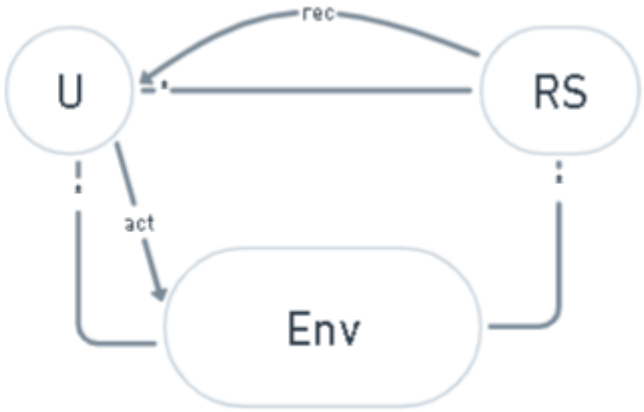
TÉCNICO LISBOA



Modelling the Recommender Alignment Problem

An Abstract Model and a Toy Experiment Where We Learn to Control Evolutionary Dynamics in Social Dilemmas Through Recommendations

Francisco Lopes Pereira de Carvalho

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Manuel Lopes
Prof. Francisco Santos

Examination Committee

Chairperson: Prof. Name of the Chairperson
Supervisor: Prof. Manuel Lopes
Members of the Committee: Prof. Name of First Committee Member
Dr. Name of Second Committee Member
Eng. Name of Third Committee Member

October 2021

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Acknowledgments

I would like to thank my parents, grandparents, siblings, and the rest of my family for their love and care all these years. I thank my girlfriend Beatriz for making my life so much better and for her indispensable emotional support.

To all my friends and colleagues who listened to countless explanations and helped me refine the ideas here presented, thank you. A special thanks to my friends Alexandre and Gonçalo for revising this document on such short notice.

I would also like to thank my advisors for their patience and assistance.

To each and every one of you – Thank you.

Abstract

Recommender systems (RS) mediate human experience online. Most RS optimize metrics that are imperfectly aligned with the best-interest of users but are easy to measure, like ad-clicks and user engagement. This has resulted in a host of hard-to-measure side-effects: political polarisation [Benkler et al., 2018], addiction [Hasan et al., 2018] [Andreassen, 2015a], fake news [Stöcker, 2020]. RS design faces a *recommender alignment problem*: that of aligning recommendations with the goals of users, system designers, and society as a whole. But how do we test and compare potential solutions to align RS?

Their massive scale makes them costly and risky to test in deployment. We synthesized a simple abstract modelling framework to guide future work. To illustrate it, we construct a toy experiment where we ask: "How can we evaluate the consequences of using user retention as a reward function?" To answer the question, we learn recommender policies that optimize reward functions by controlling graph dynamics on a toy environment. Based on the effects that trained recommenders have on their environment, we conclude that engagement maximizers generally lead to worse outcomes than aligned recommenders but not always. After learning, we examine competition between RS as a potential solution to RS alignment. We find that it generally makes our toy-society better-off than it would be under the absence of recommendation or engagement maximizers.

We aim for a broad scope, touching superficially on many different points to shed some light on how an end-to-end study of reward functions for recommender systems might be done. Recommender alignment is a pressing and important problem. Attempted solutions are sure to have far-reaching impacts. Here, we take a first step in the development of methods to evaluating and compare tentative solutions

with respect to those impacts.

Keywords

recommender systems; network science; evolutionary game theory; reinforcement learning

Resumo

Sistemas de recomendação (SR) mediam a experiência humana na internet. A maioria dos SR optimizam métrics que estão imperfeitamente alinhadas com os interesses dos seus utilizadores, mas são fáceis de medir, como *clicks* em publicidade e tempo de utilização. Isto tem resultado numa série de efeitos secundários difíceis de medir: polarização política [Benkler et al., 2018], vício [Hasan et al., 2018] [Andreassen, 2015a], notícias falsas [Stöcker, 2020]. *Design* de SR como um campo enfrenta um *Problema de Alinhamento de Recomendação*: a tarefa de alinhar recomendações com os objectivos de utilizadores, designers dos sistams, e a sociedade como um todo. Mas como podemos testar e comparar potenciais soluções para alinha SR?

A sua escala massiva significa que testá-los ao vivo é custoso e arriscado. Neste trabalho sintetizamos uma estrutura de modelação para guiar trabalho futuro. Para ilustrar esta estrutura, concebemos uma experiência-modelo onde perguntamos: ”Como podemos avaliar as consequências de usar ”utilização” como funcção recompensa no treino de SR?” Para responder a esta pergunta, aprendemos políticas de recomendação que optimizam funções recompensa controlando dinâmicas do grafo do ambiente através de recomendação. Baseando-nos nos efeitos que os recomendadores treinados têm nos seus ambientes, concluímos que maximizadores de utilização geralment levam a piores condições que recomendadores alinhados, mas nem sempre. Após a aprendizagem, examinamos o potencial papel de competição numa solução para o alinhamento de SR. Observamos que competição geralmente leva a melhores condições que monopólios de SR individuais.

Temos um foco abrangente, tocando superficialmente em vários pontos para mostrar como é que um estudo *end-to-end* sobre funções recompensa para sistemas de recomendação poderia ser feito. Alinhar sistemas de recomendação é um problema urgente e importante. Futuras tentativas terão decerto impactos a longo prazo. Aqui tomamos um primeiro passo no desenvolvimento de métodos para

avaliar e comparar potenciais soluções em função desses mesmos impactos.

Palavras Chave

sistemas de recomendação; redes complexas; teoria dos jogos evolucionária; aprendizagem por reforço

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

Contents

Recommender systems (RS) are software systems that assist users in interacting with large spaces of items, usually by presenting them with smaller personalized sets based on information such as past user behavior, user attributes, and features of the underlying items. User experience on social media, content platforms, and online stores is largely determined by RS.

Most recommender systems optimize metrics that are easy to measure and improve, like number of clicks, time spent, or number of daily active users. They are selected to do this by powerful optimization processes involving thousands of engineers and a significant fraction of global computing power. Goodhart's law [Goodhart, 1981] states that "when a metric becomes a goal, it ceases to be a good metric". In fact, choosing metrics that are imperfectly aligned with the best-interest of users has resulted in a host of hard-to-measure side-effects like political polarisation [Benkler et al., 2018], addiction [Hasan et al., 2018] [Andreassen, 2015a], fake news [Stöcker, 2020], fairness [Barocas et al., 2019] and diversity [Castells et al., 2015] concerns.

Recommender system design faces a *recommender alignment problem*: that of aligning recommendations with the goals of users, system designers, and society as a whole. We conceive it in analogy to the *value alignment problem* [Hadfield-Menell and Hadfield, 2019]: that of ensuring that an AI system's behavior aligns with the values of its principal.

A single reward function that would work for everyone forever is unlikely to exist. Thus a reward function for RS should be adaptive and error-correcting, drawing adjustments in a bottom-up fashion subject to users themselves. For example: recommender systems are currently unopposed in their respective networks. Would enabling competition between RS result in a sufficiently adaptive system?

But how do we test and compare potential solutions to align RS? How can we learn about their impacts on society and emergent effects? The massive scale of these systems makes them costly (due to the amount of resources involved), risky (due to the number of users they impact) and arguably unethical to test in deployment. Modelling and simulation approaches are used under similar constraints to study social dynamics so we consider them suitable for our case as well.

We present a modelling interface: a minimal set of requirements for models, derived from the literature around the societal effects of RS. The interaction between RS and society is a complex, multifaceted topic. What common properties of recommender systems are essential to study alignment, and which ones are domain-specific factors that should be left for implementation? We could find no prior efforts to base ourselves on, so we hope our interface can be relied on for future attempts to evaluate alignment techniques.

1.1 Context

Classically, the task of recommendation has been framed as that of providing ”relevant” items to users. In practice, RS are multi-stakeholder environments as illustrated in Figure 1.1, where multiple parties - users, content producers, and system operators - derive different utilities from recommendations, while the system as a whole effects externalities (collateral effects of an action on non-participants) on society. [Milano et al., 2020]

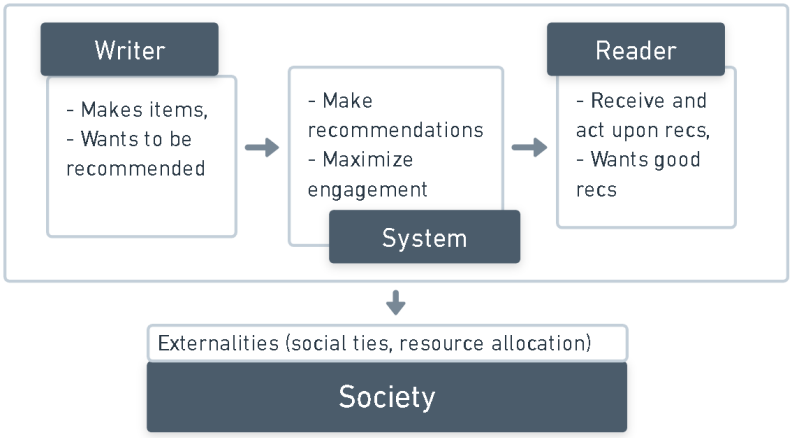


Figure 1.1: Recommender systems as peer-to-peer multi-stakeholder environments. [Milano et al., 2020] In p2p environments like Twitter, for example, users are both content producers and consumers with interests in both being recommended to other users, and to receive recommendations that advance their goals (entertainment, well-being, learning, etc). RS mediate information transfer between users and their non-local environment. The operators of RS have an interest in keeping users engaged, irrespective of whether that is in their best long-term interest or not. Finally, externalities from these interactions are expressed in society, as people take action as a result of things they learned, opinions they formed, or relationships they started online.

RS favor their operators [Burr et al., 2018b], who have the ability to tune and replace unfavorable RS. Social media platforms often make their content recommendations in ways that maximize metrics like ad-clicks or user engagement, claiming them to be used as proxies for relevance and user satisfaction. [Seaver, 2019] expressed this concisely: “Conflating retention and satisfaction has allowed developers to mediate the tension between users (whom they wanted to help) and business people who wanted to capture them.”

The information people are exposed to radically biases their beliefs, preferences and habits with both short-term and long-term effects. [Vendrov and Nixon, 2019] For example, proxy metrics used in social media, are only weakly correlated with what users care about, and users often regret time spent in social media. [Andreassen, 2015b] Beyond taking a pre-existing preference profile and tailoring recommendations to it, RS contribute to the construction of user identity dynamically. [Floridi, 2011] Preference drift from social interactions has been validated empirically. [Zafari et al., 2018] This begs

the question: Will RS be able to pursue their goals by strategically shaping our experiences to manage the evolution of our preferences?

Attempts to align RS with users have historically addressed problems reactively (only after they were already widespread) and one at a time (fake news, fairness, diversity, addiction, polarisation), often causing new harder problems. Other human-aligned design goals have also been pursued, like for user well-being, [Khwaja et al., 2019] or self actualization. [Knijnenburg et al., 2016] Higher-level approaches to RS alignment have been proposed. Namely: the incorporation of well-being metrics, participatory objective design, interactive value learning, and optimizing for informed and deliberative preferences. [Stray et al., 2021].

A higher-level approach to alignment that we have not seen discussed so far is that of promoting competition between RS. This could be done via regulation: e.g. mandating platforms support the use of third-party recommenders (Bring-Your-Own-RS). Even after techniques for building more aligned recommenders are developed, there's no guarantee they would be implemented by big platforms. Although some argue that it is in their interest to work towards the ultimate benefit of users and society, [Hohnhold et al., 2015] that might not be the case in the presence of strategic dynamics between stakeholders. [Kurland, 2019].

In our current setting, effective monopolies are protected by large-scale network effects - on internet utilities that should effectively be public goods (e.g. search, social networking, instant messaging). We expect that decoupling recommenders from applications and allowing users to choose from a marketplace of algorithms will lead to adoption of ever more aligned RS as the field progresses.

1.2 Contributions

Our works stems from the question "How can we evaluate potential solutions to the RS alignment problem?"

Our contributions lie in addressing the following questions. Our modeling interface is concerned with 1, while 2, 3, and 4 concern our toy-experiment.

1. **Modeling Interface**: How can models focus on dynamics relevant to RS alignment while abstracting over domain-specific factors?
2. **Toy environment**: How can we evaluate the societal implications of having user retention as the reward function for a recommender?
3. **Learning**: Can we learn policies that control graph dynamics to optimize arbitrary rewards?
4. **Competition**: Will competition between RS lead to better outcomes for society than recommender monopolies?

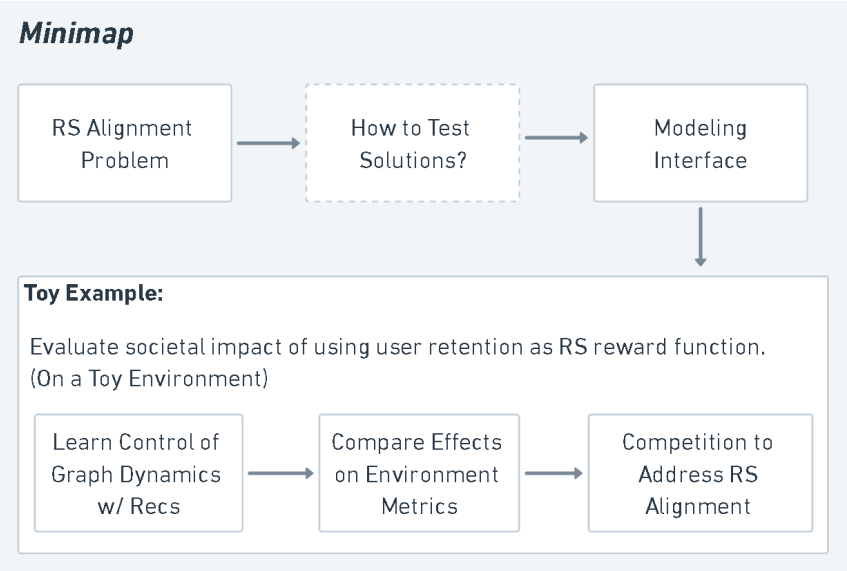


Figure 1.2: Mini-map of contributions. Potential solutions to the RS alignment problem will need to be evaluated with modeling and simulation. We construct a modeling interface based on qualitative literature on RS impacts. To illustrate its use, we run an end-to-end experiment to evaluate the effects of user retention as a reward function. We learn recommender policies aligned and misaligned with society in a toy environment and analyse relevant metrics. We further examine competition as a potential solution to alignment in our environment.

1. How can models focus on dynamics relevant to RS alignment while abstracting over domain-specific factors?

We review recent literature on societal impacts of RS to synthesize a modelling interface: a set of requirements that a researcher seeking to study RS alignment can use to build minimally complex models. Namely, a model must include a RS, users, and an underlying environment, as well as a few additional properties. To illustrate this model, we devise an experiment:

2. How can we evaluate the societal implications of having user retention as a recommender reward function?

To evaluate impacts of a proxy reward like user retention, we can compare it with an aligned reward function (assuming one can be defined). RS indirectly change their environment through users. We can train recommender policies to optimize reward functions through recommendations. These policies can be applied to a concrete environment, and their impacts on said environment measured.

To act on this plan, we construct an concrete toy environment consisting of a dynamic network of interacting users. Users can also rewire social ties based on recommendations. The task of RS is to make recommendations that tell users where to rewire. We define metrics of individual utility and social utility in the environment. Additionally, we pay attention to network topology (e.g. heterogeneity and degree distribution) as it can be compared with real world networks.

Dilemmas of cooperation have the benefit of having a natural socially aligned reward function. If strategies in a population converge towards cooperation, total payoff is maximized. However, we still don't have recommender policies that actually try to optimize reward functions.

3. Can we learn policies that control graph dynamics to optimize arbitrary rewards?

We learn MLP (multi-layer perceptron) and GNN (graph neural network) policies that rank nodes for recommendation. The top-ranked node is then recommended for rewiring. Proximal policy optimization (PPO) was chosen as the training algorithm for being a policy-gradient algorithm and simpler than the alternatives.

Baselines help interpret the performance of our learned policies. To test how our model behaves under different rewiring patterns, we apply simple heuristic policies and register relevant metrics. Heuristics are based on node strategy and degree.

Answering this question helps us make way in investigating whether RS will be able to pursue their goals by strategically shaping user experiences to manage the evolution of their preferences.

4. Will competition between RS lead to better outcomes for society than recommender monopolies?

Finally, we examine the effectiveness of competition as a higher-order intervention to align RS. We have different recommenders compete to understand which ones would dominate and whether unfavourable recommenders could be displaced by new better ones. We also study the impact of competition on cooperation and other metrics. We hypothesize that misaligned reward functions produce RS that lead to considerably worse outcomes than their aligned counterparts, and that competition would drive people away from misaligned RS, lending weight to the idea that people should be able to modularly switch which RS they use in the real world.

This dissertation begins with a short chapter (Chapter 2) describing our modelling interface, we review the literature to derive a set of requirements that a simple environment should fulfill to be used in studying user-RS misalignment. The three following chapters describe the experiment we run under the interface's conceptual fold. We present a model which fulfils those criteria in Chapter 3. In Chapter 4 we define the task of recommending in our toy environment as an MDP and then we train agents with selfish and pro-social reward functions using deep reinforcement learning and graph neural networks. Each section of the experiment has its own related work and background sections where necessary. Chapter 5 introduces a notion of competition between mediators to see if that would lead to better outcomes for society. Finally, we summarize our work and contributions and address future work in Chapter 6.

2

Modeling Interface

Contents

How can models focus on dynamics relevant to RS alignment, while abstracting over domain-specific factors? Finding no prior efforts to base ourselves on, we present a novel modeling interface intended to inform future modeling work on recommender alignment techniques.

This minimal set of requirements was derived from the literature around the societal effects of RS. We established a minimal set of relevant entities from prior views on RS as multi-stakeholder environments. [Milano et al., 2020] To abstract interactions between entities, we relied on an existing taxonomy of human interactions with Intelligent Software Agents (ISA). [Burr et al., 2018b]

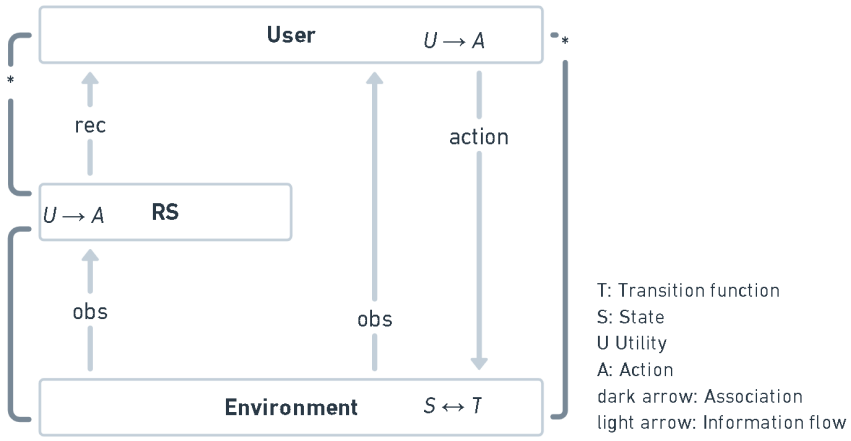


Figure 2.1: UML diagram of the relevant entities in our modeling interface: *Users*, *RS*, and the *Environment*. User has as any-to-one relationship with both RS and the environment. Both User and RS have *utility functions* that determine their actions. The environment has *State* and a *Transition function* $T : \{S, A\} \longrightarrow S$. Information about state is transmitted to User and RS via observations of different portions of the environment. RS make *recommendations* to Users providing extra information about State, Users *act* on the Environment.

2.1 Related Work

From our interpretation of RS as multi-stakeholder systems (fig. 1.1), we can tell we need to include at least users and RS in our model. And if we're looking for effects that differently aligned RS have on societies they mediate, we also need a model of an external environment that changes as users also change, so there needs to be interaction between users and environment. At this scale, users can be modelled as greedy agents [Calero Valdez and Ziefle, 2018] ; while recommenders can be modelled as the very types of algorithms that see deployment in the real world: i.e. ranking algorithms trained with reinforcement learning. [Calero Valdez and Ziefle, 2018]

In an existing taxonomy of interactions between humans and intelligent software agents [Burr et al., 2018a] both ISA and user may be assumed to pursue approximate maximisation of expected utility. (See Figure 2.2) In the context of recommender systems, both the user's and the RS's utility depends on the

state of the environment, which includes users and their actions.

RS can impose control over a system of users through trading (pursuing its own goals while increasing utility for the user) and nudging (exploiting user heuristics and biases to steer their behavior), with second-order effects over the user's beliefs and values.

The first takeaway from this taxonomy is that we need concepts of utility for RS, for users, and for society. And finally (and obviously) a notion of interaction between user and RS: a recommendation. Another way to think about recommendations is that of mediated information flow from RS to user.

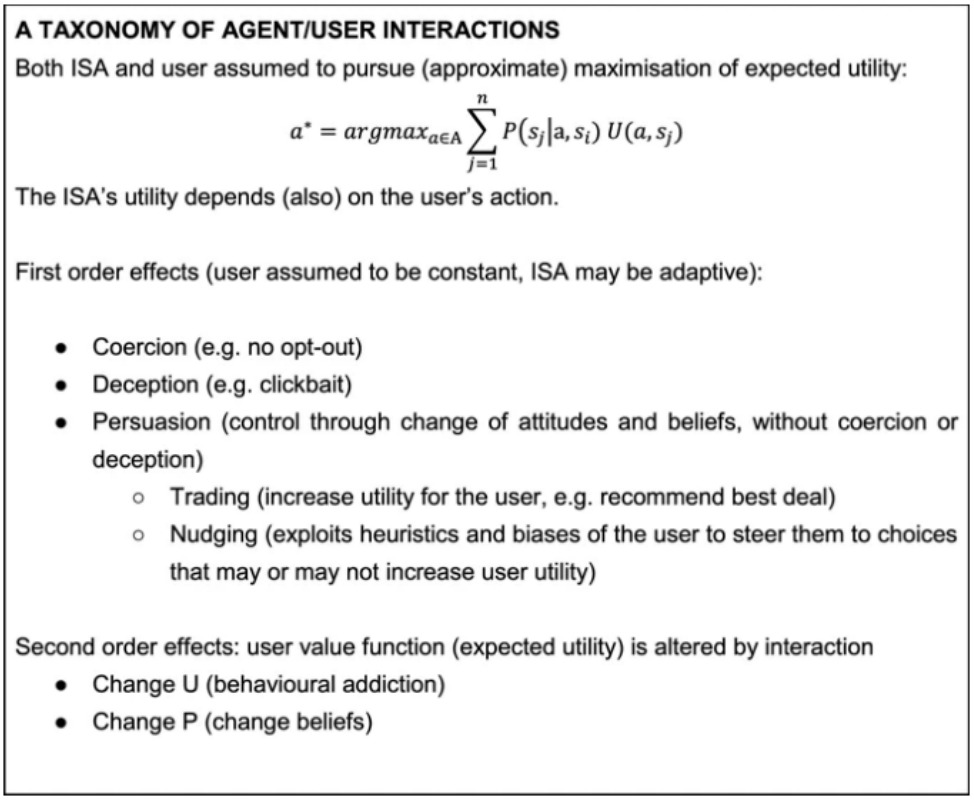


Figure 2.2: Taxonomy of ISA-human interactions. Software systems can exploit users for utility by leaving them no other options (coercion), or by deceiving them. More subtly, ISA can pursue their goals by controlling what users think is true (their world model P) and what users value (their utility function U) through persuasion. [Burr et al., 2018a]

2.2 Model requirements

We have discussed RS as multi-stakeholder environments where parties derive different utilities from recommendations. We have also reviewed how recommendations can determine interactions between users and their environment that then shape preferences and values of users. From our assessment, we compiled a minimal list of requirements that a model intended to study RS alignment should fulfill:

Table 2.1: Modelling interface requirements (see section 2.2).

Components
1a) | Environment
1b) | Users
1c) | Recommender
Interaction
2a) | User-RS interaction
2b) | User-Environment interaction
Utility functions
3a) | RS utility
3b) | User utility
3c) | Social utility

An Environment, Users with local information and Recommenders with system-level information. An environment should contain the users which act in it and provide observations for users and recommenders. The task of RS is to parse large spaces of items and present personalized subsets to users, who are unable to observe the whole space. It is sensible then to model users with local information and RS with access to more information, or even global information.

User-RS interaction: recommendations. RS mediate the relationship between users and their non-local environment. They do this by providing information about it in the form of a recommendation. Recommendations can be items of content or other users to interact with (as it happens with friend suggestions in social networks).

User-Environment interaction. The dynamical system of society. People exist and pursue their goals in the world. On social media platforms, experience consists of interacting with other users either directly or by consuming and producing content. The RS largely mediates this experience. Although an increasing part of people's lives and the economy is conducted online under mediation, the offline world represents a source and sink of externalities away from the reach of RS. As users interact with RS, they change the way they act in the world, and the environment is changed in turn. It's by measuring this change that we can evaluate the impacts of RS.

Utility functions for recommenders, users, and society. RS are selected by their operators to improve some metric, either by human selection or by reinforcement learning algorithms. In the case of many social media platforms this metric is a proxy that benefits whoever is in charge of the RS - usually related to click-through rate on advertisements or time spent by users on the platform. Defining utility for even a single individual is challenging under the full complexity of the real world. There has been work on optimizing for self-actualization or self-reported well-being, but ultimately, realistic notions of individual utility must include regular feedback and redefinition. A notion of utility for society is even more challenging to achieve. People often have incompatible goals and values that must be traded-off

even when making decisions with complete information. Nonetheless, we must assume these can be approximated and represented in models.

2.3 Discussion

We have not validated our framework empirically. To do it, we would need to be able to make predictions about the real world with regards to some question of RS alignment. That would require real data including metrics of long-term user satisfaction and or some metric of ”social benefit”, which we have not been able to find. However, we can try to explain some real world examples in terms of our abstractions.

Twitter: Users are Twitter accounts that produce and consume content. Twitter accounts can belong to individuals but also to corporations that have strong preferences for producing content rather than consuming it. Users have partial information as they have access to a tiny subset of all tweets, whereas the recommender is a centralized system pulling from all information on the platform. Twitter's recommendation system recommends users to each other by directly suggesting accounts to follow, or by displaying users' tweets to other users. Users interact with each other online, learning and updating their beliefs, preferences, and form relationships that express themselves in their offline behavior, which affects the world, the environment. Users are trying to be entertained, or learn, or reach people; while the recommender is maximizing ad-profits. Facebook works in a similarly.

Youtube still works in a similar way, but the conceptual divide between producer and consumer is harsher, as only a small proportion of users actually publishes videos. However, creators do influence each other online.

Amazon and e-commerce in general have a still deeper divide where the distinction between seller and customer is more complete, and there it would make more sense to have 2 types of populations than to model the system as peer-to-peer. Here RS are less motivated by ads than sales directly, and the influence recommendations can be more explicit through the careful selection of items and prices that users are shown. Users try to get the best value for their money, sellers try to get as much money from customers as possible, while the recommender is optimizing for sales in general (e.g. showing items frequently bought together).

We consider our model to be especially suited for modelling dynamics characteristic of recommender systems in peer-to-peer social settings, as that is what we choose to focus on. However, it can easily be extended by adding special kinds of user with different utility functions as in the case of dedicated producers and consumers.

3

Experiment: Environment

Contents

This chapter and the two following ones make up a toy experiment in RS alignment: "How can we evaluate the societal implications of having user retention as the reward function for a recommender?" We divide it in 3 steps: 1) define a toy model for society mediated by RS; 2) learn recommender policies aligned with social good and policies aligned with user engagement; 3) evaluate societal benefits of cooperation between recommenders.

In this chapter, we define a concrete model that implements the interface laid out in **??**. To do this, we extend an existing framework of networked evolutionary game theory [Santos et al., 2006b] where agents play dilemmas of cooperation, periodically rewiring social ties. Our main extension to the original model is allowing arbitrary rewiring policies to be used instead of the original default policy, thus allowing us to delegate the choice of new neighbors to a recommender system. Finally, we obtain baselines for the behavior of the system. To get a comprehensive picture, we simulate the environment as mediated by fixed rewiring heuristics using the information in strategies and node degrees.

3.1 Related Work

So far we have discussed RS as environments where stakeholders with different sets of interests interact. We've seen how recommendations can determine interactions between users that then shape preferences and values, and how the interests of RS operators and society have been misaligned in the past and present.

Strategic dynamics in content production and consumption may lead to the failure of classical principles of RS in maximizing social welfare. The need to avoid such a failure by revisiting those principles with game theory and multiple stakeholders in mind has inspired a whole research agenda. [Kurland, 2019]

In the same game theoretic paradigm, competing recommenders have been studied [Izsak et al., 2014] - although not with respect to externalities. Previous work has approached the cost that competition between strategic mediators would impose on the population of agents they mediate [Babaioff et al., 2015], although not in the context of recommendation systems.

Centrally to our model, work in evolutionary game theory has seen setups where networked agents play social dilemmas with rewiring of social ties [Santos et al., 2006b]. Users interact in 2-by-2 matrix games, and recommendations are represented by potential new neighbors to rewire to. If we focus on peer-to-peer dynamics where users are both consumers and producers (fig. 1.1), this setup can fit our requirements as a minimal base model - with a few adjustments. Recommendation mechanisms in spatial public goods games have also been modeled before, although the recommendations were made by other agents instead of by a central mediator. [Yang et al., 2013]

3.2 Model definition

We present a model of society consisting of an undirected network of agents playing 2-by-2 social dilemmas, evolving their strategies and rewiring social ties.

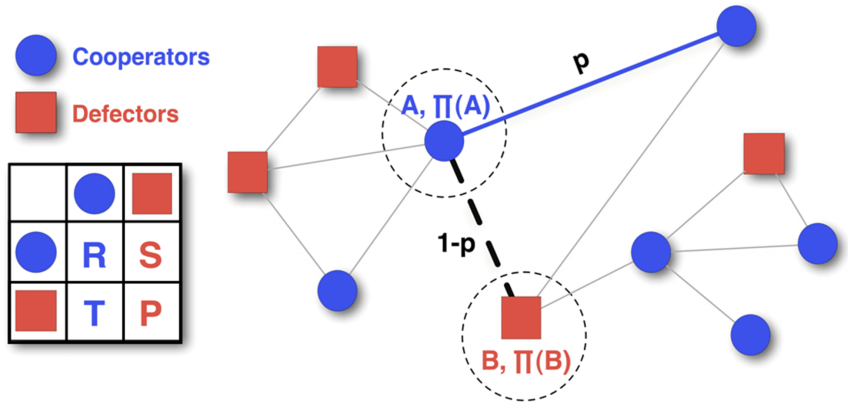


Figure 3.1: Evolving the neighborhood. [Santos et al., 2006b] Users play dilemmas of cooperation where they have two possible strategies: cooperate or defect. They receive payoff from playing other agents, which determines their fitness. A user A will also try to rewire their edge to B if B is a Defector. Whether a rewire succeeds depends on the difference of fitness of the nodes involved.

There are two types of individuals: cooperators and defectors, we call their strategies C and D respectively. They engage in social dilemmas of cooperation - specifically 2-player symmetric games - where players can either cooperate or defect when interacting. (see Figure 3.1) Individuals only interact with their neighbors on the network. Game strategies evolve in the population as users compare fitness and copy their neighbors' strategies. Individuals can also rewire their social ties if unsatisfied with their neighbors.

Network Users are connected to each other according to the edges of a network graph $G = V, E$ where V is the set of nodes and E is the set of edges. G is always initialized as a uniform random graph with average node degree k and its topology is allowed to evolve.

Games Agents interact by playing social dilemmas: symmetric, 2-player, 2x2 matrix games. (as seen in Figure 3.1) We normalize the difference between mutual cooperation (R) and mutual defection (P) to 1, making $R = 1$ and $P = 0$, respectively. As a consequence, games can be parameterized by two scalars: payoff T (temptation to cheat), which satisfies $0 \leq T \leq 2$ and payoff S (disadvantage of being cheated) satisfies $-1 \leq S \leq 1$. In this paper, we will focus on the Prisoner's Dilemma : $T = 2$, $S = -1$ as it is the hardest game to solve. We have however, explored the full space of parameters for some experiments.

Evolution The strategy of a node x evolves through imitation of a neighbor y. A node updates its strategy according to a Fermi update probability p (eq. (3.1)) based on the difference between the fitness of each player (f_A and f_B). [Traulsen et al., 2006] Fitness corresponds to the cumulative payoffs of a node, resulting from the sum of payoffs from playing each of one's neighbors.

$$p = \frac{1}{1 + e^{-\beta(f_B - f_A)}} \tag{3.1}$$

Rewiring Given an edge between A and B, we say A is satisfied with the link if B is a cooperator, being dissatisfied otherwise. If A is satisfied, they will keep the link. If dissatisfied, A will compete with B to rewire the link. (Figure 3.1) The action taken is contingent on the fitness $\Pi(A)$ and $\Pi(B)$ of A and B respectively. A redirects the link to a new neighbor given by its rewiring strategy with probability p given by eq. (3.1). With probability $1 - p$, A either stays linked to B - if A is a cooperator - or B rewires its link with A to one of A's neighbors. We call this rewiring a *structural update*.

Time-scale Strategy evolution and structural evolution can occur at different time-scales, T_a and T_e respectively (if $T_a = 2 * T_E$, strategy updates occur twice as often as structural ones). The ratio $W = T_e/T_a$, leads to different outcomes for cooperation. In realistic situations, the two time-scales should be of comparable magnitude. W serves as a measure of agents' inertia to react to their conditions: large values of W reflect populations where individuals - on average - react promptly to adverse ties, whereas smaller values reflect some inertia for rewiring social ties.

3.2.1 Simulation algorithm

The pseudo code for the update process in our simulations is described in Algorithm A.1. $fermi(A, B, beta)$ is the function that calculates eq. (3.1) given A, B, and temperature term $beta$. $cumulativePayoff(x)$ returns the sum of payoffs a node x gets after playing a game with each of its neighbors. W is the ratio between the timescales of structural evolution and strategy evolution: $W = T_e/T_a$. $Strat$ is a vector of strategies (taking values in $\{C, D\}$) and $rewireStrat$ is a vector of rewiring strategies, each of these has a length $\#V$. A rewiring strategy is a function $R : \{x, y\} -> z$ that provides a recommended node z given a focused node x and the neighbor being rewired y. $doRewire(G, x, y, z)$ deletes the edge (x, y) from G and adds edge (x, z).

3.2.2 Metrics

We introduce the metrics we are interested in studying as we vary rewiring policies.

Cooperation and payoff Our chosen utility functions for society. Summing payoffs is the trivial way of aggregating individual utility. Also, when cooperation dominates as a strategy, it maximizes total payoff.

Algorithm 3.1: Simulation algorithm

```
begin
    while t < timeLimit do
        x ⟵ randomSample(V)
        y ⟵ randomNeighbor(x)
        P_x, P_y ⟵ cumulativePayoff(x), cumulativePayoff(y)
        p ⟵ fermi(P_x − P_y, beta)
        if random(0,1) < (1 + W)^{-1} then
            if random(0,1) < p then
                Strat_x ⟵ Strat_y
        else
            if Strat_y == D and Strat_x == C then
                if random(0,1) < p then
                    z ⟵ rewireStrat_x(x, y)
                    doRewire(G, x, y, z)
            if Strat_y == D and Strat_x == D then
                if random(0,1) < p then
                    z ⟵ rewireStrat_x(x, y)
                    doRewire(G, x, y, z)
                else
                    z ⟵ rewireStrat_y(y, x)
                    doRewire(G, y, x, z)
```

We measure cooperation as the mean number of times a population converges to full cooperation. When it doesn't, we average the fraction of cooperators of the final 100 steps.

Heterogeneity and Cumulative Degree Distribution The topology of networks has many interesting implications. Heterogeneous graphs have been shown to improve the survival of cooperation. [Santos et al., 2006a] Thus, we compute the heterogeneity of the graph (eq. (3.2)) where N_k gives the number of vertices with k edges and z is the average connectivity of nodes in the graph. We additionally compute the cumulative degree distribution $D(k) = N^{-1} \sum_{i=k}^{N-1} N_i$. k_{max} - the maximum value for node degree in the graph also provides a simple measure of heterogeneity.

$$h = N^{-1} \sum_{k} k^2 N_k - z^2 \tag{3.2}$$

Rewire rate We're interested in user-engagement because it's a metric used to optimize current RS, effectively tracking use of the RS itself. It's natural to make a parallel with the number of rewire attempts to obtain a metric for the self-interest of mediators. Effectively, this number encodes the fraction between the number of "rewire attempts" (structural updates where the neighbor of the node in focus is a defector, thus triggering a rewire) and the total number of opportunities. When varying the timescale term W, in

order to track the ”rewire rate” caused by a mediator, we must consider that W will modulate the number of opportunities and so we obtain the $\#opportunities = T/(1-(W+1)^{-1})$ where T is the total time the simulations run for. Finally we get $rewireRate = \#attempts/\#opportunities$.

3.2.3 Discussion

Why this environment instead of others? This one has many benefits in common with other agent-based models: Having been studied in prior work, being inexpensive to simulate, multiple individual agents, greedy agent behavior with complexity coming from scale. But why this one in particular? Adapting *Sugarscape* [Epstein and Axtell, 1996] was a possibility. We figured that dilemmas of cooperation were a reasonable abstraction for important human interaction, and its graph topology more closely resembles the social media environments on which recommendation takes place as well as the relevant social reality.

Table 3.1: Translation table between problem and model space. The third column is a concrete example about the real world.

Problem space	**Model space**	**Twitter example**
Recommenders	Rewiring policies	Twitter RS
Users	Nodes	Users
Environment	Nodes in Network	World
User-environment interaction	2x2 game with neighbor	Action online or offline
Item recommendation	Neighbor recommendation	Tweet or User recommendation
User utility	Game payoff	Long-term life satisfaction (?)
Societal utility	Cooperator ratio	?
User engagement with RS	Number of rewires	Time spent on website

How this EGT framework implements our modelling interface:

1. The environment consists of the whole graph: nodes, and strategies. Users correspond to individual nodes on the graph and RS correspond to rewiring policies for nodes. Users only have information about the strategies and fitness of their neighbors. RS are assumed to have complete information, for simplicity. leverage complete information to make rewiring recommendations from the whole graph. Note that we consider the environment to include both online and offline experience, modelling only the acquisition of new information as a recommendation.

2. RS interact with agents by providing recommendations of new neighbors for users to rewire to. Although in some realistic settings, recommendations in social media consist of items produced by users, we abstract over these. An edge between users represents exposure to another user's content, or repeated interaction.

3. Interactions between users and their environment comes in the form of the games users play among themselves. Players derive utility from the results of interactions. Agents are both providers and consumers of content, as occurs in peer-to-peer social networks. (fig. 1.1)

4. The natural utility function for agents is their payoff obtained from games, whereas a social utility function can aggregate individual payoffs. Another natural social utility function is the prevalence of cooperation in the population, which maximizes aggregate payoff. Utility for mediators can be defined as any function of the environment: final number of cooperators, total number of rewires, total agent payoff, etc.

Criticism: One criticism of the toy model is that users only want to rewire when neighboring a defector. This isn't the only reason real people look for recommendations. It makes it so that optimizing for engagement has extra incentive to optimize for defectors.

In an abstract sense, you could say people look for recommendations to explore. Because they're dissatisfied with their current context (i.e. are connected to defectors) - and thus an RS maximizing for engagement would have to promote dissatisfaction with local context as well (i.e. defectors). However, the fact that our simulations are stopped when strategies fully converge limits the incentive to promote defection, lest the system be doomed too quickly. Additionally, as we'll see in table 4.4, it is not always the case that the policy maximizing rewires promotes defectors, but rather that it should promote whatever will prevent the system from converging.

One might also take issue with 2-player social dilemmas and specifically the prisoner's dilemma as an abstraction over general human interaction. However, easier interactions are more likely to lead to good outcomes, so it seems more relevant to focus on solving more difficult interactions.

3.3 Baselines

To get a sense for the environment's behavior under different rewiring dynamics, we define a space of heuristics based on node strategy, run simulations, and track our chosen metrics. For early runs, we plot a 2D space of game parameters T and S, before focusing in on the prisoner's dilemma (T=2, S=-1) as it's the most difficult setting to "solve". We start by showing how the environment behaves in the absence of rewiring, followed by a more complete treatment of a few select heuristics. Then a look at the full space of heuristics focused only on the PD.

3.3.1 Exploring Game-Space and Rewiring Policies

In this section we present results from simulations over the space of game parameters T,S. We evaluate a small set of recommender heuristics: BAD (always recommends random defectors), RANDOM (always recommends random nodes), GOOD (always recommends random cooperators), NO_MED (local heuristic), and FAIR (recommend random cooperators to cooperators and defectors to defectors). We observe an ordering by speed of convergence towards cooperation.

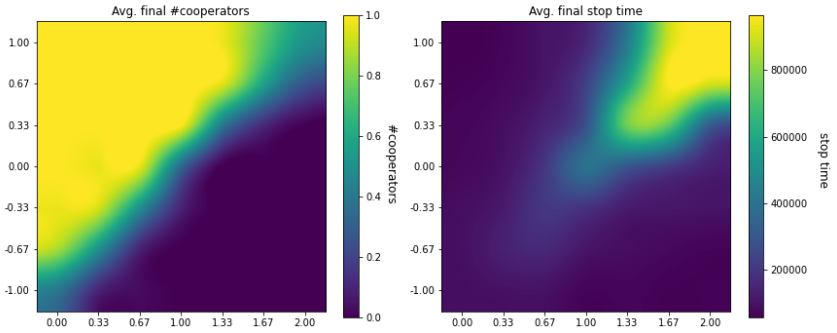


Figure 3.2: Evolution of cooperation in a uniform random network in the absence of rewiring (W=0). On the left: average final number of cooperators. On the right: final stop time. Both plotted as a function of two game-parameters: S, the disadvantage of a cooperator being defected (when $S < 0$), and T, the temptation to defect on a cooperator (when $T > 1$). Absent any of these threats ($S \geq 0$ and $T \leq 1$; upper-left quadrant) cooperators trivially dominate. The lower-left quadrant ($S < 0$ and $T \leq 1$) corresponds to the Stag-Hunt dilemma, by definition. The lower triangle in the upper-right quadrant ($S \geq 0$, $T > 1$ and $(T + S) < 2$) corresponds to the Snowdrift game, also by definition. The lower-right quadrant ($S < 0$ and $T > 1$) corresponds to the Prisoner's Dilemma domain (PD). (Left) [Santos et al., 2006a]

Local heuristic The local heuristic corresponds to the default rewiring policy defined in [Santos et al., 2006b]. Given an edge (A, B), node A rewires to a random neighbor of node B. The intuition behind this reasoning is that simple agents, being rational individuals with partial information, are more likely to interact with nearby agents [Kossinets and Watts, 2006]. Moreover, selecting a neighbour of an inconvenient partner is also a good choice, since this partner also tries to establish links with cooperators, making it more likely that the rewiring results in a tie to a cooperator.

$$BAD < NO\_MED < FAIR < RANDOM < GOOD \tag{3.3}$$

Introducing rewiring ($W > 0$) according to NO_MED to the experiment in fig. 3.2, we recover the result from [Santos et al., 2006b]. fig. 3.3 shows the fraction of successful evolutionary runs ending in 100% cooperation for different values of the time-scale ratio W. Above a critical value ($W_{critical} \approx 4.0$) cooperators efficiently wipe out defectors.

We plot the evolution of cooperation for each of our types of mediator. (fig. 3.4) As expected, BAD produces little change in outcomes. We expected FAIR mediators to incentivize cooperation even more than RANDOM or even GOOD, as these didn't punish defection, but this was not the case. Our sense is that FAIR might isolate cooperators from defectors, leading to defectors having no cooperators to imitate. RANDOM doing better than NO MED also shows that escaping one's neighborhood by rewiring to a random place in the whole graph can lead to faster convergence.

We also plot our metrics solely for the Prisoner's Dilemma (T=2,S=-1) as a more granular function of W, (fig. 3.5) to better compare the convergence speed under different mediators.

In terms of rewiring, BAD leads to the highest rate, while GOOD produces the lowest, with the

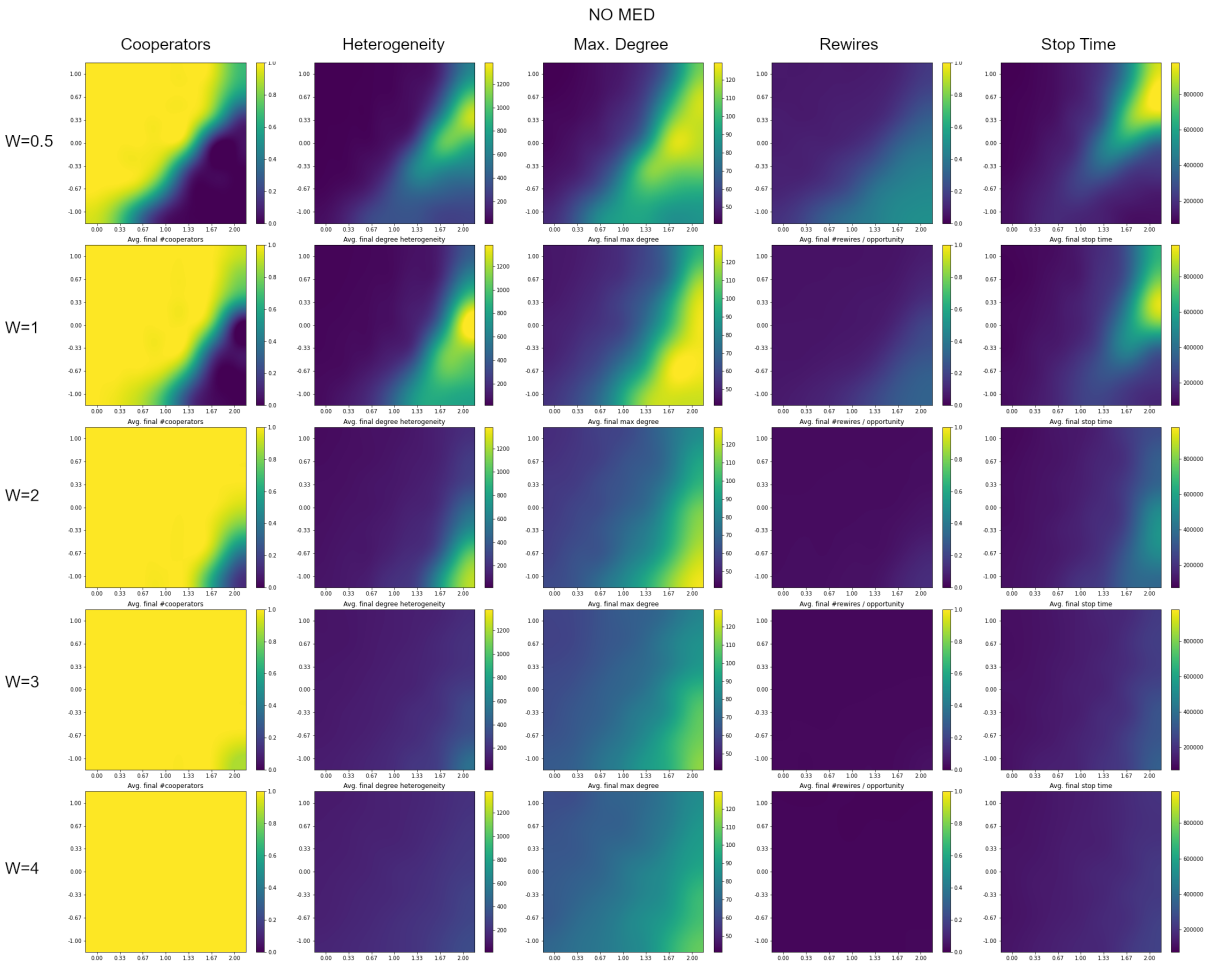


Figure 3.3: Co-Evolution of strategy and rewiring according to the local heuristic (NO_MED) for different dilemmas (t,s parameterizations) and time scales. Each row: W=0.5,1,2,3,4, each column: final fraction of cooperators, final heterogeneity, final k_{max}, rewire_n.

remaining mediators being comparable (fig. 3.5). One of the reasons is that agents will only want recommendations if the neighbor they're rewiring is a Defector, so a population with more Defectors will want more rewires, while a population that converges to full Cooperators will cease to want new neighbors.

NO MED produces by far the most heterogeneous graphs, presumably due to its local rewiring dynamics. (fig. 3.6) Among the fixed mediators, GOOD produces the highest heterogeneity around the prisoner's dilemma (T=2, S=-1) and especially at W= 0.5 and W=4.

All this means we can expect that recommending Cooperators will drive convergence to cooperation, and that recommending defectors will drive high rewiring numbers until convergence to full defection. We can see the prisoner's dilemma is the hardest setting for each of these rewiring policies and that there is no overwhelming convergence for any of them when $W = 1$.

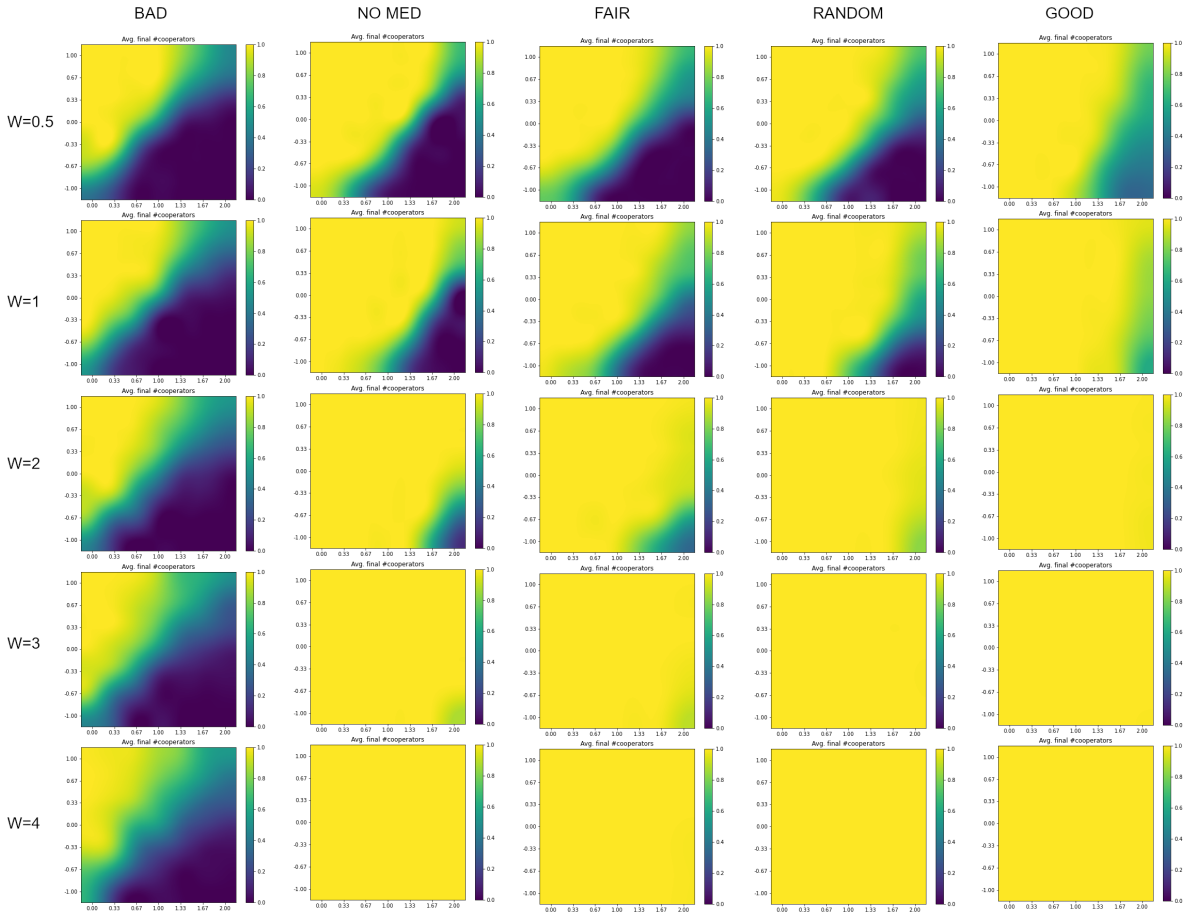


Figure 3.4: Co-Evolution of cooperation for Different Dilemmas (t,s axes) and Time Scales (rows), for Different Mediators (columns). We can verify there's an ordering of converging speed as W increases: $BAD < NO\_MED < FAIR < RANDOM < GOOD$.

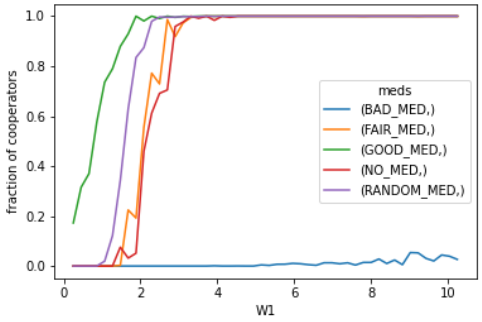
Unexpectedly, we observed FAIR does worse than RANDOM with respect to convergence towards cooperation. We believe this is because defectors and cooperators become segregated making it more difficult for defectors to convert. This hasn't been tested but we would start by computing modularity-based community finding and measuring the average strategies in each community.

Table 3.2: Possible heuristics as a combination of degree and strategy when N=500.

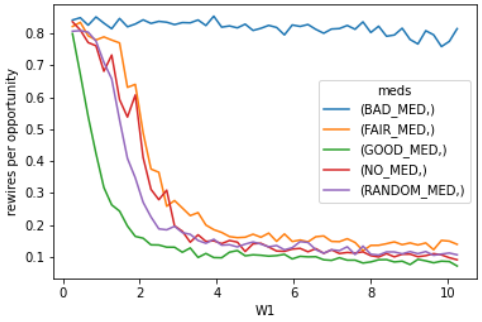
strat x degree	Low	Random	High
Defectors only	X	X	X
Random	X	X	X
Cooperators only	X	X	X

t

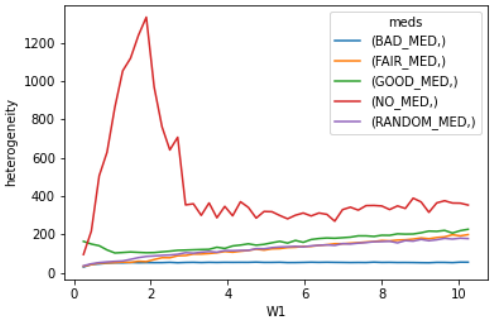
All the heuristics we used were global in scope, having a homogenizing effect, thus NO_MED generally led to higher heterogeneity than any of them. These initial baselines were obtained from a set of 5



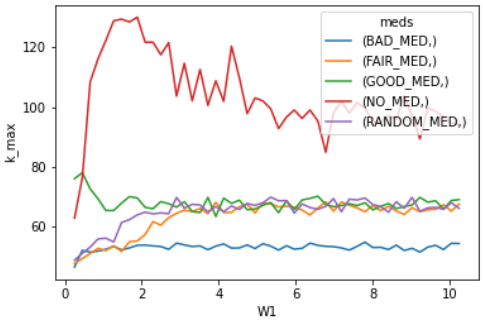
(a) Final fraction of cooperators. The order of convergence speed can be more clearly observed.



(b) Number of rewires per rewire opportunity. (agents may decide not to rewire if they're satisfied with a neighbor)



(c) Heterogeneity. NO MED clearly leads to more heterogeneous networks. We expect this to be due to its local focus, rather than global recommendations.



(d) Max degree, another metric of heterogeneity. We can see BAD leads to the absolute lowest heterogeneity.

Figure 3.5: Co-evolution of cooperation and structure in the prisoner's dilemma (t=2,s=-1) as a function of W for different mediators.

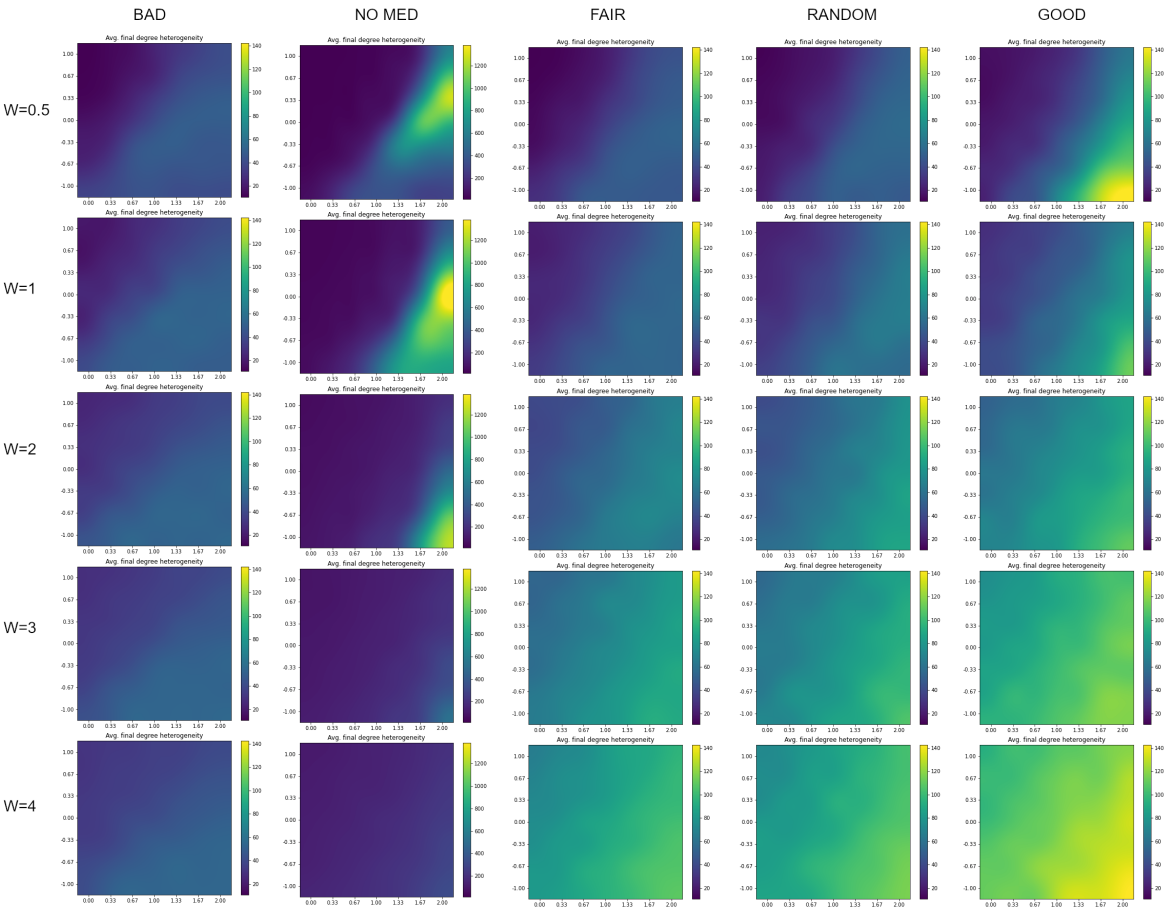


Figure 3.6: Heterogeneity for Different Dilemmas (t,s axes) and Time Scales (rows), for Different Mediators (columns). NO_MED produces by far the highest heterogeneity due to the local nature of its rewiring (as opposed to the other strategies which sample the graph globally) and it is the only column at a different value scale, otherwise we wouldn't be able to discern value differences in the other plots.

heuristics. The following chapters include a larger set of heuristics based on their strategies and degree information.

For the rest of the experiment, we'll consider a 3-by-3 space of heuristics plus the null policy (no rewiring) and the local heuristic defined in **??** .

Table 3.3: Baselines for N=500. Null policy (no rewires), Local heuristic, Strategy x Degree heuristic combinations. Values are averages over 1000 runs of: final number of cooperators, number of rewires per episode, final graph heterogeneity, time steps per episode, and mean action degree. Standard deviation in parenthesis. Max values for our training metrics #coops and #rewires are printed in bold for heuristics and learned policies.

	Heuristic	Coops	Rewire	Action Strat	Action Degrees	Het	Time
	Null	0.016±0.023	21665±2752	-	-	0.00	28113
	Local	0.596±0.241	**22505±2755**	0.36	148.40	4870.98	76570
Defectors							
	Low	0.014±0.016	21624±2570	0.00	29.00	0.01	28018
	Random	0.018±0.025	21502±2800	0.00	29.92	41.82	28176
	High	0.388±0.287	13519±3120	0.00	187.84	5353.50	29238
Random							
	Low	0.015±0.019	21612±3040	0.12	29.00	0.01	28153
	Random	0.137±0.084	18475±2460	0.13	29.92	52.63	29968
	High	0.440±0.300	11805±2640	0.27	195.90	5690.20	29550
Cooperators							
	Low	0.499±0.086	10936±1410	1.00	28.00	8.94	30000
	Random	0.518±0.087	10601±1220	1.00	31.42	103.34	30000
	High	**0.647±0.219**	8552±1540	0.93	189.55	4792.59	29557

4

Experiment: Learning to Control Graph Dynamics

Contents

Our goal in this section is to use reinforcement learning to learn two kinds of recommender policies: *aligned* policies that maximize the final **number of cooperators**, and *engagement-maximizer* policies that maximize the **number of rewires** that take place over runs. We want to learn them so that we can compare their behavior and effects on their environment. Specifically with regards to cooperation, #rewires, and network topology.

We begin by covering related work and some background knowledge on graph neural networks and the proximal policy optimization algorithm (PPO) for reinforcement learning. Then we describe our environment, training architecture, and finally present learning curves for both reward functions on environments of different sizes.

4.1 Related Work

The processes through which real-world RS are optimized may already explicitly involve reinforcement learning, [Chen et al., 2020] or at least be optimized implicitly through the selection that engineers make when designing and choosing which models to deploy. Deep RL has been applied to large discrete action spaces [Dulac-Arnold et al., 2016] and specifically to train recommenders: using RL to consider the long-term effects of recommendations [Liu et al., 2019], and optimizing them for long-term user engagement [Ie et al., 2019].

citations needed

To the best of our ability, we couldn't find previous work using RL to solve a task of mediation in evolutionary game theory. The closest application we've been able to find was RL being used to learn policies of partner selection for individual agents in the iterated prisoner's dilemma. [Anastassacos et al., 2020]

Graph neural networks (GNN) preserve useful symmetries characteristic of graphs, making them ideal to handle graph data. They have also been used in conjunction with deep reinforcement learning to solve graph optimization problems. [Darvariu et al., 2020]

We base our training architecture on recent work on the control of dynamical processes in graphs through node-level interventions, [Meirom et al., 2021] and train it with Proximal Policy Optimization (PPO) [Schulman et al., 2017] to obtain rewiring strategies that optimize different metrics in our model. We require a policy gradient algorithm instead of an action-value approach like Q-learning because policy gradients don't require us to assign an approximate value to each possible action, as the action space is prohibitively large (number of nodes). We choose PPO because it is one of the simplest and widely used policy gradient RL algorithms.

Comparing differently aligned mediators requires comparing policies trained on different reward functions. Several ad-hoc methods have been used to compare reward functions in work around reward learning. The earliest work on inverse reinforcement learning (IRL) evaluated rollouts of a policy trained

on the learned rewards, [Ng and Russell, 2000] while recent work compared reward functions by simply making scatter plots of returns. [Ibarz et al., 2018]

4.2 Background: Graph Convolutional Networks

Graph neural networks are neural networks that operate on graph data. [Kipf and Welling, 2017] They can also be seen as optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances). GNNs offer us a suitable architecture for our observations - which consist of a graph (adjacency matrix) and node features. Let us establish some notation: G is an undirected graph with nodes and edges: $G = (V, E)$, where V is the set of all nodes n and E is the set of all edges $(n_i, n_j) \in E$.

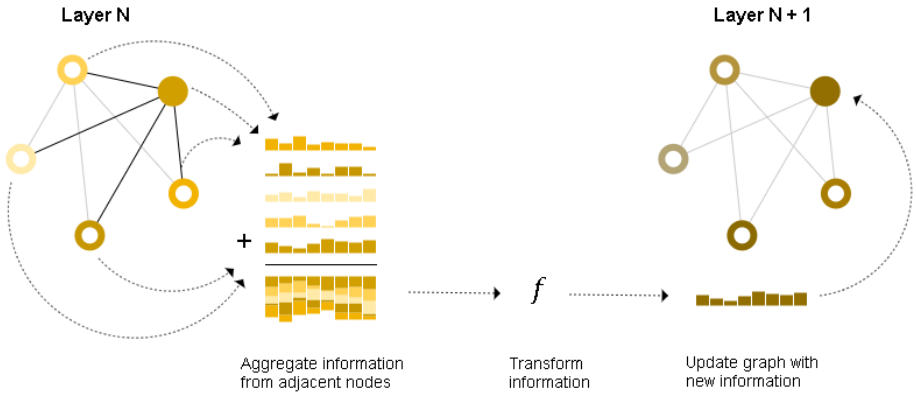


Figure 4.1: Message-passing in graphs. Message-passing in graphs generalizes convolution. [Sanchez-Lengeling et al., 2021]

A graph convolutional network is one of the simplest examples of GNN. It uses a mechanism called message passing that allows each node in a given graph to receive information about its direct neighbors. Message passing works in three steps:

1. For each node in the graph, gather all the neighboring node embeddings (or messages), which is the g function described above.

2. Aggregate all messages via an aggregate function (like sum).

3. All pooled messages are passed through an update function, usually a learned neural network.

It can be summarized as follows:

$$h_i^{(l+1)} = \sigma\left(b^{(l)} + \sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ji}} h_j^{(l)} W^{(l)}\right)$$

where $\mathcal{N}(i)$ is the set of neighbors of node i. c_{ji} is a normalization term, usually defined as the product of the square root of node degrees (i.e., $c_{ji} = \sqrt{|\mathcal{N}(j)|}\sqrt{|\mathcal{N}(i)|}$, W is the weight matrix of the

layer, and σ is an activation function. Edge features can also be considered but we don't make use of them so they are left out of the explanation.

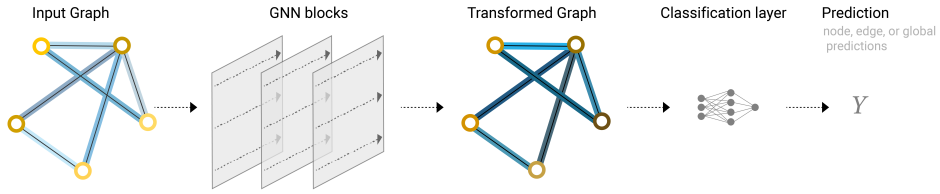


Figure 4.2: An end-to-end prediction task with a GNN model. [Sanchez-Lengeling et al., 2021]

The concept behind a GCN is similar to the local receptive field in convolutional neural networks: Each node of a GCN layer only receives information about its immediately adjacent neighbors. Information from beyond this receptive field may only reach it in the next GCN layer. The larger the graph grows, the more layers are needed to receive information from farther nodes. fig. 4.1

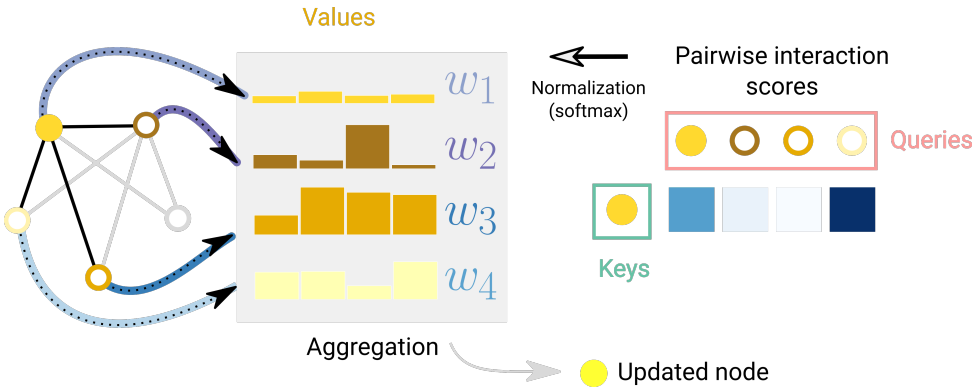


Figure 4.3: Schematic of attention over one node with respect to it's adjacent nodes. For each edge an interaction score is computed, normalized and used to weight node embeddings. [Sanchez-Lengeling et al., 2021]

Another way of communicating information between graph attributes is via attention. [Vaswani et al., 2017] For example, when we consider the sum-aggregation of a node and its 1-degree neighboring nodes we could also consider using a weighted sum. One approach to obtain these weights is to consider a scoring function from pairs of nodes to scalar. In this case, the scoring function can be interpreted as a function that measures how relevant a neighboring node is in relation to the center node. This concept is the basis of Graph Attention Networks (GAT) [Veličković et al., 2018]. Permutation invariance is preserved, because scoring works on pairs of nodes. A common scoring function is the inner product and before scoring, nodes are often transformed into query and key vectors via a linear map to increase the expressivity of the scoring mechanism.

4.3 Background: Proximal Policy Optimization

PPO can be described as a model-free, on-policy, actor-critic based reinforcement learning algorithm. Let's take a closer look at these features: [Schulman et al., 2017]

RL algorithms can be model-free or model-based: In model-based algorithms, the agent either has access to an accurate model of the environment's dynamics (state transitions and rewards), or learns it explicitly. The model-free approach does not attempt to explicitly understand the environment by first learning a model of the underlying dynamics. Instead, the algorithm simply gains experience by interacting with the environment to determine the optimal strategy.

Algorithms can be either off-policy or on-policy: Which refers to whether action selection during learning comes from the learned policy or not. In off-policy learning, actions during training come from π_{behave} which differs from π_{target}, the policy being learned. Take Q-learning: We want to optimize a greedy policy (given a state, we always choose the action that maximizes the value-function), but in order to gain experience, the agent follows a ϵ-greedy policy. In on-policy algorithms like PPO, $\pi_{behave} = \pi_{target}$.

Algorithms can be value-based, policy-based, and actor-critic based: Value-based learn a value function from which the policy can be derived by choosing the action that maximizes this function. In contrast, policy-based algorithms learn the policy itself. Actor-critic methods combine the advantages of both: The policy is learned directly without having to derive it from a value function, and a value function is learned to evaluate the policy.

PPO updates its policy weights θ via

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathrm{E}}_{s,a\sim\pi_{\theta_k}} \left[L\left(s, a, \theta_k, \theta\right)\right]$$

taking multiple steps of SGD (Stochastic Gradient Descent) to maximize the long-term reward by maximizing the objective function:

$$L\left(s, a, \theta_k, \theta\right) = \min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s, a), g\left(\epsilon, A^{\pi_{\theta_k}}(s, a)\right)\right)$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0 \end{cases}$$

where ϵ is a small hyperparameter that constrains how far a new policy can go from the old one.

$L(\theta)$ is called the clipped surrogate objective and is calculated by dividing the probability of an action a under the current policy π_{θ_t} by the probability of a under the previous policy $\pi_{\theta_{t-1}}$. This ratio is then multiplied with the estimated advantage function A. $A(s,a) = Q(s,a) - V(s)$. The advantage function

captures how much better an action is compared to the average of all rewards (caused by all possible actions taken) at a state s. Without any constraint, maximizing this result would lead to a huge policy update that might prevent it from converging. In order to ensure that π does not change too much, the ratio is clipped. (the $\min$ operation)

4.4 Environment

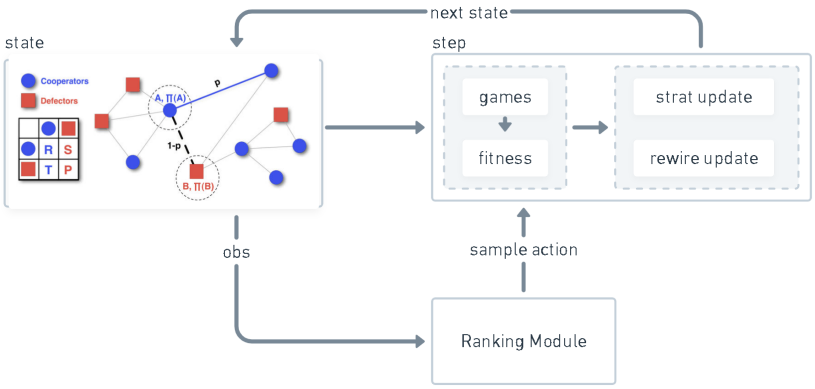


Figure 4.4: Structure of our training loop. Observations at each time-step t contain a focused node x, the graph adjacency matrix, and node features. The action consists of selecting a node z to which x will rewire one of its edges. x then goes through a strategy update. $W = 1; T, S = 2, -1$.

Our training environment consists of the model described in Chapter 3, where at each time-step a node x is randomly selected for either a strategy or rewire update. Actions consist of recommending a node for x to rewire to. Observations come in the form of a focused node x, an action mask, an adjacency matrix A, and individual node features x_{ifeat}. Node features consists solely of node strategy and node degree.

In chapter 3 we spoke of 1 update per step, whether it was a strategy or a rewire update. Since our task is recommendation, our training environment counts steps in a different way. One training step always corresponds to a rewire request, with potentially many strategy updates in between. The timescale ratio is still respected, and we focus on W=1 for the whole chapter. We also focus only on the prisoner's dilemma (T=2, S=-1) for the rest of our work. An episode ends when strategies converge (all D or all C) or when they reach a time limit.

Since there is little precedent for this task in RL, we gradually train policies at increasing values of network size N. Each N is accompanied by environment parameters such that a random baseline produces near 0.5 average final cooperators. If a given parameter setting made every heuristic converge towards 1, it wouldn't be a very challenging task, and if they all converged to 0, it might not be possible to solve. For $N \geq 100$ we also introduce a phased reward scheme where we provide 0.1 reward for every

Table 4.1: Environment configurations. N is the number of nodes in the graph, β is the temperature parameter for the fermi eq. (3.1) expression, k is the average node degree. The time limit is the number of steps the environment can take before a simulation is ended (Simulations usually finish early). Environment steps equal number of strategy plus rewire updates.

N	β	k	time limit
10	0.1	4	1000
30	0.05	8	3000
100	0.005	28	10000
500	0.005	30	30000

$N/10$ #coop above $N/2$ the agent gets.

4.5 Training Architecture

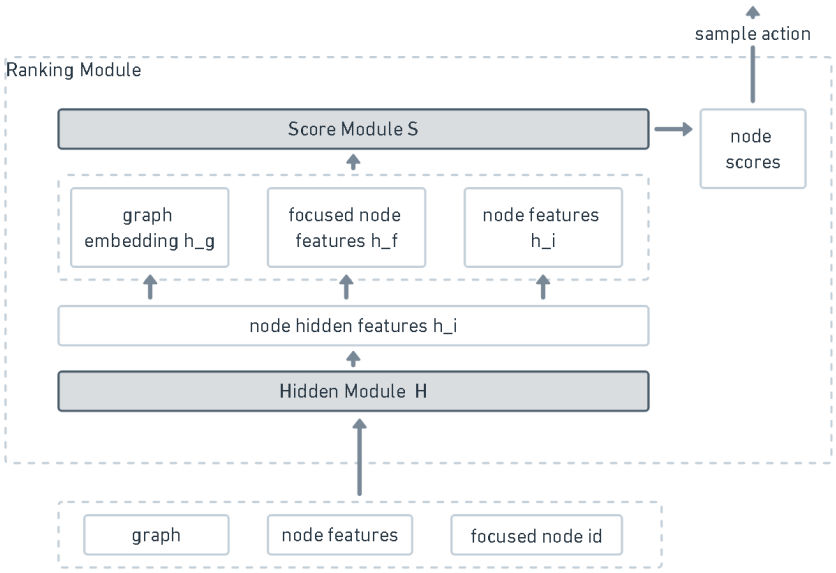


Figure 4.5: Ranking Module. It is composed of a hidden layer H that can be a MLP or a GAT, and a score module S that is a single-layer MLP. h_i stands for the hidden features corresponding to node i, h_f is the same thing where f is the id of the focused node. g_f correspond to the graph hidden features, obtained by aggregating all node hidden features. Node scores are then used to form a distribution from which the action is sampled.

Our recommender policy consists of a ranking module that scores each node in the graph. Scores generate a probability distribution over nodes from which actions are sampled. During training a Multinomial distribution is used, while in evaluation, the action is simply the *argmax* of scores.

The Ranking Module includes a hidden module H that produces node embeddings, and a score module S which calculates scores for each node. H can be either a MLP (Multi layer perceptron) or a GAT (Graph Attention Network). MLP is faster to train but considers only individual node features,

while GAT is more expressive but slower to train, relying on message passing and learning to weigh the contributions of each node's neighbor.

We picked GAT over GCN because GCN did not pass our preliminary experiments and GAT did. We were unable to parameterize a GCN that could learn a simple function of node features in a Supervised Learning setting. We tried GCN in the following parameter space: $\#GConvlayers \in \{1, 2, 3\}$; $layersize \in \{16, 32, 64\}$; aggregation function $agg \in \{sum, mean, max\}$; learning rate $lr \in \{0.01, 0.001, 0.0001\}$; dropout values $drop \in \{0, 0.5\}$.

We hypothesize this was due to the labels for our supervised task depending solely on the nodes own features. GCN would construct node representations by indiscriminately aggregating their neighborhoods, confounding neighbor features with their own. By learning to weigh edges, GAT could selectively focus on the node's self-edge. For this reason, and given the importance of a node's own features to our task, we chose to use GAT in our GNN policy.

4.5.1 Modules

A brief explanation of the sub-modules of our Ranking Module.

Input. Input to the ranking module consists of an adjacency matrix A, the focused node id n_f, node features X (strategy $s_i \in 0, 1$, and normalized node degree), and an action mask that disables invalid actions. (the node itself or nodes that are already neighbors)

Hidden Module. The hidden module computes node hidden features $h_i = H(x_i)$. It can be a MLP or GAT, we study both.

Score Module. The score module is a single-layer MLP that takes the hidden features h_i of each node, concatenated with h_f, those of the focused node, and graph features h_g obtained by summing over all node features. $score_i = S(h_g, h_f, h_i), \qquad h_g = \sum_{i \in V} h_i$

4.6 Training

Table 4.2: Recommender reward functions being studied.

Short	Recommender Name	Description
#rewires	Engagement maximizer	Total amount of rewire requests made to the recommender in each episode - as opposed to total number of time-steps (strategy updates + rewire updates).
#coops	Aligned recommender	Final number of cooperators. A natural metric for social good in our toy environment.

At each step, the Ranking Module ranks nodes based on their own features, the focused node's features, and whole-graph features (obtained by aggregating all node features). The MLP hidden layer is unable to use neighborhood information and so it will rely solely on the relationship between features of nodes and graph features. The GAT hidden layer uses neighborhood information and learns to weigh the contributions of each neighbor to a node's hidden features.

Each line on the learning curve plots is the average of the 3 best runs for that configuration, while the clouds around them delimit maximum and minimum values. We train GAT and MLP policies to maximize each of the reward functions in table 4.2. In plots for #cooperators, the reward value corresponds to $rew = 2 * (coops - 0.5)$, such that convergence to 1.0 cooperators give 1 reward and to 0.0 cooperators gives -1 reward. One training step corresponds to one batch of environment time-steps, or rewire updates.

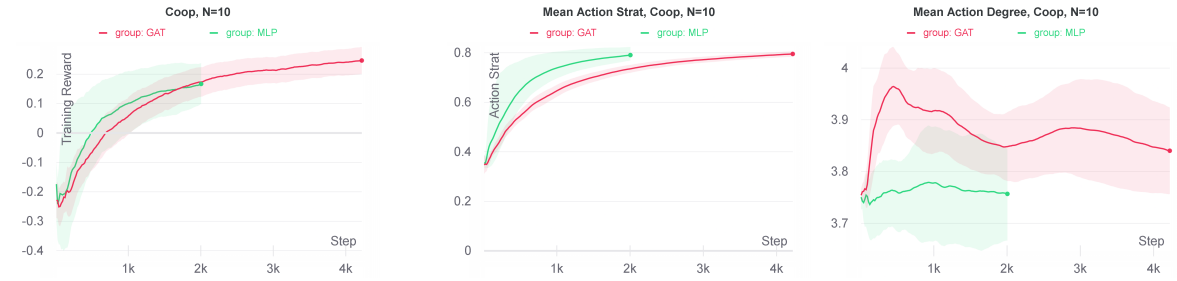
Mean Action Strat plots the mean strategy of recommended nodes over training (when $D = 0$ and $C = 1$), whereas *Mean Action Degree* tracks the mean degree of recommended nodes.

A difference in performance between training and evaluation may be observed and can be explained by the fact that in evaluation actions always results of picking the highest score, whereas during training, they're sampled from a distribution to ensure exploration.

Table 4.3: Baselines for N=10. Null policy (no rewires), Local heuristic, Strategy x Degree heuristic combinations. Values are averages over 1000 runs of: final number of cooperators, number of rewires per episode, final graph heterogeneity, time steps per episode, and mean action degree. Standard deviation in parenthesis. Max values for our training metrics #coops and #rewires are printed in bold for heuristics and learned policies. Null policy takes no actions, so average action strategy and action degree are null.

	Heuristic	Coops	Rewire	Action Strat	Action Degrees	Het	Time
	Null	0.080±0.328	**47.5±34.1**	-	-	0.00	98.80
	Local	0.260±0.479	39.9±25.1	0.35	4.32	3.43	109.00
Defectors							
	Low	0.087±0.328	47.4±33.9	0.00	3.34	0.33	97.10
	Random	0.108±0.303	45.9±42.6	0.00	3.78	1.71	96.33
	High	0.121±0.303	40.0±24.4	0.00	5.07	4.38	88.80
Random							
	Low	0.171±0.404	42.9±28.5	0.16	3.20	0.34	97.90
	Random	0.331±0.501	36.4±24.7	0.22	3.72	2.19	102.94
	High	0.283±0.431	32.6±18.9	0.26	5.21	4.75	90.30
Cooperators							
	Low	**0.617±0.467**	26.7±16.6	0.16	3.97	2.75	118.12
	Random	0.611±0.501	26.4±28.2	0.17	4.40	3.23	111.55
	High	0.530±0.482	29.1±18.5	0.16	4.52	4.04	110.81
Learned							
	MLPCoop	0.617±0.431	28.2±20.4	0.84	3.76	2.72	112.01
	MLPRewire	0.092±0.274	46.5±25.5	0.00	3.36	1.34	97.70
	GATCoop	**0.634±0.453**	26.9±30.3	0.84	3.69	2.59	116.23
	GATRewire	0.151±0.370	**50.4±28.9**	0.09	3.95	1.78	108.35
	MLPCoopN30	**0.637±0.503**	27.4±21.1	0.85	3.54	2.24	114.13

4.6.1 N=10

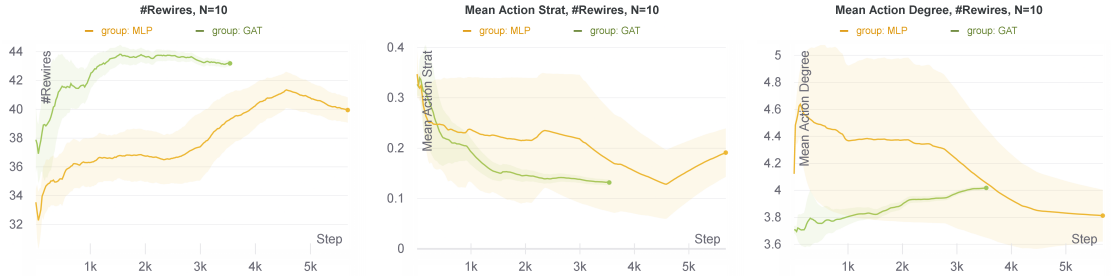


(a) #coops. **(b)** Strategy. **(c)** Degree.

Figure 4.6: Learning curves for **#cooperators** at $N = 10$. We can see both policies learn and GAT surpasses MLP. Both policies also beat heuristic baselines. We can see GAT consistently recommends slightly higher degree nodes and eventually reaches MLP's average action strategy.

The environment converges quickly when $N = 10$. We are able to learn policies that do better than any heuristic for either reward function. The modest improvements in #coops may be due to the size and convergence speed of the environment leaving little room to act strategically.

GAT policy performs slightly better than MLP on #coop but considerably better on #rewires. The #coop policy trained for $N = 30$ actually performed better on this environment than any of the $N = 10$ policies we trained. We observe the behavior of learned policies usually resemble the best heuristics in



(a) #rewires. **(b)** Strategy. **(c)** Degree.

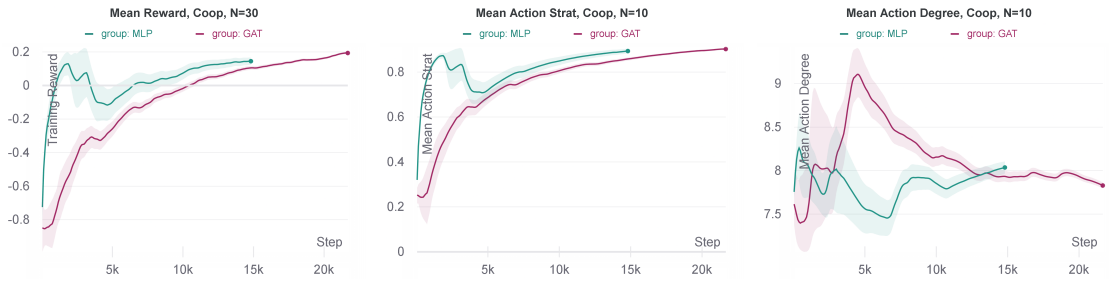
Figure 4.7: Learning curves for **#rewires** at $N = 10$. We can see both policies learn and GAT is consistently better than MLP. In this case, only GAT beats all heuristic baselines. In this case #rewires seems to correlate with lower average action strategy (rapid early increase in score) and lower action degree (decrease in score coinciding with increasing action degree).

most metrics. However, there are noticeable differences in action strategy and action degrees that might account for learned improved behavior.

Table 4.4: Baselines for N=30. Null policy (no rewires), Local heuristic, Strategy x Degree heuristic combinations. Values are averages over 1000 runs of: final number of cooperators, number of rewires per episode, final graph heterogeneity, time steps per episode, and mean action degree. Standard deviation in parenthesis. Max values for our training metrics #coops and #rewires are printed in bold for heuristics and learned policies.

	Heuristic	Coops	Rewire	Action Strat	Action Degrees	Het	Time
	Null	0.0±0.0	304±151	0.00	8.00	0.00	473.83
	Local	0.045±0.328	290±130	0.24	12.74	46.23	574.09
Defectors							
	Low	0.0±0.0	301±197	0.00	7.08	0.15	466.08
	Random	0.0±0.0	300±206	0.00	7.77	6.79	468.61
	High	0.0±0.0	264±100	0.00	14.18	49.90	444.72
Random							
	Low	0.004±0.000	331±148	0.17	7.04	0.14	533.78
	Random	0.078±0.239	361±222	0.21	7.67	8.56	690.10
	High	0.066±0.274	270±118	0.24	14.40	49.88	519.28
Cooperators							
	Low	**0.715±0.393**	484±324	0.98	7.18	8.24	1711.78
	Random	0.593±0.448	**530±343**	0.96	8.80	20.80	1657.60
	High	0.505±0.474	502±373	0.89	11.61	36.95	1369.69
Learned							
	MLPCoop	**0.739±0.434**	482±325	0.96	7.12	8.04	2075.43
	MLPRewire	0.506±0.487	**577±368**	0.90	11.92	7.50	1510.09
	GATCoop	0.652±0.402	484±315	0.97	7.48	11.21	1575.07
	GATRewire	0.0±0.0	329±164	0.00	8.84	16.41	505.04

4.6.2 N=30



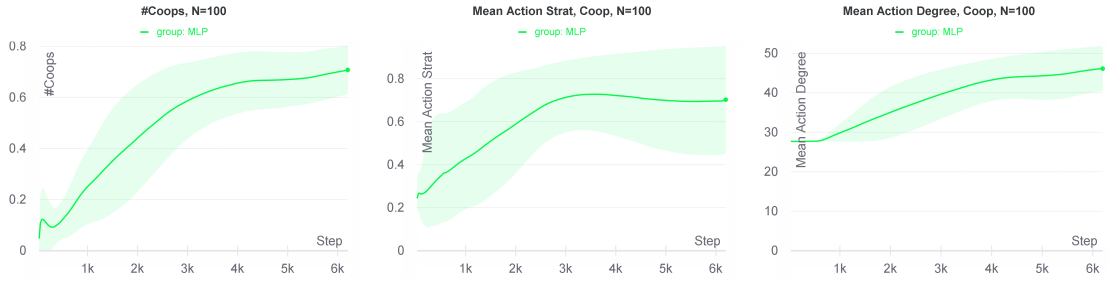
(a) #coops. **(b)** Strategy. **(c)** Degree.

Figure 4.8: Learning curves for **#cooperators** at $N = 30$. Similarly to N=10, we can see both policies learn and GAT slightly surpass MLP. Both policies also beat heuristic baselines. Despite noticeable variance in mean action degree, mean action strategy seems to be the determining factor in this case.

At $N = 30$, MLP policies significantly outperformed GAT ones both for #coop and #rewires. GAT's higher complexity might require more training episodes than MLP, which could explain its lower performance, as both policies were trained using the same resources.

Table 4.5: Baselines for N=100. Null policy (no rewires), Local heuristic, Strategy x Degree heuristic combinations. Values are averages over 1000 runs of: final number of cooperators, number of rewires per episode, final graph heterogeneity, time steps per episode, and mean action degree. Standard deviation in parenthesis. Max values for our training metrics #coops and #rewires are printed in bold for heuristics and learned policies.

	Heuristic	Coops	Rewire	Action Strat	Action Degrees	Het	Time
	Null	0.003±0.016	3262±1370	0.00	28.00	0.00	4731
	Local	0.544±0.369	3922±977	0.36	50.01	626.50	11034
Defectors							
	Low	0.000±0.000	3293±1410	0.00	27.02	0.06	4776
	Random	0.000±0.000	3353±1336	0.00	27.75	24.81	4910
	High	0.466±0.422	3167±1324	0.00	53.38	698.15	7354
Random							
	Low	0.000±0.000	3242±1292	0.19	27.01	0.06	4746
	Random	0.121±0.142	**4118±1561**	0.24	27.62	35.65	7333
	High	0.631±0.415	2773±1295	0.24	54.97	708.23	8139
Cooperators							
	Low	0.649±0.241	2967±1016	1.00	26.02	7.76	9563
	Random	0.671±0.220	2856±927	1.00	28.68	90.12	9513
	High	**0.891±0.179**	1863±760	0.99	51.60	570.02	7979
Learned							
	MLP Coop	0.887±0.257	1971±1215	0.89	52.70	611.60	8397
	MLP Rewire	0.006±0.061	3331±1267	0.01	27.52	16.96	5048
	MLP Rewire (N=30)	**0.890±0.156**	1859±756	0.98	51.85	557.86	8082
	GAT Rewire (N=10)	0.034±0.265	**5050±1025**	0.25	28.15	52.21	8916



(a) #coops. **(b)** Strategy. **(c)** Degree.

Figure 4.9: Learning curves for **#cooperators** at $N = 100$. We weren't able to train a GAT policy in this environment. GAT is computationally more expensive to train than MLP, and at this size, computational constraints start limiting our architecture.

Table 4.6: Baselines for N=500. Null policy (no rewires), Local heuristic, Strategy x Degree heuristic combinations. Values are averages over 1000 runs of: final number of cooperators, number of rewires per episode, final graph heterogeneity, time steps per episode, and mean action degree. Standard deviation in parenthesis. Max values for our training metrics #coops and #rewires are printed in bold for heuristics and learned policies.

	Heuristic	Coops	Rewire	Action Strat	Action Degrees	Het	Time
	Null	0.016±0.023	21665±2752	-	-	0.00	28113
	Local	0.596±0.241	**22505±2755**	0.36	148.40	4870.98	76570
Defectors							
	Low	0.014±0.016	21624±2570	0.00	29.00	0.01	28018
	Random	0.018±0.025	21502±2800	0.00	29.92	41.82	28176
	High	0.388±0.287	13519±3120	0.00	187.84	5353.50	29238
Random							
	Low	0.015±0.019	21612±3040	0.12	29.00	0.01	28153
	Random	0.137±0.084	18475±2460	0.13	29.92	52.63	29968
	High	0.440±0.300	11805±2640	0.27	195.90	5690.20	29550
Cooperators							
	Low	0.499±0.086	10936±1410	1.00	28.00	8.94	30000
	Random	0.518±0.087	10601±1220	1.00	31.42	103.34	30000
	High	**0.647±0.219**	8552±1540	0.93	189.55	4792.59	29557
Learned							
	MLPCoopN100	**0.682±0.089**	7901±1240	0.74	198.36	5063	29128

4.6.3 N=100

At $N = 100$, MLP #coop matches the top heuristic's performance for #coops. Our MLP trained to maximize #rewires on N=100 does not beat to top baselines, but the GAT policy trained on $N = 10$ is able to beat all heuristics by a large margin.

We have not been able to make GAT policies converge on environments where $N = 100$ or $N = 500$. Additionally, #coop seems to converge much more smoothly than #rewires at every value of N. This may be explained by the fact that there is a mapping between observations and #coops (node strategy features), whereas #rewires is merely counted and given as reward at the end of an episode, without markers in the observable state. We haven't tested this hypothesis.

4.6.4 N=500

At $N = 500$, despite having trained policies specifically on $N = 500$, the best performing policy is the one trained for MLP #Coop $N = 100$ and it beats all heuristics. Again, we hypothesize that policies in smaller environments converge faster, and that difference makes up for differences of dynamics in networks of $N = 100$ and $N = 500$.

To conclude this section, we show that **it is possible to learn to control evolutionary network dynamics by means of recommendations**. It is even possible to learn policies that outperform all heuristic combinations of strategy and node degree for some configurations of our environment. We expect that the configurations where we were unable to learn policies might be solvable with more fine-tuning, or simply applying more resources.

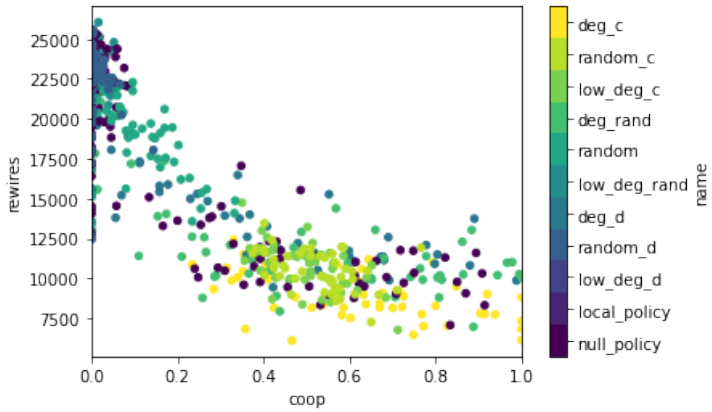


Figure 4.10: Scatter plot of runs for different recommender heuristics. The axes are **#coops** and **#rewires** for $N = 500$. We can observe a sort of convex Pareto frontier and a rough ordering among heuristics, with policies that recommend cooperators leading to higher #coop and lower #rewires, and the opposite for policies that recommend defectors.

4.7 Analysis

We began this toy experiment by asking "*How can we evaluate the consequences of using user retention as a reward function?*". More generally, to explore how we could evaluate and compare reward functions with respect to the societal effects caused by recommenders trained to optimize them. So we defined a toy environment to represent society and used reinforcement learning to optimize these reward functions.

Having evaluated our learned policies as well as many heuristics, we can ask the question in the context of our toy model: Do engagement maximizers perform significantly worse than aligned recommenders? The answer is, perhaps unsurprisingly, yes. In all our settings, engagement optimizers will lead the environment to low final numbers of cooperators. The only exception was the environment where $N = 30$, where the best performing engagement policy produces a final fraction of cooperators around 0.5 which is high in comparison with other heuristics.

This suggests something we already expected. With respect to our toy environment, misaligned recommenders like the engagement maximizer may or may not lead to bad outcomes for society, but they are worse than aligned recommenders. In the case of $N = 500$ fig. 4.10, these two reward functions are at odds with one another, at least for our heuristics which trace a convex curve in the axes of #coop and #rewires.

Strategies that recommend cooperators generally lead to higher convergence to full cooperation than the others, we don't observe such a pattern in policies that maximize #rewires.

With regards to topology, it is very clear that heterogeneity increases with mean action degree and that it is generally lower for heuristics that recommend cooperators than for the others.

Aligned policies also tend to lead to more heterogeneous networks than misaligned ones, again with the exception of $N = 30$.

One limitation in these experiments is that we are considering worlds where there is only one rewiring policy and simulating what happens if people keep using it. In the real world, if a recommender is bad enough, people will just not use it at all.

A more realistic setup would look like competition between the recommender and the null or local heuristics. In that case, there would be a bound on how bad the RS could be before users decide they might as well not rewire social ties at all, or only within their local environment, respectively.

If learning happens in these conditions, we might see a ”bait-and switch” strategy emerge. The recommender might begin by being advantageous to users while trying to get adopted by as many users as possible, and switching to an ”exploit” dynamic once dominant in the population, to further maximize its reward function.

5

Experiment: Competition

Contents

In this section, we extend our toy environment with competition between recommenders and present baselines for its the outcomes. Then we test adoption dynamics of engagement maximizers and aligned recommenders starting in majority local-heuristic environments. We also test whether aligned recommenders would be adopted in majority engagement maximizers. We conclude that competition generally leads to better outcomes for society than single recommender monopolies of engagement maximizers.

5.1 Competition Dynamics

We extend our environment to allow competition between recommenders. This is achieved by attributing a recommender strategy to each node and allowing them to evolve in the same way that game strategies evolve. That is:

Recommender Update At each time-step, there is a chance that the update performed is a recommender update. That is, the user may change which recommender she is using to rewire ties. In that case, an agent selects a random neighbor and imitates its rewiring strategy with probability p weighed on their fitness difference given by the Fermi update. (eq. (3.1))

Recommenders are exclusive, meaning they will only recommend nodes from among their own users. Keep in mind users still interact with their neighbors, even if they are using other recommenders. This reflects what we see in the world, where recommender systems (mostly) only have information about their own users.

We introduce a second time-scale ratio $W2 = t_m/(t_e + t_a)$ to regulate the relative frequencies of mediator updates (t_m) and two other kinds of updates. (strategy t_a or structural t_e)

All competition runs are initialized with 1000 nodes and a time-limit of 10^5 time-steps. The effects of competition were not as clear on populations of 500. Recommender updates use a different temperature parameter $beta_{med} = beta * 10 = 0.05$ because we observed the impact of mediator updates was negligible using 0.005. Given the stochastic nature of our simulations, all presented results are averaged over 30 runs.

Table 5.1: Baseline metrics for N=1000, used in competition. Null policy (no rewires), Local heuristic, Strategy x Degree heuristic combinations. Values are averages over 100 runs of: final number of cooperators, number of rewires per episode, final graph heterogeneity, time steps per episode, and mean action degree. Standard deviation in parenthesis. Max values for our training metrics #coops and #rewires are printed in bold for heuristics and learned policies.

	Heuristic	Coops	Rewire	Action Strat	Action Degrees	Het	Time
	Null	0.00	62763.36	0.00	30.00	0.00	76311.20
	Local	0.00	33934.28	0.24	521.95	10911.87	53428.30
Defectors							
	Low	0.00	63020.86	0.00	29.00	0.00	76539.72
	Random	0.00	62588.68	0.00	29.95	49.04	76638.92
	Deg	0.00	28041.30	0.00	405.41	11328.91	39968.52
Random							
	Low	0.00	60510.20	0.10	30.00	0.00	73182.14
	Random	0.06	**71487.22**	0.17	29.95	64.52	99819.14
	Deg	0.00	28779.84	0.20	413.62	11747.65	42195.98
Cooperators							
	Low	0.51	35770.52	1.00	27.89	7.90	100000.00
	Random	**0.52**	34542.48	1.00	31.42	144.64	100000.00
	Deg	0.06	57982.24	0.70	519.83	8702.59	100000.00
Learned							
	MLP Coop N=10	0.50	36236.70	1.00	27.89	8.24	100000.00
	MLP Rewire N=10	0.00	27284.10	0.09	373.61	10175.79	39287.24

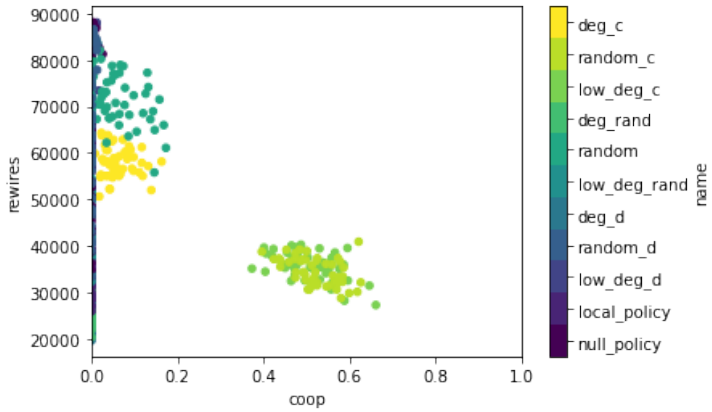
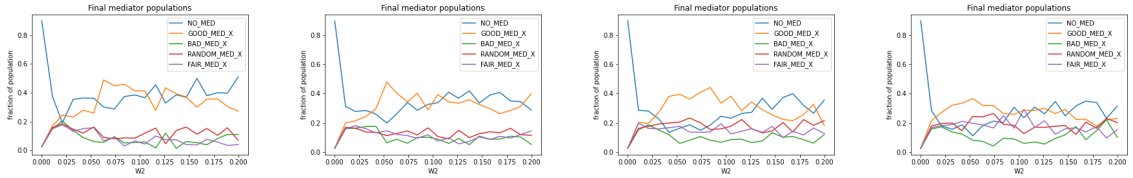


Figure 5.1: Scatter plot where the axes are #coops and #rewires for N=1000. We can observe a sort of Pareto frontier and a rough ordering among heuristics, with policies that recommend cooperators leading to higher #coop and lower #rewires, and the opposite for policies that recommend defectors.

5.2 Competing Baselines

We begin by taking an environment dominated by the local heuristic and introducing recommenders into 10% of the population. 90% of nodes use the local heuristic NO MED, while each node of the remaining 10% has one of the others. (GOOD, BAD, RANDOM, FAIR)

We began with measuring the effects of competition over various timescale combinations W and W2. ($\{0.5, 1, 2, 3, \infty\}$ and $\{0.01, 0.03, 0.1, 0.5, \infty\}$ respectively) We observe (fig. A.1) that for $W1 = \inf$ (no strategy updates, only rewires and mediator updates), the initial conditions remain practically the same,



(a) W1=1. GOOD and NO MED dominate over the others with no clear winners between the two groups.

(b) W1=1.2. Same as W1=1 but less pronounced

(c) W1=1.6. Even less pronounced than W1=1.2 .

(d) W1=2. More difficult to discern than W1=1 but BAD is clearly on the bottom, while GOOD and NO MED seem to share the top but by a smaller margin.

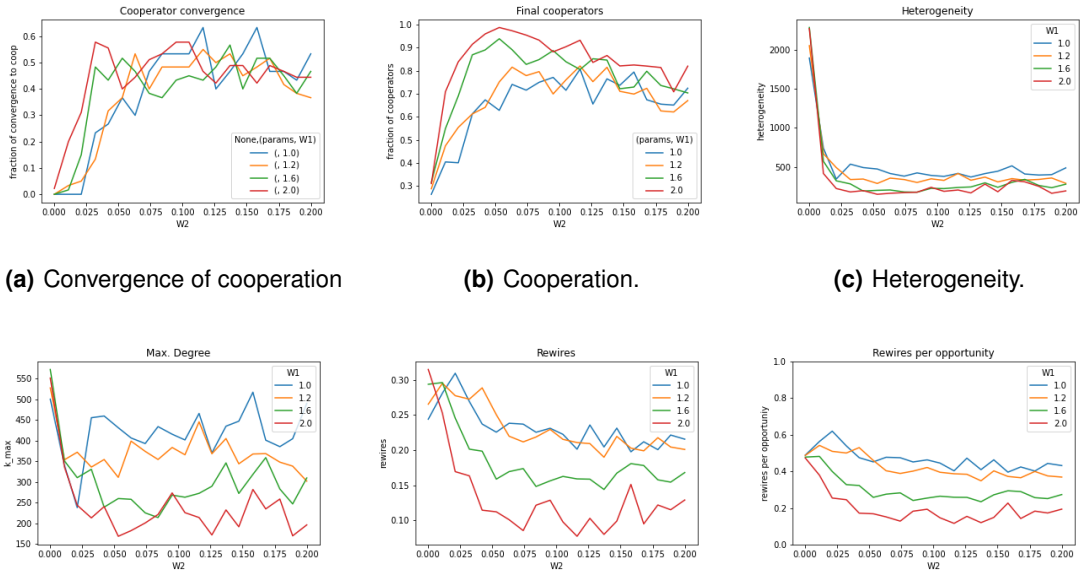
Figure 5.2: Final frequencies of mediators as functions of W2.

while for $W2 = 0$ the initial conditions remain the same due to there being no mediator updates.

We find a critical region where NO MED does not have a majority around $W1 \in [1, 2]$ and $W2 \in [0.03, 0.1]$. We investigate this range more closely in fig. 5.2 and observe that GOOD and NO MED have a tendency to dominate over the others. That is most pronounced at $W1 = 1$. As W1 increases, the gap between strategy populations becomes less evident, although BAD takes the bottom place more convincingly.

For the same interval of $W2$, we study metrics of cooperation, heterogeneity, and rewires (fig. 5.3) and find that higher values of W1 lead to higher cooperation overall (as expected) but decrease heterogeneity and rewire number. Comparing these with the metrics we obtained for the simulations with single mediators, we observe much higher heterogeneity (around 500 rather than 200) and k_max (around 200 to 450 rather than 60 to 80) in the competition scenario.

Our results are not crystal clear thanks to the chaotic nature of the system. However, they do convincingly indicate that for certain timescales, an exclusive policy that recommends cooperators will dominate over other exclusive recommenders. It does not, however, completely subsume the local rewiring heuristic, which remains on par with it for W=1 and W=1.2, while GOOD even strictly dominates for W=1.6 in the short interval of $W_2 \in [0.025, 0.125]$.



(a) Convergence of cooperation **(b)** Cooperation. **(c)** Heterogeneity.

(d) Max degree. **(e)** Rewires. **(f)** Rewires per opportunity.

Figure 5.3: Metrics for the environment resulting of competition, as a function of W2 for different values of W1.

5.3 Adoption Experiments

Questions for adoption experiments:

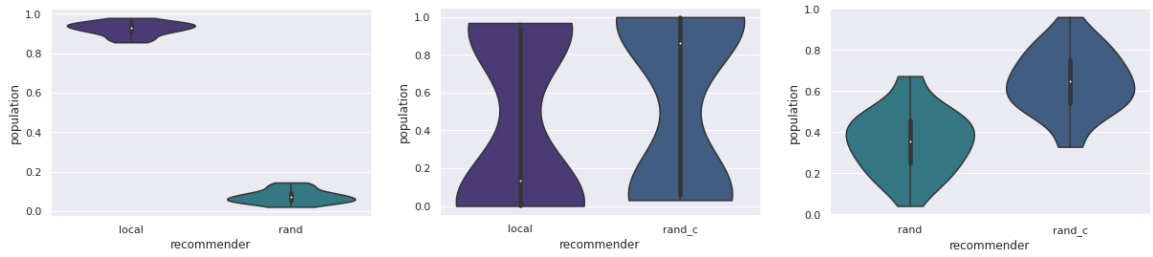
1. Will engagement maximizers be adopted and thrive in a society of local heuristics? What about aligned recommenders?

2. In a society dominated by engagement maximizers, will an aligned recommender be adopted and thrive?

3. Do worlds were RS competition exists lead to better outcomes for society than worlds dominated by engagement maximizers?

From our experiments, competition results are clearest in environments where N=1000. Due to computational and time constraints, we were unable to learn policies directly on these large environments. Despite this, we pick the heuristics that score highest in our metrics of #coops and #rewires to use as proxies for policies trained explicitly to maximize them. We pick ”Random cooperators” as our aligned recommender and ”Random” as our engament maximizer.

We run one trial for each recommender in environments with local-heuristic majorities (90% local). We want to know whether recommenders maximizing user-engagement would be able to take over a population starting from a small seed. We ask the same question for recommenders maximizing cooperation.



(a) Initial conditions: Majority local heuristic, minority engagement maximizer.

(b) Initial conditions: Majority local heuristic, minority aligned recommender.

(c) Initial conditions: Majority engagement maximizer, minority aligned recommender.

Figure 5.4: Distribution of final populations in competition. Engagement maximizer doesn't manage to get adopted, while aligned recommender gets adopted slightly above 50% of the population. Aligned recommender is able to dominate in an environment initially dominated by the engagement maximizers.

We can observe that a diverse competition scenario (Local vs Many) leads to high cooperation and wide adoption of recommenders in our toy environment. This is consistent with the suggestion that competition between RS is a desirable feature to implement in the world, protecting it from misaligned recommenders.

Table 5.2: Metrics resulting from competition between optimizer policies and local heuristic. Averages over 30 runs. Both are adopted over the local heuristic. "Initial Majority" stands for the average final proportion of the policies that began as the majority (local, local, and engagement respectively in the competition scenarios below).

Heuristic	Coops	Rewire	Initial Majority	Action Strat	Action Degrees	Het	Time
Engagement	0.06	71487.22	-	0.17	29.95	64.52	99819.14
Engagement vs Local majority	0.00	39301.03	0.93	0.22	451.46	9738.76	59731.13
Local	0.00	33934.28	-	0.24	521.95	10911.87	53428.30
Local vs Many	0.43	**277427.63**	0.23	0.51	205.33	3199.54	684760.00
Aligned vs Local majority	0.36	225244.53	0.43	0.49	358.29	5391.85	544231.13
Aligned vs Engagement majority	0.47	417636.18	0.44	0.77	44.15	498.27	989539.74
Aligned	**0.52**	34542.48	-	1.00	31.42	144.64	100000.00

6

Conclusion

Contents

This dissertation has a broad scope, touching superficially on many different points to shed some light on how an end-to-end study of reward functions for recommender systems might be done. Recommender alignment is a pressing and important problem. Attempted solutions are sure to have far-reaching impacts. The least we could do is develop methods for evaluating and comparing tentative solutions with respect to those impacts. We synthesized a simple abstract modelling framework to guide future work.

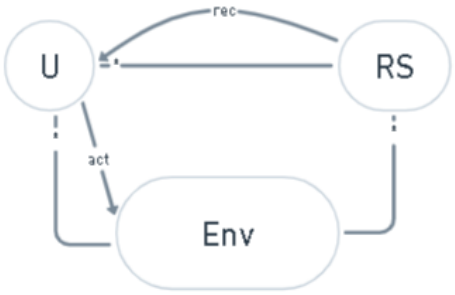


Figure 6.1: Miniature UML diagram of our modelling interface.

Namely, a model must include an underlying environment, users with partial information, and RS with global information; notions of utility (derived from the environment) for individual users, society, and the RS; and a notion of recommendation.

A toy experiment was run to show our framework in action.

1. We wanted to know whether user retention is a good idea as a proxy for social good. We adapted an existing model to implement our modelling interface and ran several simulations to see how it behaves under different heuristic rewiring strategies.

An ordering of heuristics is found with respect to speed of convergence to full cooperation. Surprisingly, recommending cooperators to cooperators and defectors to defectors leads to worse outcomes than a random policy.

2. Then we used proximal policy optimization and graph neural networks to learn to control graph dynamics through recommendations. We did this just to obtain policies explicitly optimized for the reward functions we wanted to compare.

We verify it is possible to learn to control graph dynamics using recommendations. Analyzing environment metrics, we conclude that engagement maximizers generally lead to worse outcomes than aligned recommenders but not always.

3. After that we extended our toy environment to allow for recommender competition within the same population and compared simulations of it under competition with simulations under a single recommender.

We find that recommender competition generally makes our society better-off. Aligned recommenders are found to be able to replace both local heuristics and engagement maximizers. The en-

gagement maximizers we tested are not found to be able to replace local heuristics, suggesting real world RS would've needed different strategies to be adopted.

Our main focus wasn't the results of these experiments, which are simple and self-contained, but rather the fact that we were able to construct them thanks to our modelling interface, which allows for arbitrary complexity in the concrete model used. With the second interesting benefit of learning to control the evolutionary graph dynamics of our toy environment.

We hope this has laid some foundations for future work on proposals for RS alignment and their evaluation. The experience of billions of people is shaped daily by content recommendation. As society changes, so must our objective functions, therefore the need for a bottom-up adaptive system that answers to users. Aligning recommender systems is one of the critical tasks of our time.

6.1 Future Work

The direction we're most excited about working on is in learning recommenders under competition. Especially the base case where recommenders merely need to compete with the null or the local strategies. In this case, recommenders don't learn under the assumption that they have a monopoly on their populations. Therefore, even if misaligned, they must learn to not be actively detrimental, and a little better than the baseline in order to get adopted. We expect we'll be able to find an emergent dynamic here that is also present in the real world, where after widespread adoption, when it is more disadvantageous for users to leave because of network effects, recommenders turn to "exploit mode", caring less about user utility and more about their own.

If we were able to spend more time on learning graph dynamics, it would be edifying to apply methods of interpretability to understand what strategies the agents have learned to beat to baselines. Do they focus on converting hubs? Do they target high-betweenness nodes somehow? Are focused nodes served differently based on their strategy?

Another avenue of study would be games beyond the *prisoner's dilemma*, like *ultimatum*, or public goods, extending user interaction from 1-to-1 to many-to-many. Completely different environments like *Sugarscape* [Epstein and Axtell, 1996] or sequential social dilemmas [Leibo et al., 2017] are also worth exploring.

Finally, midway through the work of this dissertation, temporal GNN libraries like Pytorch Geometric [**?**] became available, which would certainly make learning more efficient enabling training for longer periods and perhaps allow convergence in some of our larger environments.

Bibliography

[Anastassacos et al., 2020] Anastassacos, N., Hailes, S., and Musolesi, M. (2020). Partner Selection for the Emergence of Cooperation in Multi-Agent Systems Using Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:7047–7054.

[Andreassen, 2015a] Andreassen, C. S. (2015a). Online Social Network Site Addiction: A Comprehensive Review. *Current Addiction Reports*, 2(2):175–184.

[Andreassen, 2015b] Andreassen, C. S. (2015b). Online Social Network Site Addiction: A Comprehensive Review. *Current Addiction Reports*, 2(2):175–184.

[Babaioff et al., 2015] Babaioff, M., Feldman, M., and Tennenholtz, M. (2015). Mechanism Design with Strategic Mediators. *ACM Transactions on Economics and Computation*, 4.

[Barocas et al., 2019] Barocas, S., Hardt, M., and Narayanan, A. (2019). *Fairness and Machine Learning*. fairmlbook.org. http://www.fairmlbook.org.

[Benkler et al., 2018] Benkler, Y., Faris, R., and Roberts, H. (2018). *Network Propaganda: Manipulation, Disinformation, and Radicalization in American Politics*. Oxford University Press, New York.

[Burr et al., 2018a] Burr, C., Cristianini, N., and Ladyman, J. (2018a). An Analysis of the Interaction Between Intelligent Software Agents and Human Users. *Minds and Machines*, 28(4):735–774.

[Burr et al., 2018b] Burr, C., Cristianini, N., and Ladyman, J. (2018b). An Analysis of the Interaction Between Intelligent Software Agents and Human Users. *Minds and Machines*, 28(4):735–774.

[Calero Valdez and Ziefle, 2018] Calero Valdez, A. and Ziefle, M. (2018). Human Factors in the Age of Algorithms. Understanding the Human-in-the-loop Using Agent-Based Modeling. In Meiselwitz, G., editor, *Social Computing and Social Media. Technologies and Analytics*, Lecture Notes in Computer Science, pages 357–371, Cham. Springer International Publishing.

[Castells et al., 2015] Castells, P., Hurley, N. J., and Vargas, S. (2015). Novelty and Diversity in Recommender Systems. In Ricci, F., Rokach, L., and Shapira, B., editors, *Recommender Systems Handbook*, pages 881–918. Springer US, Boston, MA.

[Chen et al., 2020] Chen, M., Beutel, A., Covington, P., Jain, S., Belletti, F., and Chi, E. (2020). Top-K Off-Policy Correction for a REINFORCE Recommender System. *arXiv:1812.02353 [cs, stat]*. arXiv: 1812.02353.

[Darvariu et al., 2020] Darvariu, V.-A., Hailes, S., and Musolesi, M. (2020). Improving the Robustness of Graphs through Reinforcement Learning and Graph Neural Networks. *arXiv:2001.11279 [cs, stat]*. arXiv: 2001.11279.

[Dulac-Arnold et al., 2016] Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. (2016). Deep Reinforcement Learning in Large Discrete Action Spaces. *arXiv:1512.07679 [cs, stat]*. arXiv: 1512.07679.

[Epstein and Axtell, 1996] Epstein, J. M. and Axtell, R. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. Brookings Institution Press. Google-Books-ID: xXvelSs2caQC.

[Floridi, 2011] Floridi, L. (2011). The Construction of Personal Identities Online. *Minds and Machines*, 21(4):477–479.

[Goodhart, 1981] Goodhart, C. (1981). *Problems of Monetary Management: The UK Experience*.

[Hadfield-Menell and Hadfield, 2019] Hadfield-Menell, D. and Hadfield, G. K. (2019). Incomplete Contracting and AI Alignment. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, pages 417–422, New York, NY, USA. Association for Computing Machinery.

[Hasan et al., 2018] Hasan, R., Jha, A., and Liu, Y. (2018). Excessive Use of Online Video Streaming Services: Impact of Recommender System Use, Psychological Factors, and Motives. *Computers in Human Behavior*, 80.

[Hohnhold et al., 2015] Hohnhold, H., O'Brien, D., and Tang, D. (2015). Focusing on the Long-term: It's Good for Users and Business. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1849–1858, Sydney, NSW, Australia. Association for Computing Machinery.

[Ibarz et al., 2018] Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. (2018). Reward learning from human preferences and demonstrations in Atari. *arXiv:1811.06521 [cs, stat]*. arXiv: 1811.06521.

[Ie et al., 2019] Ie, E., Jain, V., Wang, J., Narvekar, S., Agarwal, R., Wu, R., Cheng, H.-T., Lustman, M., Gatto, V., Covington, P., McFadden, J., Chandra, T., and Boutilier, C. (2019). Reinforcement Learning for Slate-based Recommender Systems: A Tractable Decomposition and Practical Methodology. *arXiv:1905.12767 [cs, stat]*.

[Izsak et al., 2014] Izsak, P., Raiber, F., Kurland, O., and Tennenholtz, M. (2014). The search duel: a response to a strong ranker. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, SIGIR '14, pages 919–922, Gold Coast, Queensland, Australia. Association for Computing Machinery.

[Khwaja et al., 2019] Khwaja, M., Ferrer, M., Iglesias, J. O., Faisal, A. A., and Matic, A. (2019). Aligning Daily Activities with Personality: Towards A Recommender System for Improving Wellbeing. *arXiv:1909.03847 [cs]*. arXiv: 1909.03847.

[Kipf and Welling, 2017] Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*. arXiv: 1609.02907.

[Knijnenburg et al., 2016] Knijnenburg, B. P., Sivakumar, S., and Wilkinson, D. (2016). Recommender Systems for Self-Actualization. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 11–14, New York, NY, USA. Association for Computing Machinery.

[Kossinets and Watts, 2006] Kossinets, G. and Watts, D. J. (2006). Empirical Analysis of an Evolving Social Network. *Science*, 311(5757):88–90. Publisher: American Association for the Advancement of Science Section: Report.

[Kurland, 2019] Kurland, Oren, M. T. (2019). Rethinking Search Engines and Recommendation Systems: A Game Theoretic Perspective.

[Leibo et al., 2017] Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. (2017). Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 464–473, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Liu et al., 2019] Liu, F., Tang, R., Li, X., Zhang, W., Ye, Y., Chen, H., Guo, H., and Zhang, Y. (2019). Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling. *arXiv:1810.12027 [cs]*. arXiv: 1810.12027.

[Meirom et al., 2021] Meirom, E. A., Maron, H., Mannor, S., and Chechik, G. (2021). Controlling Graph Dynamics with Reinforcement Learning and Graph Neural Networks. *arXiv:2010.05313 [cs]*. arXiv: 2010.05313.

[Milano et al., 2020] Milano, S., Taddeo, M., and Floridi, L. (2020). Recommender systems and their ethical challenges. *AI & SOCIETY*.

[Ng and Russell, 2000] Ng, A. and Russell, S. (2000). Algorithms for Inverse Reinforcement Learning. *ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning*.

[Sanchez-Lengeling et al., 2021] Sanchez-Lengeling, B., Reif, E., Pearce, A., and Wiltschko, A. B. (2021). A Gentle Introduction to Graph Neural Networks. *Distill*, 6(9):e33.

[Santos et al., 2006a] Santos, F., Pacheco, J., and Lenaerts, T. (2006a). Evolutionary Dynamics of Social Dilemmas in Structured Heterogeneous Populations. *Proceedings of the National Academy of Sciences of the United States of America*, 103:3490–4.

[Santos et al., 2006b] Santos, F. C., Pacheco, J. M., and Lenaerts, T. (2006b). Cooperation Prevails When Individuals Adjust Their Social Ties. *PLoS Computational Biology*, 2(10).

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*. arXiv: 1707.06347.

[Seaver, 2019] Seaver, N. (2019). Captivating algorithms: Recommender systems as traps. *Journal of Material Culture*, 24(4):421–436.

[Stray et al., 2021] Stray, J., Vendrov, I., Nixon, J., Adler, S., and Hadfield-Menell, D. (2021). What are you optimizing for? Aligning Recommender Systems with Human Values. *arXiv:2107.10939 [cs]*. arXiv: 2107.10939.

[Stöcker, 2020] Stöcker, C. (2020). How Facebook and Google Accidentally Created a Perfect Ecosystem for Targeted Disinformation. pages 129–149.

[Traulsen et al., 2006] Traulsen, A., Nowak, M. A., and Pacheco, J. M. (2006). Stochastic Dynamics of Invasion and Fixation. *Physical Review E*, 74(1):011909.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *arXiv:1706.03762 [cs]*. arXiv: 1706.03762.

[Veličković et al., 2018] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph Attention Networks. *arXiv:1710.10903 [cs, stat]*. arXiv: 1710.10903.

[Vendrov and Nixon, 2019] Vendrov, I. and Nixon, J. (2019). Aligning Recommender Systems as Cause Area. Publication Title: EA Forum.

[Yang et al., 2013] Yang, Z., Li, Z., Wu, T., and Wang, L. (2013). Role of recommendation in spatial public goods games. *Physica A: Statistical Mechanics and its Applications*, 392(9):2038–2045.

[Zafari et al., 2018] Zafari, F., Moser, I., and Baarslag, T. (2018). Modelling and Analysis of Temporal Preference Drifts Using A Component-Based Factorised Latent Approach.

Supplementary Material

A.1 Training Architecture

Table A.1: Learning parameters.

Parameter	Value
learning rate	$1 * 10^{-3}$
λ	0.97
γ	0.99
Entropy Loss weight	0.01
Value Loss Weight	0.5
Batch Size	200
Hidden-layer size	32
#attention heads	3

We used the following Python libraries for the chapter on learning recommenders: Pytorch for basic deep learning, Deep Graph Library (DGL) for graph neural modules, and RLlib as our RL framework. At the time of the experiments, there were no high-quality frameworks for RL on dynamic graphs, therefore we resorted to using DGL which is suited for static graphs, and initializing a new DGL graph data structure at every training step, which is considerably slower than using a dedicated dynamic graph data structure.

A.2 Training Process

Since the task of learning (using RL) to control evolutionary dynamic processes via recommendation is a novel one, we took several intermediate steps before arriving at training GNNs on a fully dynamic graph. We began by defining a fixed graph without rewiring where the only dynamics were the evolution of node strategies, actions weren't rewiring recommendations but simply flipping node strategies (from C to D, or D to C).

We attempted naive Q-Learning on very small graphs and quickly hit the computational barrier of a combinatorial action space. Afterwards we switched to PPO since policy gradients don't need to keep track of the whole state space. After succeeding in learning to control dynamics in the simplified action and environment, we added rewiring, but retained the simplified flipping action. After succeeding in that setting, we move on to recommendation as the action.

We had some trouble training policies for the larger values of N, especially 500. Computational costs are at least partly to blame, as episodes take several hundred times longer to complete. We introduced intermediate rewards to mitigate this, rewarding the RS for reaching certain thresholds of cooperation or rewiring for the first time in each episode.

We were still having trouble converging after adding intermediate rewards. Originally, sum aggregation of node-representations was being used to produce graph-representations within our ranking

module. For larger Ns, the norm of these graph representations becomes hundreds of times larger than node representations, making it harder for the model to converge. We changed the aggregation function from sum to mean and afterwards we were able to get MLP coop to converge for N=100, but no others. Policies trained on smaller environments may generalize to larger ones, but what if the emergent dynamics at scale are radically different? This leaves an open question of how to learn the characteristic dynamics of larger networks.

A.3 Competition

Algorithm A.1: Simulation algorithm for mediator competition

begin
- **while** $t < timeLimit$ **do**
  - $x \longleftarrow randomSample(V)$
  - $y \longleftarrow randomNeighbor(x)$
  - $P_x, P_y \longleftarrow cumulativePayoff(x), cumulativePayoff(y)$
  - $p \longleftarrow fermi(P_x - P_y, beta)$
  - **if** $random(0,1) < (1+W2)^{-1}$ **then**
    - **if** $random(0,1) < p$ **then**
      - $rewireStrat_x \longleftarrow rewireStrat_y$
  - **else**
    - **if** $random(0,1) < (1+W)^{-1}$ **then**
      - **if** $random(0,1) < p$ **then**
        - $Strat_x \longleftarrow Strat_y$
    - **else**
      - **if** $Strat_y == D \, and \, Strat_x == C$ **then**
        - **if** $random(0,1) < p$ **then**
          - $z \longleftarrow rewireStrat_x(x,y)$
          - $doRewire(G,x,y,z)$
      - **if** $Strat_y == D \, and \, Strat_x == D$ **then**
        - **if** $random(0,1) < p$ **then**
          - $z \longleftarrow rewireStrat_x(x,y)$
          - $doRewire(G,x,y,z)$
        - **else**
          - $z \longleftarrow rewireStrat_y(y,x)$
          - $doRewire(G,y,x,z)$

Figure A.1: Average final fraction of mediator populations after competition between NO MED and exclusive fixed mediators. For $W1 \in \{0.5, 1, 2, 3, \inf\}$ and $W2 \in \{0.01, 0.03, 0.1, 0.5, \inf\}$. For higher values of W2, we see a relatively low adoption rate and uniform distribution of mediators besides NO MED. Initialized at 90% NO MED and 10% split between the rest.