it is used conventionally in single agent stochastic MAB problems Kuleshov and Precup [2014], Lattimore and Szepesvári [2020].

Methods for Models with Stationary Demand

In models with stationary demand, every agent faces separate stochastic MAB. As every agent is learning and its actions affect the reward of other agents, these are not perfect stochastic MABs. We feel that this is important to investigate since, as stated in Lattimore and Szepesvári [2020], perfectly stochastic MABs cannot necessarily be expected in the real world. We initially analyzed the applicability of three well-known multi-armed bandit algorithms: ϵ-greedy, UCB (Upper Confidence Bound), and Thompson sampling. Most of the literature deals with Bernoulli bandits; however, we are dealing with normal bandits with unknown means and variances associated with the arms. We found that the UCB-Normal algorithm Cowan et al. [2017], which is designed for such scenarios, fails to converge. Thompson sampling proposed in Honda and Takemura [2014] is also designed to handle such scenarios; however, the outcomes depend on priors. As it is required to set appropriate priors, we cannot use it for our models. We think that ϵ-greedy is the only one among these three algorithms which can work with our assumption of the lack of knowledge about reward distributions. We propose two novel approaches specifically to deal with the ordered action space in the Cournot games. Both of these approaches use ϵ-greedy as a sub-routine.

ϵ-Greedy with Hierarchical Learning (ϵ-greedy+HL)

We propose a new approach similar to hierarchical learning. This approach makes use of the fact that the action set is ordered in Cournot games. Learning happens in multiple phases. In each phase, the ϵ-Greedy algorithm is applied to certain arms. In the initial phase, the whole action set is divided into K equally-sized ranges (buckets). Each action range/bucket is then considered as a single arm. When such an arm is pulled, the action is chosen uniformly at random from the bucket associated with that arm. Each phase ends when the same arm/bucket is chosen during exploitation mode for some pre-specified number of time steps. At the end of each phase, the arm with the highest Q-value is selected, and the associated bucket is further split into smaller-sized buckets. These new smaller buckets are then considered as arms for the next phase of learning. This continues until the bucket cannot be further divided. In this way, underperforming arms are eliminated, and the focus of learning shifts to high-performing arms. Firms can select the value of K as per their choice. This learning process can also be visualized as the search for the best action in a K-ary tree.

ϵ-Greedy with Elimination of Arms (ϵ-greedy+EL)

With hierarchical learning, firms need to make partitions of their action space, which is a crucial but delicate task. To do it efficiently, firms may need some information about the environment. Wrong partitioning may lead to losing out on optimal actions, i.e., a seemingly optimal bucket may contain sub-optimal actions. For the circumstances where firms may not want to take the responsibility of partitioning the action space, we propose another novel approach that is based on eliminating, rather than partitioning, the actions.

This approach also relies on ϵ-greedy as an underlying mechanism and, similar to hierarchical learning, works in multiple phases. Unlike the previous approach, the initial phase starts with considering each action as an arm. Each phase ends when the same action a is chosen during exploitation mode for some pre-specified number of time steps. At the end of each phase, an action a with a maximum Q-value is selected. Let m be the size of the action space. If available, $m/4$ many actions on either side of action a on the ordered scale are kept in the action space, and other actions are eliminated. This continues until $m <= 3$.

Method for Models with Non-stationary Demand

In Cournot games with non-stationary demand, the reward distribution changes with time. With changing reward distribution, optimal action also changes, and it is necessary to manage exploration-exploitation properly to discover the new optimal action. In this case, it is not practical to completely eliminate actions as we did in two proposed approaches for Cournot games with stationary demand. Therefore, for models with non-stationary demand, we propose a novel approach called Adaptive with Weighted Exploration (AWE) ϵ-Greedy.

Adaptive with Weighted Exploration (AWE) ϵ-Greedy

This approach is based on the ϵ-greedy strategy combined with techniques for quantifying changes in rewards and weighted exploration. It changes the exploration rate, ϵ, as well as learning rate, α, dynamically, with the changes in rewards.

In a typical ϵ-greedy approach, a random arm is selected with a small probability ϵ; otherwise, the current optimal arm is selected. Since the optimal arm is selected, overall, at more time steps, we can say that the changes that are detected in rewards of the optimal arm reflect more up-to-date fluctuations in the market demand in comparison to other arms. Therefore, we only focus on the current optimal arm for detecting changes. As agents can see only their own rewards, our change quantification mechanisms are based on changes in rewards. Unlike other works based on change detection Liu et al. [2018], Cao et al. [2019], Ghatak [2020], our algorithm quantifies the change instead of just getting a signal when change is detected. Quantifying the changes helps in making suitable modifications to the learning rate, exploration rate, and weighted exploration.

We quantify changes for two separate purposes: one for modifying α and ϵ, and the other for weighted exploration. For our calculations, we store values that are based on rewards in memory of length M: those are Q-values and the running averages of rewards. Let $\bar{\mu}$ be the mean of any of these two sets of values. We quantify the change with formula: $|(r_t^k - \bar{\mu})/\bar{\mu}|$, where r_t^k is the reward obtained by selecting arm k (current optimal) at time t. The resulting value can be directly used as both α as well as ϵ, given that it is within their pre-specified limits. We did not find much difference in results over which set of values we use. In the algorithm given below, we use Q-values in our calculations.

For weighted exploration, we take advantage of the specific nature of this problem, i.e., the ordered action space, and that the significance of an action is dependent on its distance from the optimal action. While assigning weights to the actions, we use a normal pdf (probability density function). We consider the normal distribution with mean at the optimal action and standard deviation as a quantified change. Here, we quantify changes mainly in two ways: $|(r_t^k - (Q_t^k)|$ and std in Q-values (or the running averages) stored in memory for arm k (used in the algorithm below). In our observation, results are similar for using these techniques. Deriving std in this way ensures that the focus range of exploration narrows down or spans out based on the degree of change. These techniques play a crucial role in deriving the agility of algorithm for changing the course of actions with varying demand. As the received reward deviates from the expected (mean) reward, the exploration rate goes up, and weights associated with the arms tend towards uniformity and vice versa. This helps in promptly finding a new optimal arm and sticking to it until changes in rewards are detected again. If we skip updating both or any of the learning rate and exploration rate, we usually see a delay in changing the action course when the demand changes. If we do not weigh our sampling of actions, then it takes longer to converge to optimal actions for particular time periods, and unimportant actions are explored unnecessarily, which often causes outcomes to drift away from collusion.

4.4 Experiments

We evaluate our proposed approaches empirically by running different kinds of simulations for models with stationary and non-stationary demands. We study the resulting equilibria. We also study the effects of scaling on resulting equilibria; we scale the models in terms of the number of agents in the environment and the action space.

Parameters

We describe the hyper-parameters used in our learning approaches in the following two sub-sections.

Models with Stationary Demand

The exploration rate, ϵ, is one of the main hyper-parameters in ϵ-greedy approaches. In the reported experiments, we use the exploration rate $\epsilon = 0.1$ for all three approaches. We also tried other exploration rates and found that the final outcomes are robust to different exploration rates; however, an increase in joint cumulative regret can be seen with higher exploration rates. Moreover, the decay in the exploration rate does not make any difference in outcomes or the convergence rate. We use the running average of rewards as the Q-values; therefore, the learning rate is not needed. Instead of learning for a pre-specified amount of steps, agents learn until they consistently pick a particular action in exploitation mode for the specific number of steps. For ϵ-greedy+HL and ϵ-greedy+EL, agents have to pick the same arm in exploitation mode for 100 consecutive steps in each phase of the method. However, for baseline ϵ-greedy, agents have to pick the same arm in exploitation mode for 1000 consecutive steps. This difference is because there is no reduction in the size of the action space for baseline ϵ-greedy, and it takes more steps to get to the steady choice. Therefore, we run simulations with a varying number of time steps depending on the scale of the model and the learning approach. By simulation, we mean the entire process of a repeated Cournot game that lasts for several time steps. For ϵ-greedy+HL, we use K (the number of partitions) to be 3. The outcomes in terms of equilibrium are robust to the choice of K; nonetheless, it can affect joint

Algorithm 1 Adaptive with Weighted Exploration (AWE) ϵ-Greedy

Input:
- $K > 2$, number of arms
- $M \in Z$, memory length
- $\epsilon, \epsilon_{min}, \epsilon_{max} \in (0, 1)$, exploration rate and its thresholds
- $\alpha, \alpha_{min}, \alpha_{max} \in (0, 1)$, learning rate and its thresholds

Initialize $Q(k)$ to arbitrary values between $(0, 1)$, for $k = 1 \dots K$.
Initialize $flag = False$ (for tracking if current optimal action is taken)
Initialize $w(k) = Q(k)$, for $k = 1 \dots K$.
for $t \in \{1, \dots, T\}$ **do**
  Sample $rNum \sim \mathcal{U}(0, 1)$
  if $rNum \leq \epsilon$ **then**
    Play arm k_t chosen from arms $1 \dots K$ according to weights $w_t(k)$
  else
    Play arm $k_t = argmax_k(Q_t(k))$; $flag = True$
  end if
  Receive reward r_t^k calculated using eq. 4.2.
  Update $Q_t(k) = \alpha r_t^k + (1 - \alpha) Q_{t-1}(k)$
  if $flag = True$ **then**
    Calculate

$$\bar{\mu} = \frac{1}{M} \sum_{i=t-M}^{t} Q_i(k)$$

$$\hat{\mu} = argmax_k(Q_t(k))$$

$$\hat{\sigma} = \frac{1}{M-1} \sum_{i=t-M}^{t} Q_i(k) - \bar{\mu}$$

    for $k \in \{1, \dots, K\}$ **do**
      Update $w_{t+1}(k) = f(k|\hat{\mu}, \hat{\sigma})$, where f is normal probability density function
    end for
    Calculate $newRate = |(r_t^k - \bar{\mu}) / \bar{\mu}|$
    if $newRate < \epsilon_{min}$ **then** $\epsilon = \epsilon_{min}$
    else if $newRate > \epsilon_{max}$ **then** $\epsilon = \epsilon_{max}$
    else $\epsilon = newRate$
    end if
    if $newRate < \alpha_{min}$ **then** $\alpha = \alpha_{min}$
    else if $newRate > \alpha_{max}$ **then** $\alpha = \alpha_{max}$
    else $\alpha = newRate$
    end if
    $flag = False$
  end if
end for

cumulative regret as the number of phases in the algorithm and total time steps depend on this parameter.

Models with Non-stationary Demand

Here, we run each simulation for $100,000$ time steps. We use the memory length of 10 for all types of simulations. We found that results are moderately sensitive towards memory length, i.e., small changes do not affect the outcomes, but big changes can do so. However, a memory length of 10 helps the method to detect changes in rewards better than any other lengths, and we do not fine-tune it for every type of simulation. Modifications in learning rate α and exploration rate ϵ are confined to certain thresholds. The limits are defined as: $\alpha_{max} = 0.3$, $\alpha_{min} = 0.01$, $\epsilon_{max} = 0.3$, and $\epsilon_{min} = 0.05$. Similar to the memory length, these values are used for all the simulations, and outcomes are moderately sensitive to these hyper-parameters.

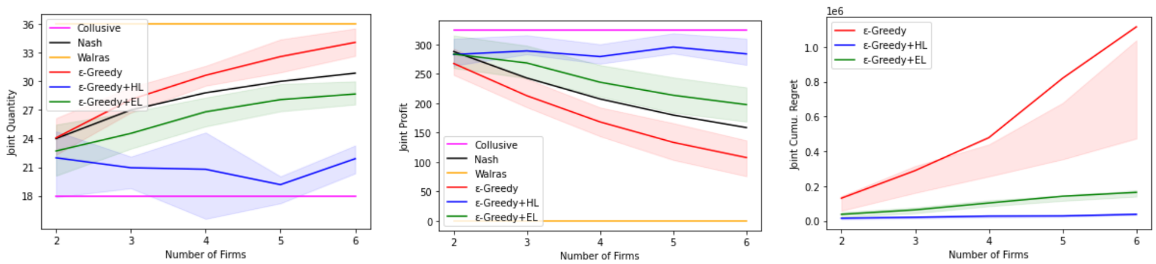
Small-scale Simulations

We first evaluate our approaches using the same simulations as those used by Waltman and Kaymak [2008]. They used Cournot models with symmetric firms in their simulations. They used $u = 40$, $v = 1$ as constants in an inverse demand function (eq. 4.1), along with constant marginal cost, $c = 4$. The action space is discrete, where agents can choose a production quantity between $0 - 40$. They ran simulations with the total number of firms varying from 2 to 6.

Models with Stationary Demand

The results of these simulations are shown in figure 4.2a, 4.2b, and 4.2c for joint quantities, joint profits, and joint cumulative regrets, respectively. From figure 4.2a and 4.2b, we can see that with ϵ-greedy, the outcomes mostly converge somewhere between Nash and Walras equilibrium; however, for the model with 2 firms, output coincides with Nash equilibrium in figure 4.2a. For both ϵ-greedy+HL and ϵ-greedy+EL, outcomes are collusive. However, with ϵ-greedy+HL, those are closer to collusive equilibrium. We consider an outcome to be *collusive* when it is anywhere in between Nash and collusive equilibrium; therefore, the degree of collusion may vary based on its closeness to collusive equilibrium. The outcomes seen in the graphs of joint quantities may not be perfectly reflected in the corresponding graphs of joint profits. This is a consistent outcome that can be seen throughout our experiments. We think that this is because we allow profits/rewards to be negative in our models. Firms lose production costs regardless of the market price. Because of this, there is more variance in joint profits obtained through different simulations. During producing graphs, we averaged the results of 100 different simulations.

From figure 4.2c, we can see that the ϵ-greedy+HL method obtains lowest regret, followed by ϵ-greedy+EL, and ϵ-greedy. ϵ-Greedy obtains very high regret in comparison to the other two methods. One of the reasons is that our proposed approaches take fewer steps to converge than ϵ-greedy. ϵ-greedy+HL on average

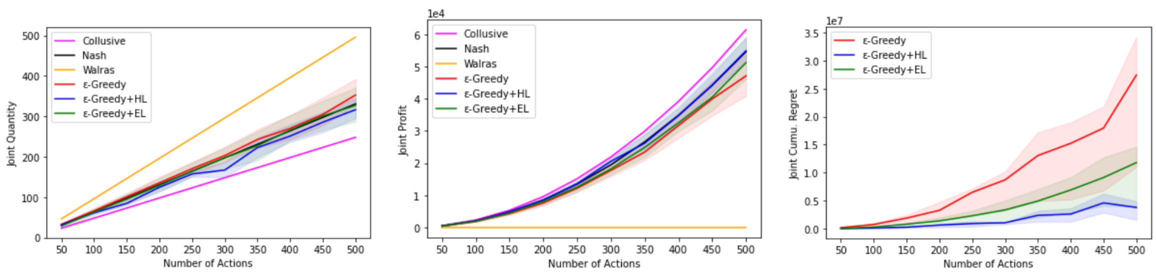


(a) Comparison of joint quantities

(b) Comparison of joint profits

(c) Comparison of joint cumu. regrets

Figure 4.2: For small-scale simulations, comparisons of different measures obtained by ϵ-greedy, ϵ-greedy+HL, and ϵ-greedy+EL algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $v = 40$, $w = 1$, and $c = 4$.



(a) Comparison of joint quantities

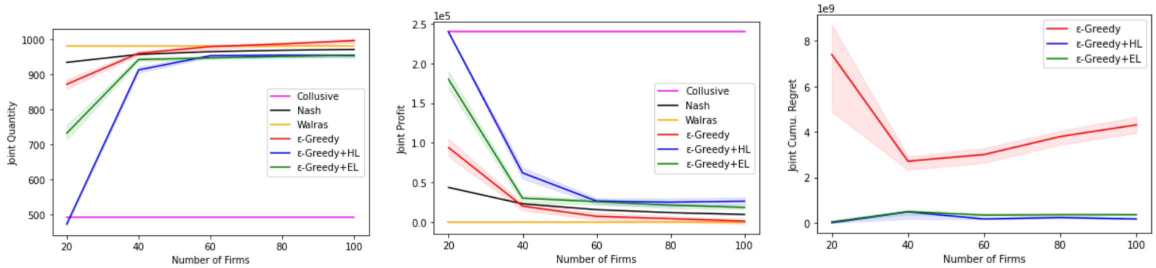
(b) Comparison of joint profits

(c) Comparison of joint cumu. regrets

Figure 4.3: For simulations with scaled actions, comparisons of different measures obtained by ϵ-greedy, ϵ-greedy+HL, and ϵ-greedy+EL algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms. The size of action space, S, varies from 50 to 500, with $u = S$ and $v = 1$, and $c = 4$.

takes only 300 steps while ϵ-greedy+EL takes around 650 steps. The graph comparing the total time steps taken by these three approaches (shown in 4.5a) is very similar to the joint cumulative regrets' graph shown in figure 4.2c. In addition to this, we think that more collusive outcomes and rapid decrease in action space size cause ϵ-greedy+HL to have lower regret than ϵ-greedy+EL.

Models with Mixed Algorithms So far, we have seen models with all the firms using the same learning algorithm. We further investigate the models having agents equipped with different learning algorithms. In figures 4.7 and 4.8, we present results for models where each agent uses either one of the three learning algorithms: ϵ-greedy, ϵ-greedy+HL, and ϵ-greedy+EL. Agents are separated into three same-size groups, and agents in each group are assigned one of the three methods. For example, in a model with 30 agents, three groups with 10 agents each are created, and agents in a group use one of the mentioned algorithms for learning; different groups use different algorithms. We compare the performance of agents

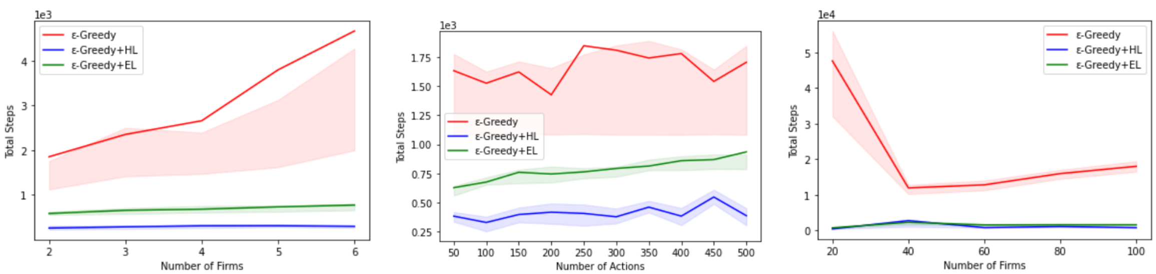


(a) Comparison of joint quantities

(b) Comparison of joint profits

(c) Comparison of joint cumu. regrets

Figure 4.4: For simulations with scaled number of agents, comparisons of different measures obtained by ϵ-greedy, ϵ-greedy+HL, and ϵ-greedy+EL algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $u = 1000$ and $v = 1$, and $c = 20$.

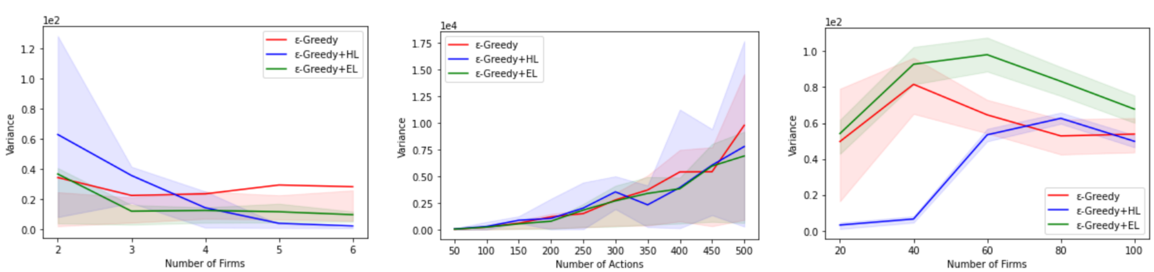


(a) Small-scale simulations

(b) Large-scale simulations (scaled actions)

(c) Large-scale simulations (scaled agents)

Figure 4.5: Comparison of average total time steps taken by agents to converge



(a) Small-scale simulations

(b) Large-scale simulations (scaled actions)

(c) Large-scale simulations (scaled agents)

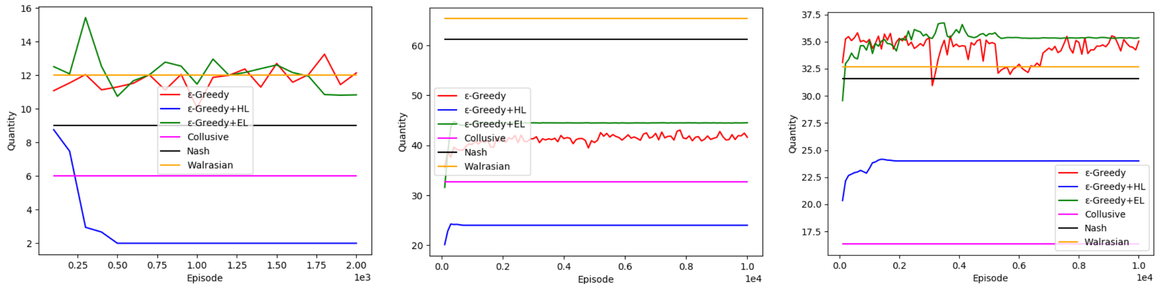
Figure 4.6: Comparison of variances in actions/production quantities chosen by different firms in an environment.

using different learning algorithms with each other, as well as with three equilibria. Figures 4.7a, 4.7b, and 4.7c show the results in terms of production quantities for models with 3, 15, and 30 total number of agents respectively. Similarly, figures 4.8a, 4.8b, and 4.8c show the results in terms of profits for models with 3, 15, and 30 total number of agents respectively. We take the average of the outputs produced by the agents using the same algorithm. In the model with 3 agents, agents can choose production quantity between $0 - 40$ as an action; here, $c = 4$, $u = 40$, and $v = 1$. Additionally, in model with 15 and 30 agents, agents can choose production quantity between $0 - 50$ as an action; here, $c = 5$, $u = 200$, and $v = 1$.

From figure 4.7, we can see that the quantities produced by using ϵ-greedy+HL are always much lower in comparison to those produced by the other two algorithms. Quantities produced by using ϵ-greedy+EL algorithms are almost always higher, although there is a small difference in quantities when compared with those produced by using basic ϵ-greedy. This difference in quantities also reflects in the profits presented in figure 4.8. However, the profit graphs do not follow the exact same pattern seen in the graphs representing quantities. We think that this is because the profit depends not only on agents' own production quantities but also on the production of other agents.

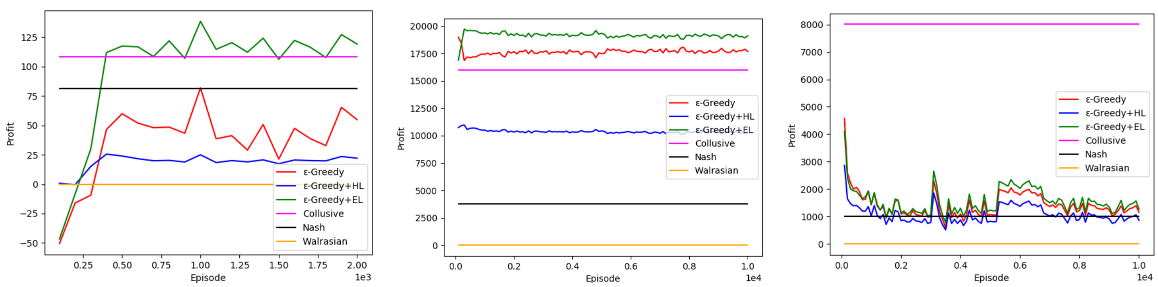
We also investigate models having agents equipped with any two of the mentioned algorithms. We use models with 10 agents where we separate them into two groups. Agents can choose any production quantity between $0 - 50$ as an action. Here, $c = 5$, $u = 200$, and $v = 1$. In figure 4.9, we compare the performances of two groups of agents where one group is using basic ϵ-greedy for learning, and the other group is using ϵ-greedy+HL. We can see that the two groups obtain almost similar profits with ϵ-greedy performing slightly better, even though there is some difference in production quantities. Unlike in figure 4.7, agents using ϵ-greedy+HL do not produce quantities that are too low. In figure 4.10, we compare the performances of two groups of agents where one group is using basic ϵ-greedy for learning, and the other group is using ϵ-greedy+EL. Similarly, In figure 4.11, we compare the performances of two groups of agents where one group is using ϵ-greedy+HL for learning, and the other group is using ϵ-greedy+EL. From both the figures, we can see that ϵ-greedy+EL outperforms the other two algorithms in terms of profit (figures 4.10b and 4.11b).

We can see from figures 4.7, 4.9a, 4.10a, and 4.11a that the firms using ϵ-greedy+EL and ϵ-greedy algorithms sometimes produce quantities even more than what they would have produced when in Walrasian equilibrium. It is mainly because agents using ϵ-greedy+HL algorithm produce much lower quantity, even lower than what they would have produced when in the cartel, i.e., when in collusive equilibrium. This gives leverage to other agents who are not following ϵ-greedy+HL algorithms. Consequently, we can see from figures 4.8, 4.9b, 4.10b, and 4.11b that the firms using ϵ-greedy+EL and ϵ-greedy algorithms sometimes obtain profits even more than what they would have obtained by being in a cartel. Despite this, the agents that are using ϵ-greedy+HL never converge to a profit lower than what they would have gotten in Walrasian equilibrium. Mostly, their profits are closer to Nash equilibrium, nonetheless, still lower than the agents following



(a) (b) (c)

Figure 4.7: Comparison of average quantities obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. (a) Cournot model has $n = 3$, $K = 40$, $c = 4$, $u = 40$, and $v = 1$. (b) Cournot model has $n = 15$, $K = 50$, $c = 5$, $u = 300$, and $v = 1$. (c) Cournot model has $n = 30$, $K = 50$, $c = 5$, $u = 300$, and $v = 1$.



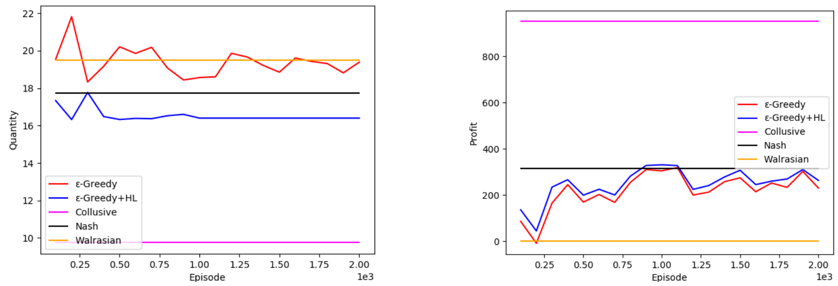
(a) (b) (c)

Figure 4.8: Comparison of average profits (corresponding to the quantities shown in the graphs above) obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. (a) Cournot model has $n = 3$, $K = 40$, $c = 4$, $u = 40$, and $v = 1$. (b) Cournot model has $n = 15$, $K = 50$, $c = 5$, $u = 300$, and $v = 1$. (c) Cournot model has $n = 30$, $K = 50$, $c = 5$, $u = 300$, and $v = 1$.

other algorithms. This behavior highlights the disadvantages of using ϵ-greedy+HL in an environment where not all the firms are following the same learning strategy. ϵ-Greedy+HL tends to eliminate good actions prematurely because of the partitioning in the action space. We have also discussed before that the performance of ϵ-greedy+HL is sensitive to the number of partitions we make in action set at each phase in learning.

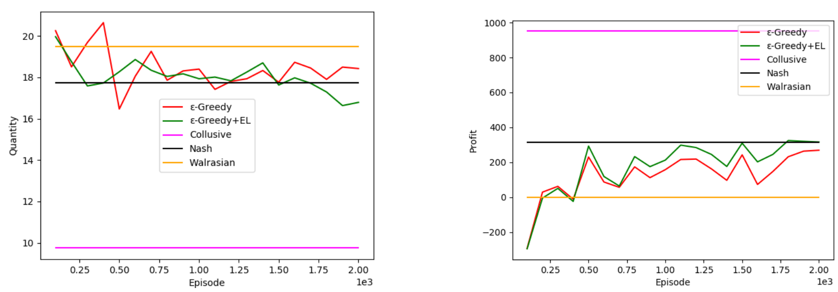
Models with Non-stationary Demand

In our experiments, adaptive ϵ-greedy consistently performed significantly worse than our approach. For ease of viewing, we present only the graphs of experiments on the duopoly model in figures 4.12 and 4.13. We do not include the comparison with adaptive ϵ-greedy in other graphs.



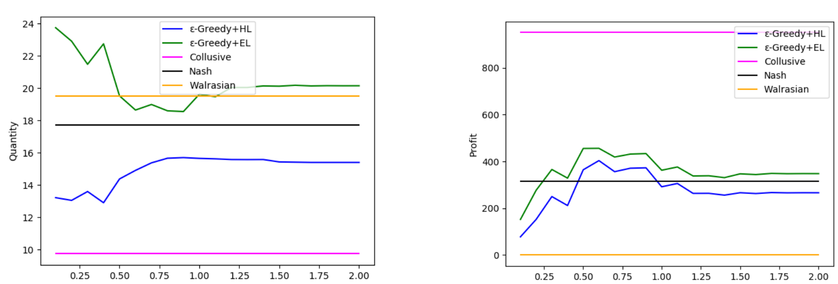
(a) (b)

Figure 4.9: Comparison of average quantities and corresponding average profits obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. Cournot model has firms with $n = 10$, $K = 50$, $c = 5$, $u = 200$, and $v = 1$.



(a) (b)

Figure 4.10: Comparison of average quantities and corresponding average profits obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. Cournot model has firms with $n = 10$, $K = 40$, $c = 5$, $u = 200$, and $v = 1$.



(a) (b)

Figure 4.11: Comparison of average quantities and corresponding average profits obtained by agents using assorted algorithms in the Cournot models, along with collusive, Nash, and Walrasian equilibrium. Cournot model has firms with $n = 10$, $K = 40$, $c = 5$, $u = 200$, and $v = 1$.

Fig. 4.12 shows the comparison of joint quantities; it shows collective changes in the action course of agents. With adaptive ϵ-greedy, agents are able to somewhat change the action course according to changes in demand, but the degrees/pattern of those changes does not match the demand pattern properly as compared to the nearly perfect alignment of outputs obtained by using our approach with the demand pattern. Moreover, with adaptive ϵ-greedy, the overall outputs are closer to Walrasian equilibrium, and in some cases more than Walrasian equilibrium. This also reflects in fig. 4.13, where the joint profits for adaptive ϵ-greedy go to negative values for some parts of the simulations. With our approach, the outcomes are either collusive or either in Nash equilibrium, with some exceptions where those are between Nash and Walrasian equilibrium. By the notion of regret discussed above, we can see that our approach causes less regret than adaptive ϵ-greedy as the results obtained by it are farther from collusive equilibrium than the results obtained by our approach. We have also observed that the outputs produced by individual but identical firms in a symmetric Cournot model are mostly similar when the firms use our approach. It means that the results are fair to each agent, and individual agents can rely on our method to get a fair share of the market profit despite being autonomous and independent.

For all the simulations of the models with non-stationary demand, we can see that the sudden changes in demand sometimes cause a sharp drop in joint outputs, however, quick recovery towards matching new demand can also be seen. The scale of simulation seems to affect the degree of sharp drops and recovery from those. This is especially true for results with demand patterns 1 and 3 which consist of sudden shocks in demand. Interestingly, the type of non-stationarity does not seem to affect the resulting equilibrium in outcomes obtained by our approach.

Scaling the Number of Firms

Cournot models with a large number of firms have rarely been explored in the literature. We simulate models consisting of 20, 40, 60, 80, and 100 total number of firms. Here, we consider the demand function with $v = 1000$ and $w = 1$, along with $c = 20$. The action choices range from 0 to 50. The results for joint quantity, joint profit, and joint cumulative regret are shown in figures 4.4a, 4.4b, and 4.4c, respectively. With the increasing number of firms, Nash equilibrium approaches Walrasian equilibrium.

Models with Stationary Demand

In figure 4.4a, for ϵ-greedy, we can see that the outcome is collusive for models with 20 firms; with 40 firms, it coincides with the Nash equilibrium; for models with 60 and 80 firms, it seems to coincide with the Walras equilibrium and interestingly, outcomes seems to be slightly more than the Walras equilibrium for the model with 100 firms. Therefore, we can say that ϵ-greedy performs very poorly for models with a large number of firms and begins to break down when the total number of firms is greater than 80. Surprisingly, if we compare its performance in terms of joint profits, it is better in the sense that it does not fall below the Walras equilibrium for models with 100 firms. This is unlike the results for previous sections where joint profits tend to be worse than their counterparts in joint quantities (if we assume that collusive $\succ$ Nash $\succ$ Walras).

The ϵ-greedy+HL and ϵ-greedy+EL algorithms both obtain collusive results in these simulations. For models with a comparatively smaller number of firms (20 and 40), ϵ-greedy+HL seems to be more collusive; however, the outcomes of both approaches are similar for larger models. Unlike previous results, ϵ-greedy+HL and ϵ-greedy+EL methods obtain nearly the same regrets. Not surprisingly, we found that both ϵ-greedy+HL and ϵ-greedy+EL run for almost equal time steps (shown in 4.5c). Overall, the graph for total time steps is very similar to figure 4.4c showing joint cumulative regrets. Interestingly, even though the model with 20 firms converges to collusive outcomes, the regret obtained by ϵ-greedy, in this case, is higher than other larger models. As the demand and size of action space stay the same across all models with the varying number of firms, agents in smaller models have more viable action choices that give positive payoffs/rewards. It causes a delay in converging to the final outcome and hence more exploration, which in turn causes more regret. From both types of large-scale simulations, we can see that having more viable action choices leads to more regret.

Models with Non-stationary Demand

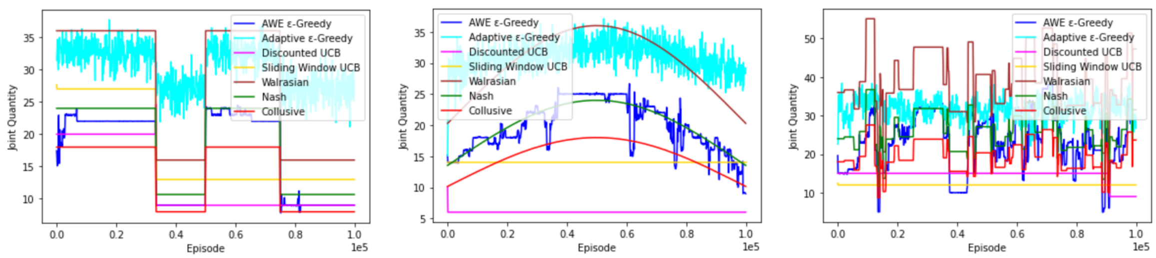
We show results for models with 10 firms in fig. 4.14, with 50 firms in fig. 4.15, and with 100 firms in fig. 4.16. With more firms in the system, there is less degree of collusion among them, and outcomes are either Nash or much closer to Nash, except for the demand pattern 2. From fig. 4.15a, we can see that, for the same demand, our algorithm might perform a bit differently in two separate time spans. Overall, these results suggest that with AWE ϵ-greedy, firms can show collusive behavior even when there are large number of firms present in the market. This is in contrast with popular belief in the literature that with bigger markets, industry outputs tend to move towards Walrasian equilibrium.

Scaling the Action Space

Firms with high production capacities can exist in the market. With high production capacity, more actions, i.e., production level choices, are available to the firms. Here, we use the Cournot duopoly model consisting of 2 symmetric firms. However, the size of the action space, S, varies from 50 to 500. Available actions in the specific model are in the range of $0 - S$. For the meaningful exploration of action space, the market demand also varies according to the size of the action space, such that the value of parameter v in the demand function is equal to the size of action space, S.

Models with Stationary Demand

From figure 4.3a, we can see that most of the outcomes converge to the Nash equilibrium. However, for ϵ-greedy+HL, outcomes are somewhat collusive, and



(a) (b) (c)

Figure 4.12: For Cournot duopoly model with 2 firms: comparison of joint quantity obtained by AWE ϵ-greedy and adaptive ϵ-greedy algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 2$, $K = 40$, $c = 4$, $u_s = 40$, and $v = 1$. Fig. 4.13a, 4.13b, and 4.13c show results for demand patterns 1, 2, and 3, resp.

for ϵ-greedy, those are a bit off from Nash equilibrium and lie between the Nash and Walras equilibria.

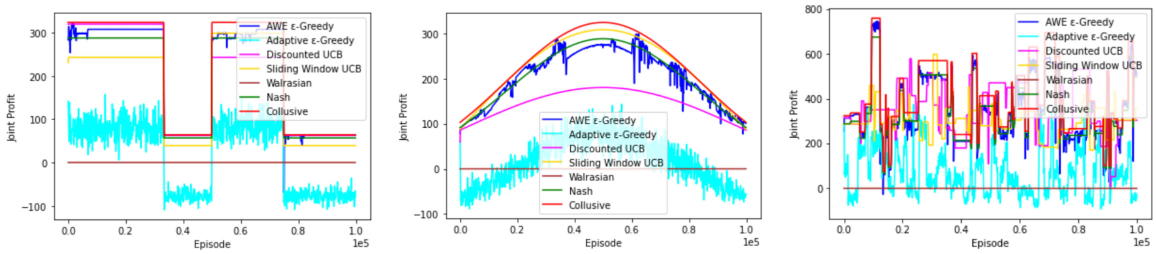
For the reasons discussed in the previous section, joint profits shown in figure 4.3b are slightly different from their counterpart joint quantities in figure 4.3a. As shown in figure 4.3c, the regret with ϵ-greedy is larger than that for ϵ-greedy+EL, which is, in turn, larger than the regret for ϵ-greedy+HL. In figure 4.3c, joint cumulative regret increases with the size of action space for all three approaches, but this pattern is not seen with the total steps required to converge (from 4.5b); for all approaches, total steps required to converge do not differ much with varying action space size; nonetheless, they are directly proportional to joint regrets.

Models with Non-stationary Demand

For the duopoly model, the degree of collusion is not affected by the increase in action space; however, for the model with 10 firms, outcomes are between Nash and Walrasian equilibrium. This shows that agents are not able to collude in a system with a large number of agents and big action space. For the duopoly model, although the joint profit drops sharply towards the Walrasian outcome when there are sudden changes in the demand, it quickly converges back to collusive or Nash outcomes. These drops are bigger than the ones observed in most of the other simulations; we think that increased exploration due to the large action space might be the reason behind it.

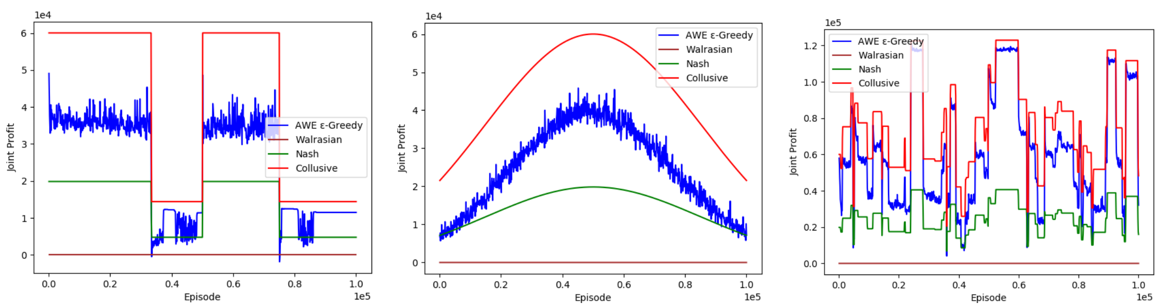
Asymmetric Firms

We also study Cournot models with asymmetric firms, i.e., firms with different marginal costs. In real-world settings, the production costs often vary among different sellers.



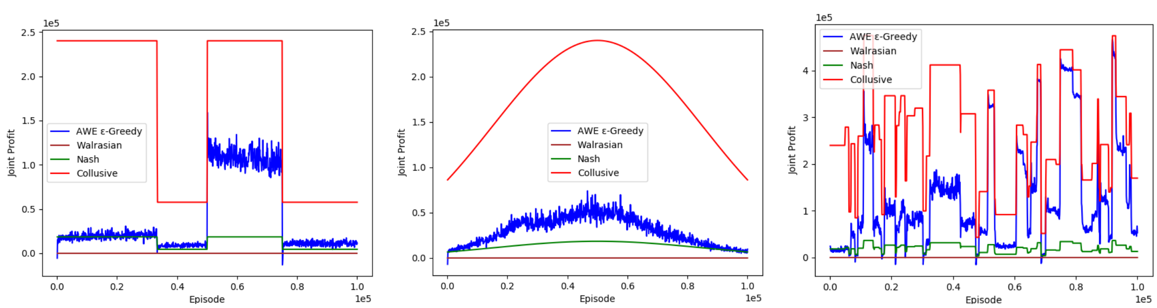
(a) (b) (c)

Figure 4.13: For Cournot duopoly model with 2 firms: comparison of joint profit obtained by AWE ϵ-greedy and adaptive ϵ-greedy algorithm, along with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 2$, $K = 40$, $c = 4$, $u_s = 40$, and $v = 1$. Fig. 4.13a, 4.13b, and 4.13c show results for demand patterns 1, 2, and 3, resp.



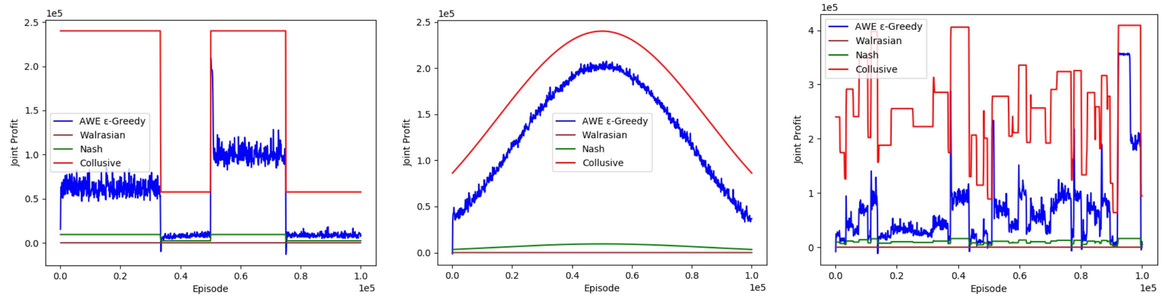
(a) (b) (c)

Figure 4.14: For Cournot model with 10 firms: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 10$, $K = 50$, $c = 10$, $u_s = 500$, and $v = 1$. Fig. 4.14a, 4.14b, and 4.14c show results for demand patterns 1, 2, and 3, resp.



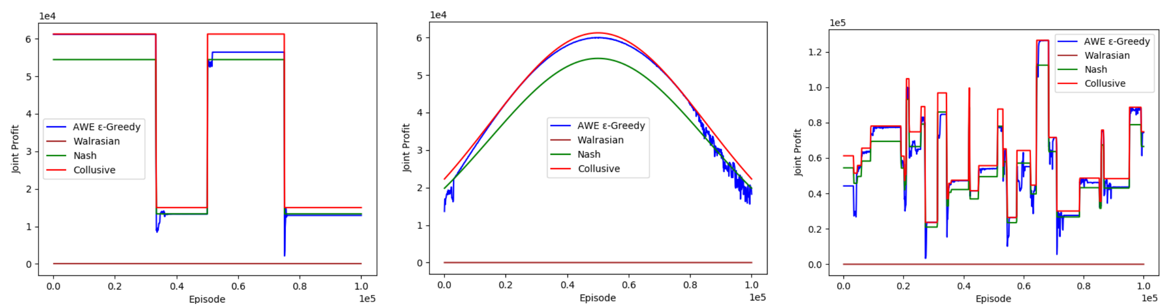
(a) (b) (c)

Figure 4.15: For Cournot model with 50 firms: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 50$, $K = 50$, $c = 20$, $u_s = 1000$, and $v = 1$. Fig. 4.15a, 4.15b, and 4.15c show results for demand patterns 1, 2, and 3, resp.



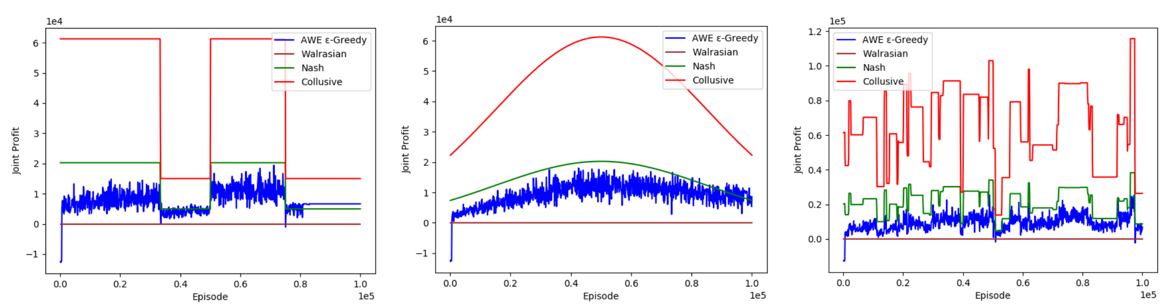
(a) (b) (c)

Figure 4.16: For Cournot model with 100 firms: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 100$, $K = 50$, $c = 20$, $u_s = 1000$, and $v = 1$. Fig. 4.16a, 4.16b, and 4.16c show results for demand patterns 1, 2, and 3, resp.



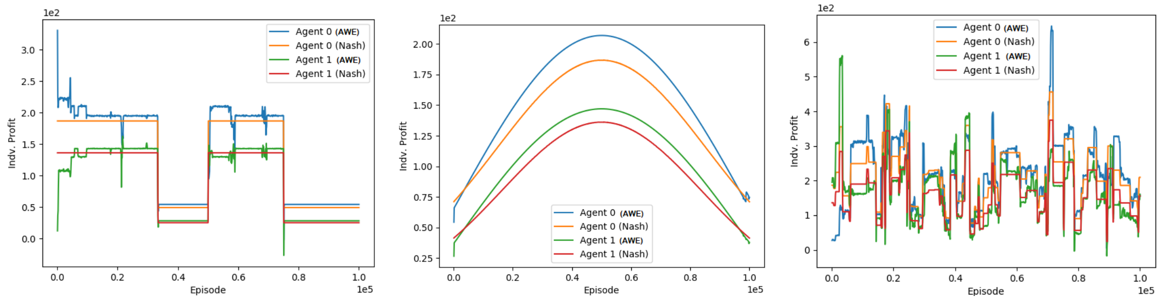
(a) (b) (c)

Figure 4.17: For Cournot model with scaled actions: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Corunot model has symmetric firms with $n = 2$, $K = 500$, $c = 4$, $u_s = 500$, and $v = 1$. Fig. 4.17a, 4.17b, and 4.17c show results for demand patterns 1, 2, and 3, resp.



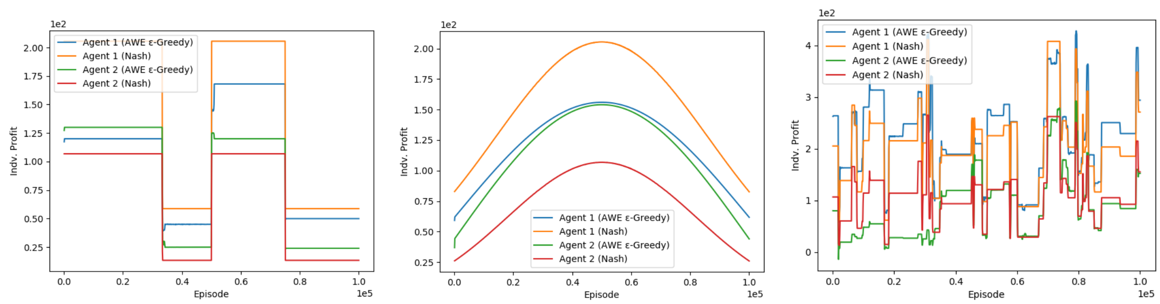
(a) (b) (c)

Figure 4.18: For Cournot model with scaled actions: comparison of joint profit obtained by AWE ϵ-greedy with collusive, Nash and Walrasian equilibrium. Cournot model has symmetric firms with $n = 10$, $K = 500$, $c = 4$, $u_s = 500$, and $v = 1$. Fig. 4.18a, 4.18b, and 4.18c show results for demand patterns 1, 2, and 3, resp.



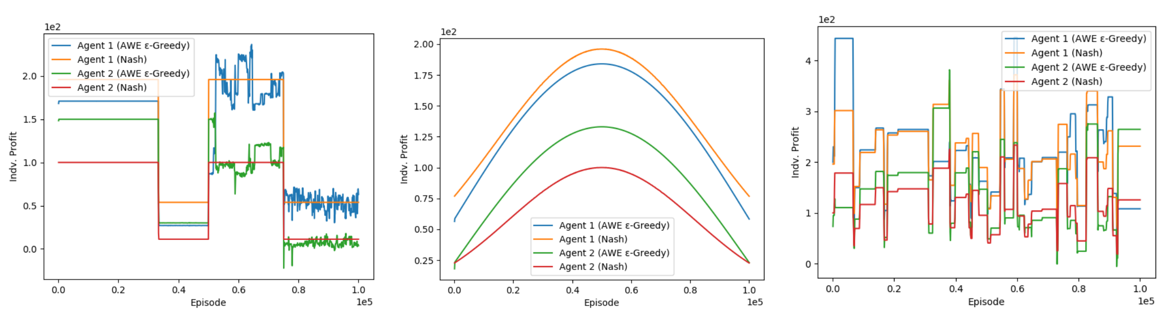
(a) (b) (c)

Figure 4.19: For Cournot duopoly model with 2 asymmetric firms: comparison of quantity obtained by an individual agent using AWE ϵ-greedy algorithm with respective Nash outputs. Corunot model has asymmetric firms with $n = 2$, $K = 40$, $c_0 = 1$, $c_1 = 3$, $u_s = 40$, and $v = 1$. Fig. 4.19a, 4.19b, and 4.19c show results for demand patterns 1, 2, and 3, resp. (AWE) in graphs refer to AWE ϵ-Greedy.



(a) (b) (c)

Figure 4.20: For Cournot duopoly model with 2 asymmetric firms: comparison of quantity obtained by an individual agent using AWE ϵ-greedy algorithm with respective Nash outputs. Corunot model has asymmetric firms with $n = 2$, $K = 40$, $c_0 = 1$, $c_1 = 5$, $u_s = 40$, and $v = 1$. Fig. 4.20a, 4.20b, and 4.20c show results for demand patterns 1, 2, and 3, resp.



(a) (b) (c)

Figure 4.21: For Cournot duopoly model with 2 asymmetric firms: comparison of quantity obtained by an individual agent using AWE ϵ-greedy algorithm with respective Nash outputs. Corunot model has asymmetric firms with $n = 2$, $K = 40$, $c_0 = 2$, $c_1 = 6$, $u_s = 40$, and $v = 1$. Fig. 4.21a, 4.21b, and 4.21c show results for demand patterns 1, 2, and 3, resp.

$(C1, C2)$		Nash Equilibrium	ϵ-greedy	ϵ-greedy + HL	ϵ-greedy + EL
(1,2)	Q	[13.3, 12.3]	[13.8±4.3, 12.4±3.9]	[11.3±2.9, 10.8±2.7]	[11.4±3.9, 12.7±4.7]
	P	[177.8, 152.2]	[170.6±49.0, 143.2±49.3]	[181.6±35.0, 164.2±27.7]	[163.9±38.2, 164.5±30.3]
(1,3)	Q	[13.7, 11.7]	[13.7±5.2, 12.3±5.5]	[11.8±3.7, 10.0±3.0]	[12.8±3.1, 10.5±4.1]
	P	[186.7, 136.1]	[166.8±69.5, 122.3±66.5]	[196.9±45.3, 142.4±43.0]	[189.2±29.4, 138.3±36.0]
(1,4)	Q	[14.0, 11.0]	[13.5±4.6, 12.1±5.2]	[11.9±3.2, 9.7±2.4]	[12.8±3.6, 10.1±3.2]
	P	[196.0, 121.0]	[177.2±62.1, 117.6±58.8]	[199.6±39.1, 129.6±43.5]	[196.7±23.1, 125.3±31.8]
(1,5)	Q	[14.3, 10.3]	[13.5±4.6, 11.3±4.8]	[12.6±2.4, 8.4±3.1]	[12.4±2.7, 10.0±2.1]
	P	[205.5, 106.8]	[180.9±59.6, 107.1±56.4]	[216.0±47.5, 109.5±34.4]	[194.6±24.5, 122.9±31.9]
(2,3)	Q	[13.0, 12.0]	[13.1±5.1, 13.2±5.7]	[11.6±3.0, 9.6±2.2]	[12.8±2.7, 11.1±3.3]
	P	[169.0, 144.0]	[148.6±67.1, 133.2±67.5]	[186.9±43.8, 142.8±36.2]	[173.9±26.4, 141.3±35.0]
(2,4)	Q	[13.3, 11.3]	[13.6±5.0, 11.2±4.9]	[11.9±3.3, 10.0±3.6]	[12.3±3.4, 10.7±2.1]
	P	[177.8, 128.5]	[167.5±67.2, 115.8±60.1]	[185.9±44.3, 131.8±31.7]	[175.9±36.0, 133.6±33.0]
(2,5)	Q	[13.7, 10.7]	[13.3±4.7, 12.1±4.7]	[12.7±3.9, 7.9±3.2]	[12.6±2.6, 9.6±2.7]
	P	[186.7, 113.7]	[160.1±62.2, 106.5±64.1]	[211.8±49.8, 103.0±32.4]	[186.1±30.4, 118.2±27.0]
(3,4)	Q	[12.7, 11.7]	[12.9±4.5, 11.9±5.1]	[11.6±2.3, 8.9±2.1]	[11.4±2.4, 11.4±2.3]
	P	[160.4, 136.1]	[149.6±62.4, 121.3±64.0]	[179.9±29.2, 130.1±27.2]	[152.6±23.9, 142.8±31.9]
(3,5)	Q	[13.0, 11.0]	[13.3±5.4, 11.3±3.5]	[11.5±2.9, 9.0±3.0]	[12.2±2.2, 10.2±2.8]
	P	[169.0, 121.0]	[149.8±58.4, 115.6±54.7]	[180.8±23.0, 121.1±21.3]	[168.5±27.9, 120.5±23.6]

Table 4.1: Results of computer simulations for asymmetric Cournot duopoly model with firms having different marginal costs $C1$ and $C2$; $u = 1000$, $v = 1$. Q=Quantity; P=Profit. Results are in format $[q_1, q_2]$ for quantities and $[p_1, p_2]$ for profits; q_i and p_i represents quantity and profit for firm i, respectively.

Models with Stationary Demand

Here, we use Cournot duopoly models with two firms having different marginal costs. These costs vary in different simulations. We used the same model parameters as those used in small-scale simulations. We have $u = 40$ and $v = 1$ as constants in an inverse demand function (eq. 4.1), but the constant marginal cost c is different for the two firms. Let $c1$ and $c2$ be the marginal costs for two firms, then $c1$ lies in the range $1 - 3$ and $c2$ in the range $3 - 5$. Overall, the results (shown in 4.1) are similar to those with symmetric firms. ϵ-greedy mostly converges to outcomes worse than Nash equilibrium. However, for ϵ-greedy+HL and ϵ-greedy+EL, most of the outcomes are collusive. We see closer-to-collusive policies with ϵ-greedy+HL than with ϵ-greedy+EL.

Models with Non-stationary Demand

Again, we use Cournot duopoly models with two firms having different marginal costs. We show the results for the Cournot duopoly model where the constant marginal cost of one firm is 1, while that of another is 3. We show a comparison with Nash outcomes in fig. 4.19. We can see that both firms perform better than their respective Nash outcomes, i.e., showing collusive behavior. However, during further investigation, we found that as the difference between marginal costs increases or as the number of agents in the market increases, firms tend to produce highly diverse outcomes in terms of comparison with their respective Nash quantities. Also, not all firms are able to produce collusive or Nash outcomes.

4.5 Summary

We have investigated the modeling of Cournot games in the MAB setting, especially when the firms have no knowledge of the market. Using the MAB setting allows for the implementation of approaches based on the exploration-exploration paradigm. To deal with ordered action sets, we proposed two novel approaches, which are extensions of the ϵ-greedy algorithm. Given the assumption of static demand, the proposed methods optimize the exploration by reducing the action space. ϵ-greedy+HL performs best in terms of joint cumulative regret but comes with the additional responsibility of deciding the number of partitions in action space. With ϵ-greedy+EL, there is no such burden, but it may cause more regret than ϵ-greedy+HL. Our proposed approaches converge much faster than the baseline ϵ-greedy method. ϵ-greedy rarely produces collusive outcomes but mostly obtains Nash quantity or any state between Nash and Walrasian outcome. Our proposed approaches mostly obtain somewhat collusive outcomes for all kinds of simulations, although full collusion usually does not emerge.

We also modeled the decision-making in repeated Cournot games with *non-stationary demand* as a non-stationary bandit problem. We used three different non-stationary demand patterns to represent various kinds of non-stationarities found in real-world settings. We proposed AWE ϵ-greedy algorithm, which incorporates mechanisms for quantifying changes in rewards, to help it adapt to the varying demand. Outcomes obtained by our proposed approach show that it can help firms to readily and smoothly change their action course according to the changing demand. We think that our results are supportive of the concern over online algorithms being collusive without any external intervention Klein [2020].

Chapter 5 Conclusion

In the previous chapters, we have described our work in detail. In Chapter 2, we have included the work done on decentralized hedonic coalition formation games Taywade et al. [2018]. Here, we modeled the problem as a grid-world exploration problem and proposed a novel decentralized learning approach. We also introduced a novel coalition discovery technique called budding, in which large coalitions spawn smaller sub-coalitions if they would increase the total utility of agents in the new sub-coalition. Our experiments showed promising results as our techniques, both with and without budding, performed well on a variety of hedonic games and showed huge improvement over random partitioning.

In Chapter 3, which is based on three different matching problems, we have worked on optimal bipartite matching problem Taywade et al. [2020], roommate matching Taywade et al. [2018], and stable bipartite matching problem. In Taywade et al. [2018], we have worked on the roommate matching problem and also obtained results close to optimal matching as we implemented a modified version of the decentralized learning algorithm. In Taywade et al. [2020], we have considered the decentralized marriage problem, modeled spatially as a grid world, and non-spatially as an affiliation network and a small-world network. We implemented the novel decentralized learning approach (different from the one in Taywade et al. [2018]) and obtained results very close to optimal matching, for every type of the model. In Taywade et al. [2021], we have shown that the multi-agent reinforcement learning paradigm can be successfully used for decentralized stable matching problems that are formulated spatially in a dynamic and uncertain environment, with independent and autonomous agents having minimum initial knowledge. Our approach is also applicable for variations such as stable matchings with incomplete lists and ties. Agents tend to be happy with their final matches, as outcomes are stable or close-to-stable and fair for everyone. Even with unstable outcomes, agents are less likely to exploit instability.

In Chapter 4, we modeled the decision-making in repeated Cournot games with stationary as well as non-stationary demand. For Cournot games with stationary demands, we modeled the problem using the stochastic multi-armed bandit framework and proposed two extensions of ϵ-greedy algorithm, i.e., ϵ-greedy+HL and ϵ-greedy+EL Taywade et al. [2022]. For Cournot games with non-stationary demands, we modeled the problem using the non-stationary multi-armed bandit framework and proposed Adaptive with Weighted Exploration (AWE) ϵ-greedy algorithm. This algorithm is able to assist agents in dealing with non-stationary market demand. The collusive nature of most of the outcomes in our simulations is supportive of the concern over online algorithms being collusive without any external intervention Klein [2020].

Bibliography

Sherief Abdallah and Victor Lesser. Organization-based cooperative coalition formation. In *Intelligent Agent Technology, 2004.(IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, pages 162–168. IEEE, 2004.

Jasmina Arifovic and Michael K Maschek. Revisiting individual evolutionary learning in the cobweb model–an illustration of the virtual spite-effect. *Computational Economics*, 28(4):333–354, 2006.

Haris Aziz, Felix Brandt, and Hans Georg Seedig. Optimal partitions in additively separable hedonic games. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 43, 2011a.

Haris Aziz, Felix Brandt, and Hans Georg Seedig. Stable partitions in additively separable hedonic games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 183–190. International Foundation for Autonomous Agents and Multiagent Systems, 2011b.

Haris Aziz, Felix Brandt, and Hans Georg Seedig. Computing desirable partitions in additively separable hedonic games. *Artificial Intelligence*, 195:316–334, 2013.

Yoram Bachrach, Richard Everett, Edward Hughes, Angeliki Lazaridou, Joel Z Leibo, Marc Lanctot, Michael Johanson, Wojciech M Czarnecki, and Thore Graepel. Negotiating team formation using deep reinforcement learning. *Artificial Intelligence*, 288:103356, 2020.

Suryapratim Banerjee, Hideo Konishi, and Tayfun Sönmez. Core in a simple coalition formation game. *Social Choice and Welfare*, 18(1):135–153, 2001.

William L Berry. Lot sizing procedures for requirements planning systems: A framework for analysis. *Production and Inventory Managements*, 13(2):19–34, 1972.

Lilian Besson and Emilie Kaufmann. What doubling tricks can and can't do for multi-armed bandits. *arXiv preprint arXiv:1803.06971*, 2018.

Kostas Bimpikis, Shayan Ehsani, and Rahmi Ilkılıç. Cournot competition in networked markets. *Management Science*, 65(6):2467–2481, 2019.

Gian Italo Bischi, Fabio Lamantia, and Davide Radi. An evolutionary Cournot model with limited market knowledge. *Journal of Economic Behavior & Organization*, 116:219–238, 2015.

Anna Bogomolnaia and Matthew O Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.

Béla Bollobás. Random graphs. 2001. *Cambridge Stud. Adv. Math*, 2001.

James W Boudreau. A note on the efficiency and fairness of decentralized matching. *Operations Research Letters*, 39(4):231–233, 2011.

Djallel Bouneffouf and Raphael Féraud. Multi-armed bandit problem with known trend. *Neurocomputing*, 205:16–21, 2016.

Djallel Bouneffouf, Romain Laroche, Tanguy Urvoy, Raphael Féraud, and Robin Allesiardo. Contextual bandit for active learning: Active thompson sampling. In *International Conference on Neural Information Processing*, pages 405–412. Springer, 2014.

Lucian Bu, Robert Babu, Bart De Schutter, et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Yang Cao, Zheng Wen, Branislav Kveton, and Yao Xie. Nearly optimal adaptive procedure with change detection for piecewise-stationary bandit. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 418–427. PMLR, 2019.

Emanuele Cavenaghi, Gabriele Sottocornola, Fabio Stella, and Markus Zanker. Non stationary multi-armed bandit: Empirical evaluation of a new concept drift-aware algorithm. *Entropy*, 23(3):380, 2021.

Nicolo Cesa-Bianchi and Paul Fischer. Finite-time regret bounds for the multiarmed bandit problem. In *ICML*, volume 98, pages 100–108. Citeseer, 1998.

Georgios Chalkiadakis and Craig Boutilier. Bayesian reinforcement learning for coalition formation under uncertainty. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '04, pages 1090–1097, Washington, DC, USA, 2004. IEEE Computer Society.

Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Learning to optimize under non-stationarity. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1079–1087. PMLR, 2019.

Michael Chletsos and Anna Saiti. Hospitals as suppliers of healthcare services. In *Strategic Management and Economics in Health Care*, pages 179–205. Springer, 2019.

Margherita Comola and Marcel Fafchamps. An experimental study on decentralized networked markets. *Journal of Economic Behavior & Organization*, 145:567–591, 2018.

Antoine Augustin Cournot. *Recherches sur les Principes Mathématiques de la Théorie des Richesses*. L. Hachette, 1838.

Wesley Cowan, Junya Honda, and Michael N Katehakis. Normal bandits of unknown means and variances. *J. Mach. Learn. Res.*, 18:154–1, 2017.

Herbert Dawid. *Adaptive learning by genetic algorithms: Analytical results and applications to economic models*. Springer Science & Business Media, 2011.

Arnoud V Den Boer. Dynamic pricing and learning: historical origins, current research, and new directions. *Surveys in Operations Research and Management Science*, 20(1):1–18, 2015.

Effrosyni Diamantoudi, Eiichi Miyagawa, and Licun Xue. Random paths to stability in the roommate problem. *Games and Economic Behavior*, 48(1):18–28, 2004.

Effrosyni Diamantoudi, Eiichi Miyagawa, and Licun Xue. Decentralized matching: The role of commitment. *Games and Economic Behavior*, 92:1–17, 2015.

Peter Sheridan Dodds, Roby Muhamad, and Duncan J Watts. An experimental study of search in global social networks. *science*, 301(5634):827–829, 2003.

Alexandre dos Santos Mignon and Ricardo Luis de Azevedo da Rocha. An adaptive implementation of ε-greedy in reinforcement learning. *Procedia Computer Science*, 109:1146–1151, 2017.

Federico Echenique and Leeat Yariv. An experimental study of decentralized matching. Technical report, 2012.

P Erdos and A Renyi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

Kimmo Eriksson and Olle Häggström. Instability of matchings in decentralized markets with various preference structures. *International Journal of Game Theory*, 36(3-4):409–420, 2008.

Tamás Fleiner, Robert W Irving, and David F Manlove. Efficient algorithms for generalized stable marriage and roommates problems. *Theoretical computer science*, 381(1-3):162–176, 2007.

Leonard N Foner. Yenta: a multi-agent, referral-based matchmaking system. In *Proceedings of the first International Conference on Autonomous Agents*, pages 301–307. ACM, 1997.

Reiner Franke. Coevolution and stable adjustments in the cobweb model. *Journal of Evolutionary Economics*, 8(4):383–406, 1998.

Drew Fudenberg, Fudenberg Drew, David K Levine, and David K Levine. *The theory of learning in games*, volume 2. MIT press, 1998.

Tomer Gafni and Kobi Cohen. Learning in restless multi-armed bandits using adaptive arm sequencing rules. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1206–1210. IEEE, 2018.

Martin Gairing and Rahul Savani. Computing stable outcomes in hedonic games. In *International Symposium on Algorithmic Game Theory*, pages 174–185. Springer, 2010.

David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*, pages 174–188. Springer, 2011.

Gourab Ghatak. A change-detection based thompsonsampling framework for non-stationary bandits. *IEEE Transactions on Computers*, 2020.

Gourab Ghatak, Hardhik Mohanty, and Aniq Ur Rahman. Kolmogorovsmirnov test-based actively-adaptive thompson sampling for non-stationary bandits. *IEEE Transactions on Artificial Intelligence*, 2021.

Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4): 1141–1144, 1959.

John C Gittins and David M Jones. A dynamic allocation index for the discounted multiarmed bandit problem. *Biometrika*, 66(3):561–565, 1979.

Stephen C Graves and Sean P Willems. Strategic inventory placement in supply chains: Nonstationary demand. *Manufacturing & service operations management*, 10(2):278–287, 2008.

Dan Gusfield. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*, 16(1):111–128, 1987.

Dan Gusfield and Robert W Irving. *The stable marriage problem: structure and algorithms*. MIT press, 1989.

Guillaume Haeringer and Myrna Wooders. Decentralized job matching. *International Journal of Game Theory*, 40(1):1–28, 2011.

Karsten Hansen, Kanishka Misra, and Mallesh Pai. Algorithmic collusion: Supra-competitive prices via independent algorithms. *CEPR Discussion Paper No. DP14372*, 2020.

Jaap-Henk Hoepman. Simple distributed weighted matchings. *arXiv preprint cs/0410047*, 2004.

Junya Honda and Akimichi Takemura. Optimality of thompson sampling for gaussian bandits depends on priors. In *Artificial Intelligence and Statistics*, pages 375–383. PMLR, 2014.

Steffen Huck, Hans-Theo Normann, and Jörg Oechssler. Through trial and error to collusion. *International Economic Review*, 45(1):205–224, 2004.

Robert W Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6(4):577–595, 1985.

Robert W Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48 (3):261–272, 1994.

Robert W Irving and David F Manlove. The stable roommates problem with ties. *Journal of Algorithms*, 43(1):85–105, 2002.

Robert W Irving, Paul Leather, and Dan Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM (JACM)*, 34(3):532–543, 1987.

Kazuo Iwama and Shuichi Miyazaki. A survey of the stable marriage problem and its variants. In *International Conference on Informatics Education and Research for Knowledge-Circulating Society*, pages 131–136. IEEE Computer Society, January 2008.

Pavel Janovsky and Scott A DeLoach. Increasing coalition stability in large-scale coalition formation with self-interested agents. In *ECAI*, pages 1606–1607, 2016a.

Pavel Janovsky and Scott A DeLoach. Multi-agent simulation framework for large-scale coalition formation. In *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*, pages 343–350. IEEE, 2016b.

Jian-Guo Jiang, SU Zhao-Pin, QI Mei-Bin, and Guo-Fu ZHANG. Multi-task coalition parallel formation strategy based on reinforcement learning. *Acta Automatica Sinica*, 34(3):349–352, 2008.

Jeffrey O Kephart and Gerald J Tesauro. Pseudo-convergent Q-learning by competitive pricebots. In *Proc. 17th Int'l Conf. Machine Learning*. Citeseer, 2000.

Arif Khan, Alex Pothen, Md Mostofa Ali Patwary, Nadathur Rajagopalan Satish, Narayanan Sundaram, Fredrik Manne, Mahantesh Halappanavar, and Pradeep Dubey. Efficient approximation algorithms for weighted b-matching. *SIAM Journal on Scientific Computing*, 38(5):S593–S619, 2016.

Steven O Kimbrough and Ming Lu. A note on Q-learning in the Cournot game. In *Proceedings of the Second Workshop on e-Business*, 2003.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Daniel S Kirschen and Goran Strbac. *Fundamentals of Power System Economics*. John Wiley & Sons, 2018.

Timo Klein. (M)isunderstanding algorithmic collusion. *Competition Policy International (CPI)*, 2020.

Samuel S Komorita and David A Kravitz. Coalition formation: A social psychological approach. In *Basic group processes*, pages 179–203. Springer, 1983.

Ville Könönen. Dynamic pricing based on asymmetric multiagent reinforcement learning. *International Journal of Intelligent Systems*, 21(1):73–98, 2006.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Stephan Lauermann and Georg Nöldeke. Stable marriages and search frictions. *Journal of Economic Theory*, 151:163–195, 2014.

Nir Levine, Koby Crammer, and Shie Mannor. Rotting bandits. *arXiv preprint arXiv:1702.07274*, 2017.

Xin Li and Leen-Kiat Soh. Investigating reinforcement learning in multiagent coalition formation. In *Amer. Assoc. Artif. Intell. Workshop on Forming and Maintaining Coalitions and Teams in Adaptive Multiagent Systems Tech. Rep. WS-04-06*, pages 22–28, 2004.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

Fang Liu, Joohyun Lee, and Ness Shroff. A change-detection based framework for piecewise-stationary multi-armed bandit problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Haoyang Liu, Keqin Liu, and Qing Zhao. Learning in a changing world: Restless multiarmed bandit with unknown dynamics. *IEEE Transactions on Information Theory*, 59(3):1902–1916, 2012.

Michael Luck, Peter McBurney, and Chris Preist. *Agent technology: enabling next generation computing (a roadmap for agent based computing)*. AgentLink, 2003.

Tim Matthews, Sarvapali D Ramchurn, and Georgios Chalkiadakis. Competing with humans at fantasy football: Team formation in large partially-observable domains. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*. aaai.org, 2012.

Joseph Mellor and Jonathan Shapiro. Thompson sampling in switching environments with bayesian online change detection. In *Artificial Intelligence and Statistics*, pages 442–450. PMLR, 2013.

Rahul Meshram, D Manjunath, and Aditya Gopalan. On the whittle index for restless multiarmed hidden markov bandits. *IEEE Transactions on Automatic Control*, 63(9):3046–3053, 2018.

Tomasz Michalak, Jacek Sroka, Talal Rahwan, Michael Wooldridge, Peter McBurney, and Nicholas R Jennings. A distributed algorithm for anytime coalition structure generation. In *Proceedings of the 9th International Conference on Autonomous*

Agents and Multiagent Systems: volume 1-Volume 1, pages 1007–1014. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

Kanishka Misra, Eric M Schwartz, and Jacob Abernethy. Dynamic online pricing with incomplete information using multiarmed bandit experiments. *Marketing Science*, 38(2):226–252, 2019.

E Mohebbi and F Choobineh. The impact of component commonality in an assemble-to-order environment under supply and demand uncertainty. *Omega*, 33(6):472–482, 2005.

George L Nemhauser and Glenn M Weber. Optimal set partitioning, matchings and lagrangian duality. *Naval Research Logistics Quarterly*, 26(4):553–563, 1979.

Muriel Niederle and Alvin E Roth. Making markets thick: How norms governing exploding offers affect market performance. *preprint*, 2006.

Muriel Niederle and Leeat Yariv. Matching through decentralized markets. *Discussion Paper, Stanford University*, 2007.

Muriel Niederle and Leeat Yariv. Decentralized matching with aligned preferences. Technical report, National Bureau of Economic Research, 2009.

Mihaela Oprea. Applications of multi-agent systems. In *Information Technology*, pages 239–270. Springer, 2004.

Joana Pais, Agnes Pintér, and Robert F Veszteg. *Decentralized matching markets: a laboratory experiment*. ISEG-Departamento de Economia, 2012.

Joana Pais, Ágnes Pintér, and Róbert F Veszteg. Decentralized matching markets with (out) frictions: a laboratory experiment. *Experimental Economics*, pages 1–28, 2017.

Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Stability and optimality in matching problems with weighted preferences. In *Agents and Artificial Intelligence*, pages 319–333. Springer Berlin Heidelberg, 2013.

Robert Preis. Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs. In *STACS 99*, pages 259–269. Springer Berlin Heidelberg, 1999.

Davide Radi. Walrasian versus cournot behavior in an oligopoly of boundedly rational firms. *Journal of Evolutionary Economics*, 27(5):933–961, 2017.

Talal Rahwan and Nicholas R Jennings. An improved dynamic programming algorithm for coalition structure generation. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1417–1420. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

Talal Rahwan, Sarvapali D Ramchurn, Nicholas R Jennings, and Andrea Giovannucci. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, 34:521–567, 2009.

Vishnu Raj and Sheetal Kalyani. Taming non-stationary bandits: A bayesian approach. *arXiv preprint arXiv:1707.09727*, 2017.

Moses Richardson. On finite projective games. *Proceedings of the American Mathematical Society*, 7(3):458–465, 1956.

Thomas Riechmann. Cournot or Walras? long-run results in oligopoly games. *Journal of Institutional and Theoretical Economics (JITE)/Zeitschrift für die gesamte Staatswissenschaft*, pages 702–720, 2006.

Alvin E Roth. A natural experiment in the organization of entry-level labor markets: regional markets for new physicians and surgeons in the United Kingdom. *The American Economic Review*, pages 415–440, 1991.

Alvin E Roth and John H Vande Vate. Random paths to stability in two-sided matching. *Econometrica: Journal of the Econometric Society*, pages 1475–1480, 1990.

Alvin E Roth and Xiaolin Xing. Turnaround time and bottlenecks in market clearing: Decentralized matching in the market for clinical psychologists. *Journal of political Economy*, 105(2):284–329, 1997.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

Yoan Russac, Claire Vernade, and Olivier Cappé. Weighted linear bandits for non-stationary environments. *arXiv preprint arXiv:1909.09146*, 2019.

Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Anytime coalition structure generation with worst case guarantees. *arXiv preprint cs/9810005*, 1998.

Mark Satterthwaite and Artyom Shneyerov. Dynamic matching, two-sided incomplete information, and participation costs: Existence and convergence to perfect competition. *Econometrica*, 75(1):155–200, 2007.

Jacob Schlueter and Judy Goldsmith. Proximal stability, 2018. In progress.

Sandip Sen and Gerhard Weiss. Learning in multiagent systems. *Multiagent systems: A modern approach to distributed artificial intelligence*, pages 259–298, 1999.

Julien Seznec, Pierre Menard, Alessandro Lazaric, and Michal Valko. A single algorithm for both restless and rested rotting bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 3784–3794. PMLR, 2020.

Edward A Silver. Inventory management: an overview, canadian publications, practical applications and suggestions for future research. *INFOR: Information Systems and Operational Research*, 46(1):15–27, 2008.

Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

RS Sutton and AG Barto. *Reinforcement learning: An introduction.* MIT Press, Cambridge, MA, 1998.

Jimmy JM Tan. A maximum stable matching for the roommates problem. *BIT Numerical Mathematics*, 30(4):631–640, 1990.

Taywade, Goldsmith, and Harrison. Decentralized multi-agent approach for hedonic games. *EUMAS*, 2018.

Taywade, Goldsmith, and Harrison. Decentralized marriage models. *FLAIRS*, 2020.

Kshitija Taywade, Judy Goldsmith, and Brent Harrison. Multi-agent reinforcement learning for decentralized stable matching. In *Algorithmic Decision Theory: 7th International Conference, ADT 2021, Toulouse, France, November 3–5, 2021, Proceedings*, pages 375–389, 2021.

Kshitija Taywade, Brent Harrison, Adib Bagh, and Judy Goldsmith. Modelling cournot games as multi-agent multi-armed bandits. In *The International FLAIRS Conference Proceedings*, volume 35, 2022.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Jeffrey Travers and Stanley Milgram. The small world problem. *Phychology Today*, 1 (1):61–67, 1967.

Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. In *Social Networks*, pages 179–197. Elsevier, 1977.

Francesco Trovo, Stefano Paladino, Marcello Restelli, and Nicola Gatti. Multi-armed bandit for pricing. In *12th European Workshop on Reinforcement Learning*, pages 1–9, 2015.

Huseyin Tunc, Onur A Kilic, S Armagan Tarim, and Burak Eksioglu. The cost of using stationary inventory policies when demand is non-stationary. *Omega*, 39(4): 410–415, 2011.

M Utku Ünver. On the survival of some unstable two-sided matching mechanisms. *International Journal of Game Theory*, 33(2):239–254, 2005.

Thomas Vallée and Murat Yıldızoğlu. Convergence in the finite Cournot oligopoly with social and individual learning. *Journal of Economic Behavior & Organization*, 72(2):670–690, 2009.

Marcelo M Vanzin and KS Barber. Decentralized partner finding in multi-agent systems. In *Coordination of Large-Scale Multiagent Systems*, pages 75–98. Springer, 2006.

Fernando Vega-Redondo. The evolution of Walrasian behavior. *Econometrica: Journal of the Econometric Society*, pages 375–384, 1997.

Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European Conference on Machine Learning*, pages 437–448. Springer, 2005.

H H Viet, L H Trang, S Lee, and T Chung. A bidirectional local search for the stable marriage problem. In *2016 International Conference on Advanced Computing and Applications (ACOMP)*, pages 18–24. ieeexplore.ieee.org, November 2016.

Nicolaas J Vriend. An illustration of the essential difference between individual and social learning, and its consequences for computational analyses. *Journal of economic dynamics and control*, 24(1):1–19, 2000.

Ludo Waltman and Uzay Kaymak. Q-learning agents in a Cournot oligopoly model. *Journal of Economic Dynamics and Control*, 32(10):3275–3293, 2008.

Mirjam Wattenhofer and Roger Wattenhofer. Distributed weighted matching. In *Distributed Computing*, pages 335–348. Springer Berlin Heidelberg, 2004.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440, 1998.

Chen-Yu Wei, Yi-Te Hong, and Chi-Jen Lu. Tracking the best expert in non-stationary stochastic environments. *Advances in Neural Information Processing Systems*, 29: 3972–3980, 2016.

Qinggong Wu. A finite decentralized marriage market with bilateral search. *Journal of Economic Theory*, 160:216–242, 2015.

Junyi Xu. Reinforcement learning in a Cournot oligopoly model. *Computational Economics*, pages 1–24, 2020.

Jia Yuan Yu and Shie Mannor. Piecewise-stationary bandit problems with side observations. In *Proceedings of the 26th annual International Conference on Machine Learning*, pages 1177–1184, 2009.

Jinfeng Yue, Yu Xia, and Thuhang Tran. Selecting sourcing partners for a make-to-order supply chain. *Omega*, 38(3-4):136–144, 2010.

Dongbin Zhao, Haitao Wang, Kun Shao, and Yuanheng Zhu. Deep reinforcement learning with experience replay based on SARSA. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. IEEE, 2016.

Vita

Kshitija Taywade

Education

- B.E. Computer Science, Prof. Ram Meghe Institute of Technology & Research, India (June 2010 – May 2014)

Professional Positions

- Graduate Teaching Assistant, University of Kentucky (Jan 2019 – May 2023)
- Graduate Research Assistant, University of Kentucky (Aug 2018 – Dec 2018)

Publications & Preprints

- Siddique, A. B., Maqbool, M. H., Taywade, K., & Foroosh, H. (2022). Personalizing Task-oriented Dialog Systems via Zero-shot Generalizable Reward Function. In CIKM (pp. 1787–1797).
- Taywade, K., Harrison, B., Bagh, A., & Goldsmith, J. (2022). Modelling Cournot Games as Multi-agent Multi-armed Bandits. In The International FLAIRS Conference Proceedings (Vol. 35).
- Taywade, K., Harrison, B., & Goldsmith, J. (2022). Using Non-Stationary Bandits for Learning in Repeated Cournot Games with Non-Stationary Demand. arXiv preprint arXiv:2201.00486.
- Taywade, K., Goldsmith, J., & Harrison, B. (2021). Multi-agent Reinforcement Learning for Decentralized Stable Matching. In International Conference on Algorithmic Decision Theory (pp. 375–389).
- Taywade, K., Goldsmith, J., & Harrison, B. (2020). Decentralized Marriage Models. In The Thirty-Third International FLAIRS Conference.
- Taywade, K., Goldsmith, J., & Harrison, B. (2018). Decentralized Multiagent Approach for Hedonic Games. In European Conference on Multi-Agent Systems (EUMAS) (pp. 220–232). Springer.