Multi-Armed Bandit Networks

Exploring Online Learning with Networks

June 1, 2018

Author:
Viktor Hansen

Supervisor:
Richard E. Moe

University of Bergen

A thesis submitted in fulfillment of the requirements for the degree of
Master of Information Science
at the
Department of Information Science and Media Studies

Abstract

Classical Multi-Armed Bandit solutions often assumes independent arms as a simplification of the problem. This has shown great results in many different fields of practice, but could in some cases, presumably leave untapped potential. In this paper I explore network based MAB solutions using explore-exploit algorithms as nodes to further minimize regret, and take advantage of inter-Bandit dependencies. I explore two network approaches; Hierarchical and Flat network. As well as a special cases of the Bernoulli Bandit with dependent arms, referred to as Symbiotic Bandit. The results show that some networked solutions prevail the single node versions in both the Bernoulli Bandit and the Symbiotic Bandit regret wise.

Contents

Abbreviations & Terms

- MAB - Multi-Armed Bandit
- EFE - Elusive Feature Exploitation
- TS - Thompson Sampling
- MCTS - Monte Carlo Tree Search
- Regret - Potential optimal reward minus the agents achieved reward.
- Arm/Bandit - An option that has some probability of returning a reward.

1 Introduction

1.1 Introduction

Multi-Armed Bandits(MAB) refers to a general problem where one has to make sequential choices over a set of options in hopes of discovering the best pick as early as possible in order to reap its rewards. Its name comes from a hypothetical casino scenario where a gambler attempts to play several slot machines(One-Armed Bandits) in order to find and invest in the best rewarding machine and maximize his winnings.

As this can be abstracted to a very general problem, it has many practical applications ranging from clinical trials to online advertisement. Many variations of this problem has been introduced in an attempt to capture different contexts that the problem might arise in. As a result we also see the need for a broad variation of solutions, this is because although the problems are equivalent at its core encompassing the explore vs exploit dilemma, they often introduce variables that can have significant effects on how the problem plays out. What follows is the need for different approaches for what at first sight might seems like equivalent problems.

Clinical trials is one of the earliest theoretical explored ideas within MAB problematics. Initially one might ought to use A/B split testing. Where one splits the set of patients in different groups, such that each group is given a specific treatment. You then record the results of each group, and try to establish which treatment is best. The issue with this is that you end up potentially giving the majority of the patients a suboptimal treatment. (Thompson, 1933), (Kuleshov & Precup, 2010) Suggest the application of MAB solution to incrementally treat the majority of the patients with the treatment that has so far the best results, resulting in a reduction of patients needlessly given a suboptimal treatment. In this case each treatment is viewed as an arm on a bandit. When a patient or group of patients are given a treatment, this is considered as a pull of a specific arm.

A more modern emergent field of interest is the web based arena. Where advances in display of advertisement or links such as news articles on a website can result in highly economical benefits. Different extensions of the MAB scenario has been introduced to capture the fact that these areas often also have a lot of external data which can be directly associated with the users and how they interact with the links. This is typically known as personalized content, and is highly common in all sorts of media, such as websites, advertisement or streaming services. And is often referred to as the Contextual Bandit, which is popular in the web based media settings, it uses external data to guide the choice of bandit. Recommendation systems are often used to generate the prior data that is used to guide the bandit selection process. These systems often try and cluster users around certain shared features found in the context of the arms.

It should also be noted that these practical applications of the MAB problem, are a lot more complex in nature then the classical hypothetical casino example where each arm is associated with a independently operating slot machine. Truthfully, if one considers an newspage as an MAB example, which has many different articles as arms. The arms are related in many different ways, either by theme, author or recurrent characters(celebrities) etc, all of which has an affect on how interesting the visitors find the newspage. These relations are often subtle and extensively many, which means that an attempt to capture and use these relations can inflict spuriouse assumptions about how impactful each relationships is. And in some cases one might miss out on the usefulness of elusive features that are not included in the recommendation systems. Or in cases where there are no extra data, these features might be even more beneficial to generate a system that optimizes more true to the environment than a simplified assumption of independent arms.

This thesis considers the use of MAB algorithms as nodes in a network based model, which selects over a set of bandits. The exploration of this setup is divided in two different approaches, Flat networks and Hierarchical(tree) networks. These network models could potentially constructs more nuanced

information, than a single MAB algorithm considers, but at the same time does not seek to gain deep knowledge such as more heavy duty model might do through an extensive learning phase prior to the application e.g: Artificial Neural Networks. This is because it is supposed to be able to act relatively lightweight with an emphasis on online or batched learning. Where the data input is rather stochastic and rapidly changing. The whole purpose is then, to on the go capture slight bits of this complexity in its choices to further improve its behavior.

1.2 Motivation

The spawn of this thesis was in the lights of news articles of a webpage, where an algorithm tries to update the display of articles by using a underlying recommendation system as a prior, and then update live when users interact(click) with the links. The live update is effectively a sequential optimization problem as the algorithm tries to figure out which link is the one the guests actually want to see, and through trial and error correct for this.

The proposal of this could be effective, but it also boldly assumes that all articles are completely independent. As if the theme and content of news articles does not dictate which other articles the users might be interested in. Chances are that if you find concern for the latest tragic mass shooting in the USA, than you probably also care about articles involving gun control or mental health. Is there a way to capture this elusive relationship between the articles on the go, without attempting to label the relationship.

When creating an intelligent systems, it fundamentally relies on a set of assumptions about the world, and the problem at hand. These assumptions heavily affects the choice of technique or models used, and as follows restricts how well the model captures reality and what truly is the best results. In complex and uncontrolled environments you can often not be certain that the results you have are the best you could get, rather they are simply the best you got. In effect, a simple model might be an improvement of the current system, but it says nothing about how well it actually uses the true nature

of the environment.

By introducing sequential dependencies to the MAB problem, the search field quickly explodes, as solutions now has to consider sequential combinations of arms. The fact is that these sorts of problems are even in offline-learning scenarios, considered genuine tough problems, and is in general considered as a fundamentally challenging problem in the field of Artificial Intelligence. Because of this, games such as Chess and Go was once considered too complex for computers. However, great advances has recently been made in this area partly because of the application of a tree search method known as Monte Carlo Tree Search(MCTS)(Chaslot, Bakkes, Szita, & Spronck, 2008).

The main motivation for this thesis is a curiosity towards whether elusive causalities(e.g: the news article relationships), can be used in systems, without the attempts of deep understanding. This is then explored through networks topologies mimicking traits of the MCTS and other network based models.

1.3 Research Method

The fundamentals of the Multi-Armed Bandit field is a mix between hard proofs, mathematical laws and simulated empiric data. This thesis will focus on empiric results, in the form of plots generated from a stochastic simulations, to showcase the performance(regret optimization) of MAB networks versus traditional MAB algorithms, and in the process map traits associated with different network properties.

The topological structures of the networks are initially inspired by graph theory considering directed networks, either flat or hierarchical. The connectivity of the networks are then experimented with, in the lights of what other studies suggests about the performance of typical MAB algorithms and scenarios. One such trait is that the less amount of arms the agent has to concerns itself with, the easier it is to pick up on nuances of the probability distributions of each arm, as you can explore less. Another trait is the prob-

ability distribution variance between the arms. The higher this variance is the less attempts has to be spilled, as one arm is more prevalent than the other (Kuleshov & Precup, 2010).

The research method relies heavily on exploratory science, with weight on stochastic simulations. This is because even though the Multi-Armed Bandit problem is an active and extensively researched field, it is such an encompassing problem with the explore-exploit dilemma at its core, and the online learning networks component of the problem further complicates this as it is not that much researched. Prior research ranges from focuses on hard proof of policies, to research hardly involving the MAB problem at all, but rather focuses on the applicational dilemma. In between we find research trying to explain and stitch together the theoretical with a thought provoking extension of the dilemma, that might have important applicational values. This is where different types of bandit scenarios are introduced to reflect situations where other properties of the dilemma has to be taken in to consideration, e.g: Extra contextual information, internal behaviours and inter-bandit dependencies.

Exploratory science methods focuses on cause and effect relationships between variables. It is suitable in situations where a particular problem has yet to be thoroughly studied, and as a result lacks priorities and formar definitions. This methods is often used as the first stage of research to establish a scope worth further research either in the same studie, or to prep the ground for future ones. Since the problem lacks research, one often has to make due with what is available of literature and data, either in the same domain, or related fields(Hellevik, 1999). In cases of vast data this is often the only game in town, as techniques to derive meaning out of these kinds of data often focuses purely on relations between variable rather then trying to explain the meaning of the relations(Carroll & Goodstein, 2009). Generally speaking, the method is meant to establish foundations for a hypothesis and should not draw definite conclusion based on the results of the exploratory research it self.

The focus of this thesis is on networked policies for MAB online learning. This

approach is not that greatly researched, and the closest relative is found in deep reinforcement learning, where Artificial Neural Networks are used as the policy. However this approach usually relies on a vast training phase, to prime the policy for an enormously complex environment(e.g: Chess, Go)(Silver, Hubert, et al., 2017). Networks in online learning is meant to considered less complex environments, and have to hit the ground running for its very first timestep. The open ended question is then, what sort of network structure can achieve this with profitable results?

Within exploratory research, there are 2 ways of exploring relationships between variables, experimentation, and simulation. To draw the landscape for online learning networks, I will use stochastic simulations to mimic bandit scenarios. The focus is then to discover traits of network models with different configurations in order to separate which parameters are worth investing in. With this in mind I have incrementally tuned parameters of the two network approaches and recorded the results as regret(theoretical optimal reward minus the agents reward) visualized as plots. Some specifics of the research questions result from this initial process, such as the Symbiotic Bandit and the interest in network configurations that can take advantage of the dependent arms in the Symbiotic Bandit.

1.4 Research Question

The process of this exploratory method has to be constrained in order to reduce the scope so that it can show useful progress. In order to do this I will only consider two types of bandit scenarios. The first type is the Bernoulli Bandit, which is suitable because it simulates an environment of independent arms, and a binary reward distribution. First of all this bandit is a simplistic version of the general bandit problem, which primarily concerns itself with the explore-exploit dilemma with no extra influential parameters. It can also be considered to mimic scenarios of web based applications, where user recordings are often binary in the form of a click or no click.

The second bandit will be the Symbiotic Bandit, which can be considered

as an extension of the Bernoulli Bandit in combination with the Restless Bandit. This bandit still has a binary reward distribution, but also contains a set of dependent arms. The usefulness of the dependent arms is that it mimics scenarios where arms share a common feature that is rather elusive in nature. The intention of the agent is then to benefit from the pattern of this common feature, rather than understand it, describe meaning or simply ignoring it for simplicity's sake.

In order to compare the performance of networks, we need a foundation to build upon. This foundation is the single node version of the network, meaning a single normal Multi-Armed Bandit algorithm. I will use the Thompson Sampling method as the basis of all networks, and network performance will then be compared in part to the normal version to see how the network impacts its behavior.

There are many ways of evaluating the performance of the MAB algorithms. One popular heuristic is Regret(opportunity loss) measurement, which is useful when you have access to the actual reward values of the bandits. Regret is then simply the measurement of the discrepancy between the actual values and the agents estimations. A well performing agent is then an agent capable of reducing its regret as quickly as possible without limiting its long term potensial. I will consider instantaneous regret in order to predicts the agents future behavior, and cumulative regret to consider which agent performed best over a set interval.

The network topologies are inspired by the properties achieved by other network based models in vastly different fields. There are however important constraint of the network topologies. They have to select an action and therefore conclude a run, and the cumulative reward of a future run has to be an significant improvement of the prior runs. Also since this is an online learning endeavour, the networks has to be finite in size, and limited when it comes to expansions.

This thesis will not include considerations of computational complexity of the different approaches. Although this is crucial in regards to their applicability

and usefulness, it was considered outside of the scope of this thesis. This is primarily because the problem has been approached through an explorative design, where the main goal is to find out what is worth further research. This means that the process explores a lot of useless paths that is not worth analysing further. Complexity analysis and algorithmic proofs are often an endeavor of it self, hence would take too much time, furthermore in many cases a design is often shown useless in a simple empiric regret plot.

With these limitations in mind, while going through an initial exploration of network and prior studies the following research question where designed:

1. Can network based MAB models achieve lower regret than traditional solutions(TS) in the Bernoulli Bandit(independent arms)?
2. Can network based MAB models achieve lower regret than its single node counterpart on dependent arm MAB scenarios(Symbiotic Bandit)?
3. Are there networked solutions that can reduce regret compared to a single node, in both scenarios, Bernoulli and Symbiotic Bandit?

1.5 Contribution

While the idea of creating networks with MAB algorithms as nodes sounds novel, it has many similar traits with other models and techniques. Ranging from Hidden Markov Model, Bayesian Networks to graph search algorithms or the recent popularized Monte Carlo Tree Search used in google's deepmind project Alpha Go/Zero (Silver, Schrittwieser, et al., 2017)(Silver, Hubert, et al., 2017).

Here, I try to explore different ways of arranging networks for a online explore-exploit problem, such that they further utilize established knowledge about how Multi-Armed Bandit algorithms performs best. (Kuleshov & Precup, 2010) Finds that for most traditional algorithms the only characteristics that matter are the number of potential choices, and the reward

distributions between the choices.

Since networks also allows for structuring more information regarding dependencies between the arms, it provides for use cases where the bandits have some sort of relationships that affects their performance. Therefor I also introduce a generalized MAB problem called Symbiotic Bandits, which is supposed to encapsulate the news website problem with somewhat dependent arms where the sequence of pulled arms has an impact on the reward probabilities.

As well as answering the research questions posted. I will also in the process draw some markers in the landscape of the possible network topologies. This scope will include ideas formed by prior research done on classical bandit solutions and their impact/usefulness on network based solutions. Furthermore I will exemplify with an array of structures to showcase the boundaries of parameters introduced by networked models.

2 Background

2.1 Multi-Armed Bandit(MAB)

Multi-Armed Bandits is a widely recognizable problem that captures situations where an agent has to make sequential choices over a set of stationary options. The assumption here is that given this set of options there has to exist an optimal choice. One of which you may only discover through trial and error. Since problems of this nature often involves some form of investment, be it money or simply time, one ought to seek a sequence of choices that grants the highest reward over investment.

The problem was first proposed by Herbert Robbins (Robbins, 1952), where the name stems from an hypothetical casino situation, where a gambler proceeds to plays a set of slot machines often referred to as one armed bandits, with the assumption that some machines will let you win more often than

others. Choices are often referred to as arms. The act of pulling an arm is associated with some form of investment and a reward in return. The strategy of making choices is called a policy, which is the core part of the agents way of evaluating whether it should explore or exploit.

2.1.1 Bandit Variations

While the original explanation of Bandit problem is the casino scenario, it is mostly meant to capture a real dilemma through an intuitive hypothetical scenario. Under this hypothetical scenario, many others have spawned to reflect real world scenarios that hold similar characteristics, but has extra properties that make the solutions for the problems slightly different.

Bernoulli/Binary Bandit
Is the simplest version of the bandit problem, in which the bandit has some probability distribution of returning a single reward r, otherwise 0. This scenario has a close link to modern scenarios related to web based problems. As user clicks are a binary signal, which can be interpreted as a reward. E.g: Links on a website are viewed as arms, where the agent's goal is to optimize which links are most popular. This could be advertisement links, article links etc.

Contextual Bandit
Here we seek to use additional information to guide the choices in a more informed way. The context can be any historical or current information you have about the choice. The point is that this context information is extra information, that can be used to further leverage a good choice on top of the typical recorded success vs failure data.

A typical example is an online advertisement slot, in which one seeks to display the best suited advertisement for each given visitor. If you already have obtained information specifically about the current user, how can this guide the choices of advertisement (May, Korda, Lee, & Leslie, 2012).

Adversarial Bandit

At each turn you are given a choice of arms, but at the same time the adversary picks the payoff distribution over the arms.

Restless Bandit
This bandit has a internal markov chain, which may change state when it self is pulled or others. In effect, this means that each arm may not have the same probability at any given state of the system, which makes for a highly chaotic scenario to solve (Whittle, 1988).

The Symbiotic Bandit which will be introduced in this paper, is a mix between the Restless Bandit and the Bernoulli Bandit. This is because the Symbiotic Bandit has in this thesis a binary reward. As well as trading a success probability between the dependent bandits, such that the probability of a set of bandit are changed depending on which state they are in.

2.2 Explore vs Exploit

The fundamental issue of Multiple Armed Bandit problems is a tradeoff between exploring or exploiting. Exploring is too seek information about what possabilities is within your scope of choices, whereas Exploiting is an attempt to reap rewards of what is currently assumed to be the best choice. The issue at hand is however problematic, as optimal approaches depends highly on the specifications of the scenario, and often leads to a range possible tweeks to further optimize policies for the given context. Not only will the complexity of the given MAB scenario play a role, but core properties such as the amount of total bandits at play, the probability of a reward given a pull, reward variance or simply the longviety of the problem may have huge implications. On top of these we have circumstantial properties tied to the given bandit problem.

2.3 Classical Algorithms

2.3.1 Epsilon greedy(e-greedy)

Perhaps the simplest policy, in that it has a fixed epsilon parameter that represents the probability of exploration. Otherwise a pure greedy strategy is applied, in which the arm with the highest success divided by failed trials(attempts) is selected.

results, is a 2-by-n matrix containing all prior data recorded, where each column(n) is a specific bandit. The first row(m) contains successful trials, while the second holds failed trials.
indmax, return the index of an given array which hold the largest value.
indrand, return a uniform random index representing any bandit.

$$results_{m,n} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ l_{2,1} & \cdots & l_{2,n} \end{bmatrix}$$

1: **function** $\epsilon - greedy(results, \epsilon)$
2: **if** $random[0,1] \leq \epsilon$ **then**
3: $pick \leftarrow indrand(results)$
4: **else**
5: $pick \leftarrow indmax(results_{1,n}/results_{2,n})$
6: **end if**
7: **return** pick
8: **end function**

2.3.2 Upper Confidence Bound(UCB)

A popular deterministic approach, which is often intuitively referred to as optimism in face of uncertainty, where the core principle is to compare the upper bound of each arms distribution. Meaning that we are comparing the estimated better case scenario of each arm and investing further plays on the

best of those.

results, is a 2-by-n matrix containing all prior data recorded, where each column(n) is a specific bandit. The first row(m) contains successful trials, while the second holds failed trials.
indmax, return the index of an given array which hold the largest value.
numPlays, the current number of times a bandit has been played.
step, how many time steps the algorithm has been played so far.

$$results_{m,n} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ l_{2,1} & \cdots & l_{2,n} \end{bmatrix}$$

function $upperbound(step, numPlays)$
    return $sqrt(2 * log(step + 1)/numPlays)$
end function

function $UCB(results)$
    for i In $results_{1,n}$ **do**
        $ub_i \leftarrow upperbound(sum(results), results_{1,i} + results_{2,i})$
        $sample_i \leftarrow (results_{1,i}/results_{2,i}) + ub_i$
    end for
    return $pick \leftarrow indmax(sample)$
end function

Being deterministic has a certain advantage when it comes to proofs. You can also rerun experiments from a given state and understand its choices.

2.3.3 Thompson Sampling(Bayes Bandit)

Thompson sampling is a probabilistic policy, often referred to as a probability matching method. This policy compares samples drawn at random from a Beta distribution of each arm, where the parameters of the distribution are the two following: successful trails and failed trails.

The method of Thompson Sampling is older than the theoretical formulation

of the multi armed bandit problem. Where it was originally introduced in 1933 by Thompson in the interests of finding out which of two drugs were better. A important metric in this quest is that he wanted to expose as few patients as possible to the lesser drug while finding the best (Thompson, 1933). This process is an obvious and genuinely example of a explore-exploit problem that has real consequences.

This policy has lately shown great empirical results in most MAB scenarios, and is therefore the primal choice in this thesis (Chapelle & Li, 2011). Being probabilistic also means that it can work in batch updated situations, which is often the case in online applications. This trait, also lessens the need for computational speed in networked MAB algorithms.

results, is a 2-by-n matrix containing all prior data recorded, where each column(n) is a specific bandit. The first row(m) contains successful trials, while the second holds failed trials.
indmax, return the index of an given array which hold the largest value.
Beta, The beta function has two parameters, alpha and beta, which forms its curve.

$$results_{m,n} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ l_{2,1} & \cdots & l_{2,n} \end{bmatrix}$$

1: **function** $thompsonSampling(results)$
2: **for** i In $results_{1,n}$ **do**
3: $sample_i \leftarrow rand(Beta(results_{1,i}, results_{2,i}))$
4: **end for**
5: **return** $pick \leftarrow indmax(sample)$
6: **end function**

An intuitive explanation of the Beta function in Thompson Sampling is that it tries to aline its probability of choice with the true probability of the arm, hence its class name; probability matching policy. It starts off knowing nothing, and therefore assumes nothing with a uniform distribution over the probabilities. The more data it gathers the more it allocates its resources over

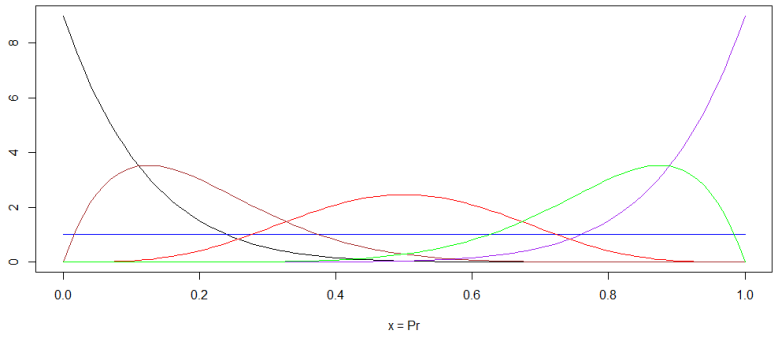

Figure 1: Beta Function

the estimated true arm probability. Moving from a flat uniform distribution, to a highly peaked distribution over a small estimate on the x-axis.

Say we have two arms with reward probabilities of pr(0.2) and pr(0.5). Let us ignore the stochastic noise of the arms by the use of the law of large numbers, and assume that the data are reflections of the true probability of the arms we see the development of the probability estimations in figure reffig:ts2080 and reffig:ts5050.

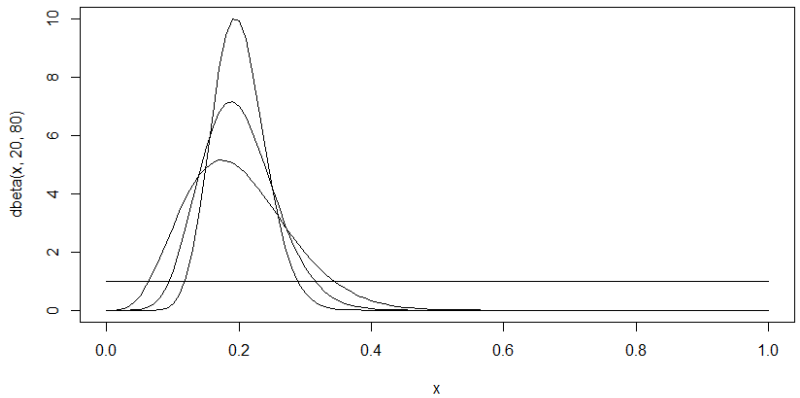

Figure 2: Pr(0.2)

Thompson Sampling is the process of sampling from this given Beta curve

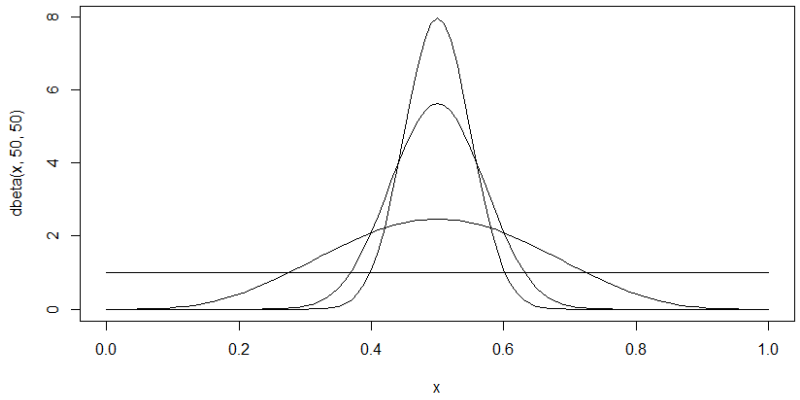

Figure 3: Pr(0.5)

from each arm, and then proceeding with the arm with the highest sample. The result is that over the long run one will tend to pick the arm with the probability distribution that is heaviest towards the righter part of the x-axis. In other words, the arm that has a curve heavier over high probabilities.

Finally, the thompson sampling method does not necessarily have any parametrics to tune, making it easy to use as a baseline when comparing with its net versions. As it is easier to not concern yourself with the possibility that the algorithms could have been optimised better, and would therefore show different results.

2.4 Markov Chain

Markov chain is a simulation of a probabilistic change between states. Where the probability of the new state is only dependent on the current state, known as the Markov Property. This property makes it so that you can calculate the probable future outcome, or state, given N amount of steps, and the current state. No prior history is needed. Making it so that you can estimate the probabilities of any modeled system.

2.5 Markov Decision Process

Is a mathematical framework for modeling decision making. It models decisions by states and actions, and is associated with many field other field other than reinforcement learning, or the Multi-Armed Bandit problem. Markov Decision Process is an extension of the Markov Chain, as it is reducible to a Markov Chain. The difference is that MDPs model actions or possible decisions the agent can make.

Typically used with these terminologies:
S: Is the state of the environment.
A: Is the set of actions the agent can choose between.
R(s,a): a function that returns the reward associated with choosing a certain action in a given state.
T(s'|s,a): is a probability transition function. Which establishes the likelihood that the environment will transition to **s** given that the agent chooses action **a** in state **s**. The probability of state **s** given **s** and **a**.

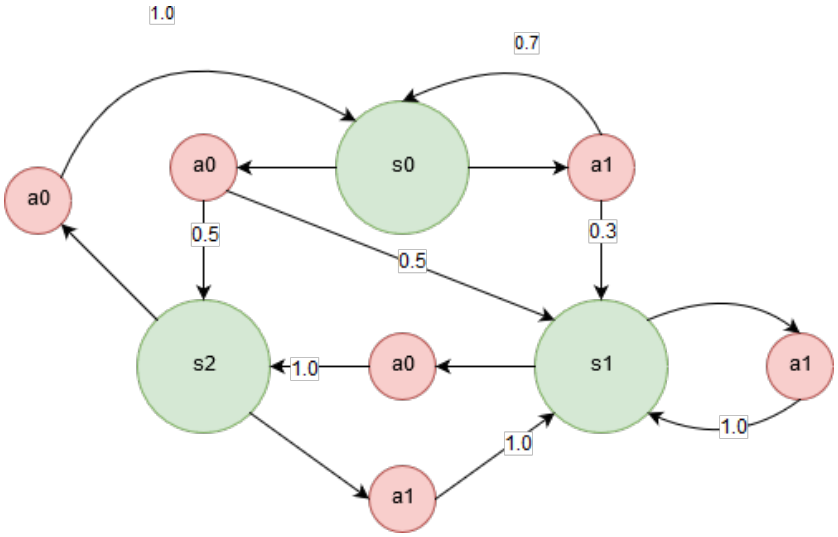

Figure 4: Markov Decision Process Example

The formal Multi Armed Bandit problem can be described as a single state Markov Decision Process, where each action has a 100% probability of returning to the only state.

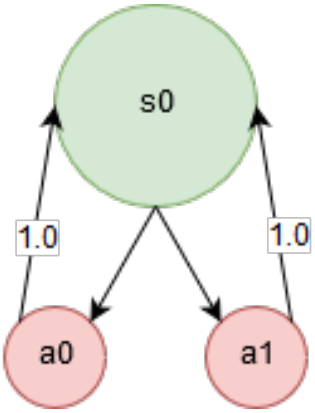

Figure 5: Multi-Armed Bandit Markov Decision Process

2.6 Reinforcement Learning

The Multiple Armed Bandit problem is fundamentally a reinforcement learning problem. Reinforcement learning is a subfield of machine learning, but has conceptual work in many different fields such as: behaviourist psychology and game theory, to name a few.

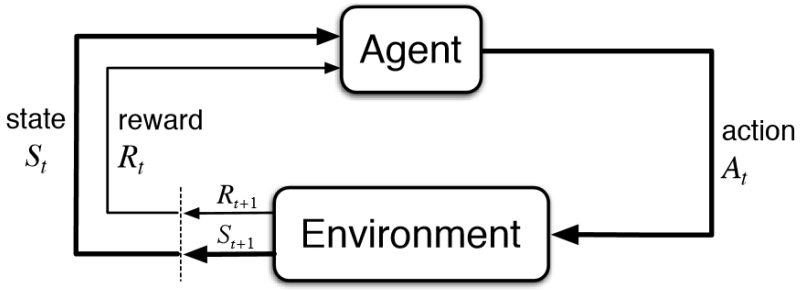

Figure 6: Reinforcement Life Cycle
(Sutton & Barto, 1998)

Reinforcement learning is often quickly assumed to be about creating systems that portray a fictional level of intelligence. As if the goal is to create a human brain, or at least a systems that behave like a human. In reality, its process and theorems are much more related to step wise optimizations based on

states.

In recent time, large steps has been made in systems that perform better than humans in game previously assumed to be out of the scope of computers. Such as Go, Chess, and other games that requires a well understanding of virtual physics and moving objects in a game. However the field has had a long successful run in less grandiose areas prior to this, be it clinical trials, online advertisement or resource allocation, but it is often not expressed as a significant deal. As the issues are at its core a question about exploration versus exploitation, it is the other stuff built around the reinforcement learning core that makes the system as a whole more interesting, e.g: a systems that can learn any type of game, or reach superhuman excellens in games believed to requiring human strengths such as intelligence and intuition.

2.6.1 Model-based vs Model-free

Model Based

A model based approach gives the agent a reference that it can use to make prediction about future states. What follows is that after some iterations of training, the agent can estimate the transition function, T(s'|s,a), given a state and a set of actions, predict the future state and the reward function, R(s,a), associated with these actions. This model can then be used to plan ahead (Sutton & Barto, 1998).

Model free

Model free approaches does not estimate the transition function, but rather learns by trial and error to estimate the reward function alone given a current state and its actions. In many cases this is sufficient to solve the problem. The best known model free reinforcement learning method is called Q-learning. Where a Q-matrix operates as the brain of the agent, containing all states and its associated actions. The process of the Q-learning algorithm is to use this Q-matrix as a way of picking the next move given the current state and its actions estimated rewards. It then updates the Q-matrix with new rewards associated with the specific state and action (Sutton & Barto,

1998).

2.6.2 On-policy vs Off-policy

On-policy
On-policies are methods where the agent uses a policy to directly interact with the environments. Given the state and optional action, the agent learns a policy to directly make a choice (Sutton & Barto, 1998).

Off-policy
Off-policies separates its policy of choice from the learning function(evaluation function). In other words it uses two methods in its process. One method to evaluate the different values associated with taking certain action given a state, independently from any policy, and a second method, the policy, to actually make the choice over these values (Sutton & Barto, 1998).

2.7 Monte Carlo method

Monte Carlo method is a class of algorithms that rely on random sampling until a substantial numerical value is aggregated. These methods are often used to simulate or solve problems that are deterministic in nature, and they shine particular bright in scenarios where other approaches are teadesum or impossible. One such problem is the exploration of possible movies in a game where the branching factors is to large in apply deterministic approaches over. One such particular approach is called Monte Carlo Tree Search(MCTS).

2.8 Monte Carlo Tree Search(MCTS)

Being at the core of great advances within general artificial intelligence. MCTS is a method that excels at finding good choices in problems with large amount of branching. In recent time it has been used by Deep Minds, AlphaGo and AlphaZero to master the game of Go, and to create a single

system that can learn how to master both Go and perform better than any other chess engine so far, without any prior implemented knowledge of the game simply by playing itself (Silver, Hubert, et al., 2017). MCTS uses a MAB policy to exploring branches at each step which seems most promising, and runs randomised simulations from that given choice to record success or failure. It is an immensely powerful tree search method, which is capable of good results in situations where the search space numerous (Chaslot et al., 2008).

Monte Carlo Tree Search has 4 stages.

- Selection - According to a specific policy, move down the tree from the root to a selected leaf.
- Expansion - Unless the game is over. Establish a new node at the end of current leaf.
- Simulation - Play random choices from the new leaf.
- Backpropagation - Record and update the success or failure of the new node, and its path back to the root node.

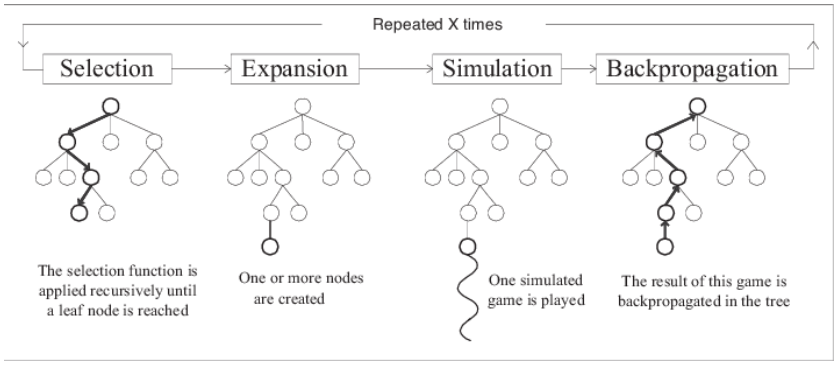

Figure 7: Monte Carlo Tree Search
(Chaslot et al., 2008)

Traditionally, the policy used to selecting a path has been of the Upper Bound Confidence class, but other policies are also experimented with such as Thompson sampling (Bai, Wu, Zhang, & Chen, 2014)

The selection process of the MCTS is essentially a Multi-Armed Bandit algorithm. Which is of course essential to the rest of this thesis, but the part of the system that is mention worthy in the scope of this thesis, is the backpropagation stage. Typically when backpropagation is mentioned, it is understood as the calculation of a gradient by the use of the chain rule, which is popular with the rise of Artificial neural networks. However, in this thesis the form of backpropagation referred to is the one used in the MCTS. Where one simply passes information up and back the same path as one ventured down a hieratical tree.

2.9 Online machine learning

Online machine learning is a subset of reinforcement learning. The word online does not refer to internet or any web applied machine learning. Rather it revolves around learning over sequential data input. Such that there are no opportunities to learn over a dataset prior to the application of the agent. The classical Bandit scenario is a archetypical example, where the agent is has to make predictions at the same time as its being trained.

2.9.1 Batch learning

A slightly less constraint situation are the batch learning scenarios. Here the agent is not feed sequential data based on its prior choice, but rather, the agent is in intervals given a set of data to update over. This of course also means that the agent as a less instantaneous impact on its behavior. As the results of its behavior is only given after it self has made a sequence of choices. Batch data can however be analysed in more nuance. In search of patterns.

2.10 Connectionism

Is a umbrella term describing emergent processes in network based models. The term is also used in many other fields than machine learning, e.g: Linguistics, Cognitive science etc. However, within machine learning the most well known connectionist models are those of the Artificial Neural Networks(ANN) family, originally coined Parallel Distributed Processing(PDP), these processes are inspired by the nature of neural processing. In these systems information is processed in a distributed manner, such that problems are solved in a emergent manner where each node of the network only contains a very limited amount of knowledge. The connections and data flow between these single nodes as a whole are what generates the capability to solve advanced problems(Rumelhart, Hinton, & Mcclelland, 1986).

Today this class of algorithms consists of many different structure, where both how data flows through the system and how data is viewed at each node are parameters that there are endless of research behind. This also require immense expertise trying to find optimal combination that may result in a useful systems. Lastly, a crucial component that where pivotal for the booming of these models, are the learning or correcting methods of these systems. Until backpropagation where introduced artificial neural networks where considered purely a theoretical endeavor. As the network had no way of automatically acquiring the right specification(tuned parameters).

The structures I will explore does not take direct use of these ideas, and in many ways, especially some of the structures, does not resemble the process in any other way other than it could be visualized as a network. They are still worth mentioning as they are network based models, and functions as inspiration of technique in acquiring more complex information in online learning processes, which I will experiment with through alternative versions of the bandit scenario with dependent arms.

ANNs usually adjust weights that are results of the activation functions of the prior layer. Each node input is a single simple number form each edge.

While in the online learning networks in this thesis, has nodes that hold their own success/failure matrix records, where each input is usually a way of modifying how this matrix is looked at when it is put in to the explore-exploit algorithm.

2.11 Thesis specifics, parallels and differences

I view usefulness of these network models by two different properties. Branching Factor, and Information Leveraging. These are two ways of viewing how to control how information is passed through the system. It should be noted that they are rather arbitrary and vague, as they can essentially mean anything and possess any arbitrary combination. I combine these with an overarching design in the form of tree based or a flat structure, which further emits constraints that shape a more reasonable field worth exploring.

2.11.1 Branching Factor

Branching factor is simply how many edges connect a node to the rest of the network. Depending on structure of the model, this effects the different ways a network can make choices and learn. A branch symbolise an action that the agent can choose. Either directly at a bandit returning a reward, or to a new internal node. This structural property by it self can create effects mimicking binary search, where this limitation of branches makes it so that the search field is cut in half at each step.

2.11.2 Information Leveraging

The original MAB problem, can be symbolised a single state MDP, with N amount of options. In other words the agent only uses one set of data to pick an arm, but in networked models, the agent often split data up in different branches. Information Leveraging is then the process of combining or neglecting data from many branches to make a more informed final decision.

One example could be using other data from other paths or nodes as weight in the current selected one.

This could resemble the connectivist approaches such as an artificial neural network, were the information flow is being processed parallely to converge to an output. In contrast to a strict binary search where the information flow reduces to a single branch at each point while moving down the tree.

2.11.3 Tree Net

A tree usually has a root node where the process starts. The next steps involve moving down the tree until a leaf is reached and the process is concluded. Since this a reinforcement learning process, the tree needs to acquire the results and gain some knowledge in the process to use in the future choices. Generally, this is done by passing the reached results back up the tree. A tree model is then a way of creating several layers of sub-choices before reaching the bedrock of bandits as a normal bandit algorithm would work. The use of branch numbers and information leverage in the layers between the root and bandits, is then responsible for generating smart choices both in the present and the future.

2.11.4 Flat Net

Flat networks has in difference to the tree based net no distinct point of origin. Its origin is randomly selected somewhere in the net between the bandits. A time step is concluded when a agent has selected a bandit to run. The way the agent operates through the net is then determined by branching factor as a way of bridges between the bandits and its intersections

2.12 Law Of Large Numbers

Since we are dealing with stochastic processes in multiple armed bandit scenarios, one way of measuring an algorithm's performance is to samples its

measurement N amount of times and look at the average measurement in comparison to each other. The fact that the average should equate to the performance of the process, lies in the theorem of the law of large numbers. That states that as the sample N approaches infinity, the average of the observed value will approach the true average value.

$$\overline{X}_n = \frac{X_1 + \cdots + X_n}{n}$$

$$\overline{X}_n \to \mu \quad for \quad n \to \infty$$

$$Pr(\lim_{n\to\infty} \overline{X}_n = \mu) = 1$$

Where μ is the true expected average, and X_n is a random sample.

2.13 Regret

Regret is a way of measuring outcome when making decisions under uncertainty. In MAB scenarios it is formalised as the optimal potential reward minus the currents agents reward, at each time step. Capturing how far away the agents choice is from the optimal choice in a numeric value.

The general idea is that a hypothetical lucky agent that always picks the best arm is considered as the optimal strategy. This lucky agent has 0 regret, as he always picks the best choice. Then we measure how far away our agent is from picking the optimal.

Regret as a performance measurement has however its down fall in real world scenarios, as you often do no have access to any true optimal model to compare the agents result with. This means that in real world applications of MAB algorithms, other performance measurement methods are often used. Such as in web based scenarios, Click Through Rates(CTR) are often used to compared the results of the different algorithms (May et al., 2012)(Chapelle & Li, 2011).

2.13.1 Instantaneous Regret

There several ways of looking at regret, one of which is instantaneous regret per time step. What is interesting here is to see how much change of regret is achieved between each time step. This is done by sampling rewards of N amount of a complete sequence of choices conducted by the agent. Then an optimal lucky agent is considered to hold the true mean value of choosing the optimal arm at each time step. Differences in regret is then calculated by subtracting the agents reward, which is divided by the amount of samples, from the theoretical optimal agents reward.

$\mu_{j(t)}$ is the reward collected by playing an arm at index j at time step t.
$\mu^* = \max_{i=1,..k} \mu_i$ is the reward gained by selecting the maximum scoring arm.

$$R_t = t\mu^* - \mu_{j(t)}$$

2.13.2 Cumulative Regret

In order to tell which bandit solution performed best, regret wise, over a simulated interval, one has to combine the regret of each timestep. The result of this has two ways of being showcased. One way is to simply show the numeric regret value as the sum over the whole interval. This is a concise way of showing how one approach is simply a better choice then another, given configured timespan. A more explanatory option is to show a plot of cumulative regret over each timestep. This shows the process of which regret is accumulated at each choice, but can be considered redundant if instantaneous regret is also considered.

$\mu_{j(t)}$ is the reward collected by playing an arm at index j at time step t.
$\mu^* = \max_{i=1,..k} \mu_i$ is the reward gained by selecting the maximum scoring arm.

$$R_T = T\mu^* - \sum_{t=1}^{T} \mu_{j(t)}$$

2.14 Alternative Measurements

2.14.1 Percentage Optimal Arm Plays

Another popular measurement of the MAB scenario is by capturing how much the know optimal arm is played at any given time trial. Many MAB scenarios considers reaching a point of where the optimal arm is played 100/

2.14.2 Click Through Rate

Like most real world applications of MAB problem, web based scenarios are highly stochastic and uncontrolled. One popular way of measuring the performance of algorithms in scenarios like these are through click logs, which record visitors behavior in regards to the content of the site while on the website. This log can then be used to view click through rates associated with each website version running its unique configuration (May et al., 2012).

3 Related Studies

3.1 Algorithm Results

There exists a vast pool of proofs and empirical data on the behavior of different MAB algorithms. Under different circumstances, some arguments are made for that simpler heuristics can often outperform more advanced ones. This is simply a result of the fact that algorithms perform differently depending on their strengths, and the setting of the experiment. One example is the highly popular Upper Confidence Bound(UCB) policy, which has a good performance in cases of small numbers of arms, and a high reward variance, but has a significantly weaker performance if one were to increases the amount of arms. (Kuleshov & Precup, 2010) shows the importance and difference in arm scales between the algorithms. Another such point is made about

reward variance, it is important as the strengths of the different algorithms differs depending on the scenario.

Probability matching policies has a few advantages that is especially useful in this thesis. In that they are well suited in cases of batch updating (Scott, 2010). The probability matching policy that is especially interesting is the Thompson Sampling method. (Chapelle & Li, 2011)(May et al., 2012)

3.2 Dependent Arms

One way of looking at dependent arms is set in the lights of online advertisement. (Pandey, Chakrabarti, & Agarwal, 2007) argue that similar advertisements could have similar click probabilities. Their approach is to assume that similar arms can be considered as a cluster. In effect, this means that a large amount of arms can be reduced to a lesser amount of clusters, which algorithms can make use of.

(Wang et al., 2017) looks at dependent arms in the form where arms are considered related within a single cluster if they shares a topic. The practical cases for this approach is rooted in recommendations systems using collaborative filtering for movies and news.

3.3 Different ways of interpreting network models

There are generally not that much research about Multi-Armed Bandit algorithms set up as networks. There are however many algorithmic approaches that considers networks as a model, where maybe the best known ones are the Artificial Neural Networks family. The nature of these model are vastly different from those of the MAB problem. Artificial Neural Networks are networks that prevails at approximating any function, by feeding the model enough training data, until its approximation is usable. Multiple armed bandit algorithms are essentially explore-exploit algorithms which for the most part addresses on the go estimation. Meaning that they try to, by some

policy, make a more and more informed guesses. In other words, the model is built and used at the same time, and its rewards is a cumulation of the process.

In this thesis, I view networks in somewhat general equivalent terms as a Artificial Neural Network. In that each explore-exploit algorithm is viewed as a node in a larger network, governing the total reward cumulation. The model is a network that accumulate information both at a single node basis, but also at a macro level. The end result of the performance is that of the macro reward.

In the neural network family there are also other algorithms with different topological structures, such as Boltzmann machines or hopfield networks, which are not feedforward structures, but rather a recurrent way of storing associative memory.

Random Forest Bandit

The Contextual Bandit is a highly useful bandit for real world scenarios, and in the light of network based approaches has some research. The general notion is that in the Contextual Bandit you can often benefit by having a slightly more advanced model. This is because you have extra information that might be pivotal to the what choices are optimal.

(Féraud, Allesiardo, Urvoy, & Clérot, 2015), proposes a Random Forest approach to the contextual bandit, where a decision tree is built over the contextual data to find the optimal combinations of these features. In the paper they exemplify this by how an advertisement on a web page has contextual data in the form of what site it is on, where its positioned on the website and other user specific data. By dividing these features in a decision tree, they propose a way of valuing combinations to further improve advertisement displayal.

Neural Bandit

(Allesiardo, Feraud, & Bouneffouf, 2014), uses a pre-trained Artificial Neural Network as a way of evaluating the external data in the Contextual Bandit scenario. The approach tries to create a model that value of rewards mappes

to a context.

3.3.1 MCTS Applied

The version of network interpretation that assimilates best with the work done in this paper is the Monte Carlo Tree Search(MCTS)(Chaslot et al., 2008). This method has also been of great inspiration. It is especially fitting as the Multi-Armed Bandit problem is a component of this algorithm, but its success stories spawns from a slightly different approach than the network structures I am considering. MCTS often represents each split branch as a new optional way in some step based strategy. It is also highly dynamically as it expands its branches when needed. While this thesis, considers more rigid structures, where branches might not directly symbolise a new state, but is rather a way of severing the information into smaller/local portions.

The algorithm has been used in several different games, as a way of simulating intelligent behavior in game AI such as the popular turn based strategy computer game Total War: Rome 2. More recently it has been used to achieve hugh steps in systems considered to reflect more general intelligence. With systems such as AlphaZero beating all contenders both human and expert knowledge fed systems, in both Chess and Go starting with nothing but the rules/restrictions of the game (Silver, Hubert, et al., 2017)(Silver, Schrittwieser, et al., 2017).

3.3.2 Deep Reinforcement Learning

Deep Reinforcement Learning is an advanced way of using a deep Artificial Neural Net as a policy in a reinforcement learning problem. This opens up possibilities to run simulations that teaches its policy how to act from scratch. Typically, deep neural nets has a constraint, in that to be able to teach the neural net its functional approximation. The system requires enormous amounts of labeled data, so that given a set of input, one can correctly state by how much its prediction is astray from the actual value.

While in situations where deep reinforcement learning is appropriate, one can run simulations, where the agent is a neural network that is taught state-action mapping to associated rewards. This is done by the agent making prediction given its current state, and action options. The correction come in terms of the reward. (Silver, Schrittwieser, et al., 2017)

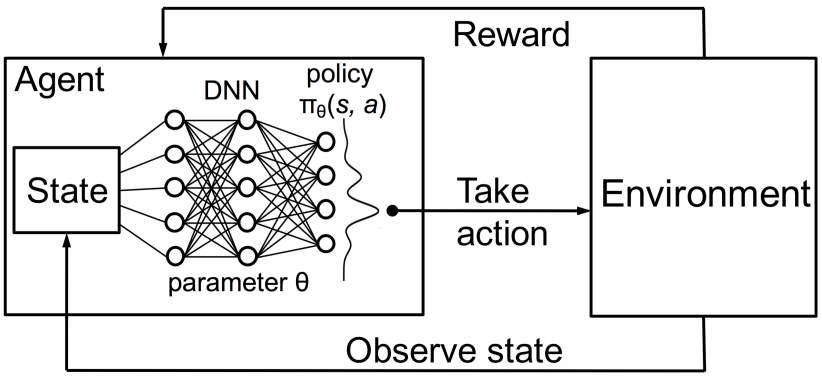

Figure 8: Deep Reinforcement Learning
(Mao, Alizadeh, Menache, & Kandula, 2016)

3.4 Inspiration

There are a few points of inspiration gathered from related studies. First of which is the point that most algorithms perform better in circumstances with few arms(Kuleshov & Precup, 2010). This led to the idea of trying to reduce the amount of arms considered at each timestep when we are dealing with scenarios of large pool of arms. This is call sequential arm elimination, which is a special interest case of the Multi-Armed Bandit scenario (Shahrampour, Noshad, & Tarokh, 2017) (Jamieson & Nowak, 2014). Where one usually seeks to permanently eliminate arms, until one are left with only one estimated optimal arm. This is not the approach I will take in this thesis, but it inspired a few approaches that effectively play a more lightweight dynamic version of this, where the binary tree based versions are especially in thought.

Generally, when we see solutions that apply more complex approaches, they are often aimed at the Contextual Bandit scenario. The use of Random Forests (Féraud et al., 2015) or a Neural Networks(?, ?) is then aimed at describing the contextual data in a way that the online selection process can achieve even better rewards from. The main issue in regards to this thesis is of course that these contextual models are usually trained offline. The use of complex models such as Artificial Neural Networks, and Random Forests in a online learning process is still inspiring in regards to this thesis nonetheless.

4 Experiments

4.1 Experiment Parameters

The structures of the experiments are drawn from other papers studying MAB algorithms in an empiric fashion (Kuleshov & Precup, 2010)(Chapelle & Li, 2011). Where one typically aims at showing the performance of algorithms in comparison to each other in a plot where the best performers are those with the least cumulative regret. One other method is to compare a single algorithm in regards to an realistic estimated regret bound. As expressed in (Kuleshov & Precup, 2010), how the algorithms scale in lights of how the MAB scenario parameters are set up matters extensively. Certain algorithms perform differently depending on the setup. Where some algorithms are better at severing differences between a small set of arms, others prevail at large amounts of arm. There are also cases for batch update vs online update, where deterministic approaches are less dynamic in cases of batch learning, non-deterministic approaches such as probability matching methods has an inbuilt way of distributing its pulls without getting direct feedback (Agrawal & Goyal, 2011).

K Amount of Arms

The larger amount of arms an agent has to consider, the more trials it requires to establish any secure beliefs. A good decision is dependent on relative

factors between the arms. As information is relative to other data in the Multi-Armed Bandits problem. A successful or failed pull of any one arm is relative to the total amounts of arms, and the total amount of pulls recorded so far. In a cases of 2 arms, even a uniform random policy would play the optimal arm 50% of the time. If there were 100 arms, the same policy would only do so only 1% of the time. Furthermore, depending on how long each sessions lasts. In the case of 100 arms data have more value, and therefore has to be distributed more sparingly. This means that some arms may only gets a few pulls to test its usefulness.

Probability Distributions

Seen from the extremities, one could imagine a two armed bandit scenario; let us consider probabilities of 0.9 and 0.2 versus 0.2 and 0.25. The former scenario does not require a lot of exploration, and making the wrong choice will have significant differences in expected reward. While the latter probability distribution, requires an enormous amount of data to separate the nuances, but a wrong choice is still very close to the optimal. (Kuleshov & Precup, 2010)

Reward Variance

(Kuleshov & Precup, 2010) expresses the importance of the reward variance and its effect on the algorithms. Reward variance, is the difference between the amount of reward that a successful arm pull might return. An arm with high variance, means that the arms can at any successful pull return a highly different amount as reward. While a low variance implies that the arms return is within a small range.

This metric is not used in this thesis, as we are dealing with a simplistic version of the bandit problem(Bernoulli Bandit) which is directly associated with click through recordings of a website. The reward is binary, not a sample from a distribution. Much like a click on a link is a binary signal. In effect, Bernoulli Bandit becomes a subproblem, as a binary reward could be any arbitrary number, where its reward distribution is so highly stacked over one value that there effectively are no variance. This is also done deliberative to limit the scope of this thesis, as there are endless different versions of the

MAB problem.

Session Longevity

Generally, we want an agent that picks up on nuances from the very beginning, so that it can reduce regret from the very first steps. This would reduce the total regret in the long run, but at some point the experiment has to end. This point is usually picked as result of most algorithms flat lining at some regret percentage. This point, where differences in regret becomes insignificant in comparison with other algorithms is highly dependent on a combination of the previously mentioned parameters.

Sample Size

Sample size, is a parameter not directly associated with the MAB dilemma, but for empirical studies of stochastic events, it is required to establish a mean reward. Ideally, we would want the samples size to be as close to infinity as possible. As it relies on the law of large numbers to illuminate the true mean value. What follows is that an arbitrary number has to be picked to be able to establish the approximate mean. (Kuleshov & Precup, 2010) uses 1000, but the higher the better. As not only is the numbers more true to the mean, but visually it is less noisy when we are comparing plots. The cost of precision however, is the need for extensively more computation. In the end, this is a matter of access to hardware, time and how much precision is actually needed.

4.1.1 Experimental Setup

The setup considers the parameters and experience of other non-networked MAB studies as a baseline. This led to a design which attempts to shed light on the impact of scale of arms and variance of cumulative and instantatiouse reward between the different algorithms. (Kuleshov & Precup, 2010) is primarily used as the main inspiration in terms of parameter defaults, but I view these in comparison to (Chapelle & Li, 2011), (May et al., 2012), as well as other studies. The key point is that the simulations and experimental design varies a lot depending on what MAB scenario is at hand, but also what is

interesting to capture in terms of the solution. In cases with large numbers of arms, solutions often compromise by seeking for simple arm improvements rather than finding the best arm. Nonetheless, there are certain boundaries such as, it is often a goal in showing that the algorithms does not improve any further at some time step. This range in then determined by the rest of the parameters.

In this paper the sample size is set at minimum 1000. Where the longevity of the experiment is based at 1000, but are explored up to 10000 if any significant development of regret continues. Nonetheless, what is important is the cumulative regret in comparison to other algorithms over the same span. Instantaneous regret is also salient to ensure for cases that does not allow the algorithm to reach a zero regret state.

I terms of arms scale, the choice is rather arbitrary. This thesis uses 8 as the low end, and 128 as the high end. This choice is based upon a few considerations. One of which is inspired by other studies, in terms of range of the scale, but the distinct numbers are made as a result of being able to use certain network algorithms without writing too many exceptions in the logic of the algorithms. The binary scale is made partially as the branching factor of the network algorithms start at 2 as the minimum. And is there for easier to scale by the same factor.

Networked algorithms are compared to its corresponding normal(single node) MAB bandit. Since this thesis is based around the Thompson Sampling method, and most network approaches uses it as its nodes. We compare most network algorithms towards a normal Thompson Sampling algorithm. The initial value matrix of each algorithm is put as 1. As (Kuleshov & Precup, 2010) refers to this as optimistic initialization, and claim that it gave the best results. In view of the Beta distribution used in Thompson Sampling this also makes sense in terms of the development of the curve, and how the curve is supposed to try and match the true distribution of the arms.

4.1.2 Test Case: Bernoulli Bandit

The baseline test case is of the Bernoulli Bandit scenario. In many ways this is the most stripped down version of the Multi-Armed Bandit problem, which focuses mostly on the core explore versus exploit tradeoff. This also means there are less variables to consider when looking for a optimal solution. To reiterate; this problem considers a set of independent bandits with a given probability distribution. A successful pull of a bandit returns a single stationary value, say 1, otherwise 0. The success probabilities of each arm is drawn each sample run from a uniform distribution between the interval [0,1].

4.1.3 Test Case: Symbiotic Bandit

This bandit is not meant to be considered as a problem of its own, but rather as an extension of the Bernoulli Bandit inspired by the Restless Bandit. The intention is to test bandits that can perform optimally at the Bernoulli Bandit, but does not close the door in regards to making use of slightly more complex data. Its tests consists of a set of Bernoulli Bandits, with a given probability distribution. Then a subset of the bandits are given a probability bonus that is passed between each other when they are played. This scenario introduces a fake optimal regret floor, if you only consider independent probability distributions. E.g: a set of 8 bandits with probability distributions: [0.1,0.1,0.2,0.2,0.2,0.1,0.05,0.3]. The optimal independent choice is bandit nr.8 with a pr(0.3), but given that a probability bonus state of pr(0.3), is passed between two suboptimal bandits nr 1 and 2, the real regret floor is at 0.4. If sequential choices between the two are made, then the distributions are drawn each sample run from a uniform distribution in the range [0,0.5]. While the symbiotic relationships has a added bonus of 0.5. As a result the independent optimal probability is between 0 and 0.5, while a symbiotic relationship can be between the probabilities 0.5 and 1.

4.2 Early Networks

Leading up to the final network designs discussed later in this paper, there was an initial phase where more fundamental design idea where settled through rapid prototyping. These early proto-networks were not included in the module, or the in any of the final simulation in this paper. The reason for this is because the majority of the results did not perform well, on top of the fact that they did not have any seemingly interesting properties worth exploring. The selection in this phase should of course be taken with a grain of salt, as there might very well have been hope for some of these configurations. However, this thesis is not meant to prove any particular approach, rather it is meant to suggest paths for future implementation.

Another important aspect for the reason they are not included is that these results were less statistically sound, in that they often did not use a sample sizes as large as related MAB papers suggest(Kuleshov & Precup, 2010). This is partly because the early stages of the network exploration tests, were carried out more light hearted in search for configurations worth spending more time on.

Figure 9 shows one of the earliest attempts at creating tree structures of different MAB algorithms as nodes. Here we see a scenario of 10 Bernoulli Bandits over a span of 2000 timesteps. Where I have compared normal Bayes(Thompson Sampling) and epsilon-greedy, to a two level overlapping tree. The results suggests that nested Bayes is the best configuration. Which could even outperform normal Bayes in percentage of Optimal arm plays. This result is the main reason that the rest of the network constructions does only use Thompson Sampling as its nodes. Of course a caveat should be made, regarding the fact that this is only 1 type of configuration, and that I only tested mix of epsilon greedy and Thompson Sampling.

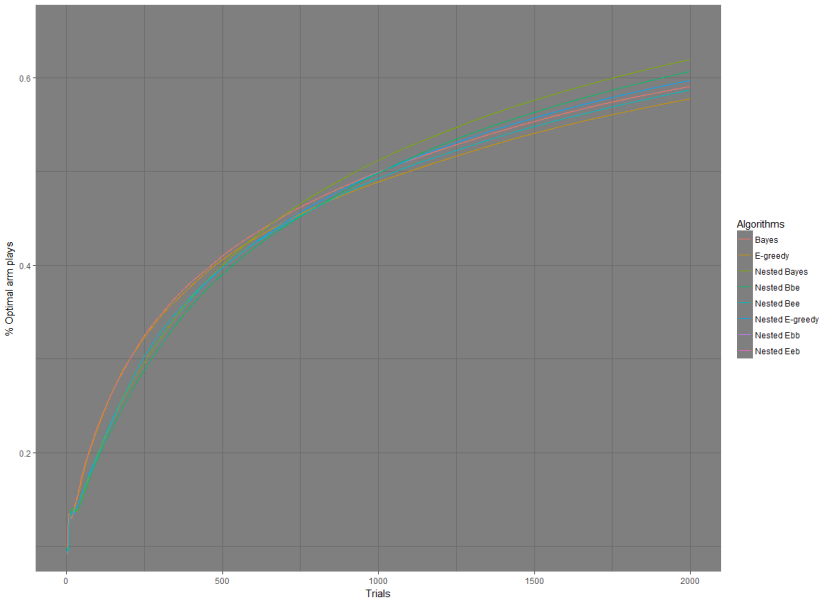

Figure 9: Nested Variations

4.2.1 Topological Usefulness

The network design of this paper is heavily influenced by methods used in indirectly similar fields. This neighborhood of related methods is extensive in many different dimensions, including Binary Search Trees, Artificial Neural Networks etc. When we consider these neighboring methods, its is often transparent and well understood what property makes these techniques useful in their own domain. e.g: Binary Search Tree methods generally only has to compare half of the values, while connectivist methods split input in weighted parallel processes. In regards to this thesis, the details of the methods are not that important, as we are not interested in the same use case. When we apply binary tree to the MAB problem, we do not seek to reduce lookup speed, but rather to impose an estimated order of success probabilities between the arms to further reduce low value explorations.

A key limitation which occurs when introducing these properties, are that as seen in figure 10 the networks still has to operate over inputs of reward and environment status, while selecting a single actions as the output. By

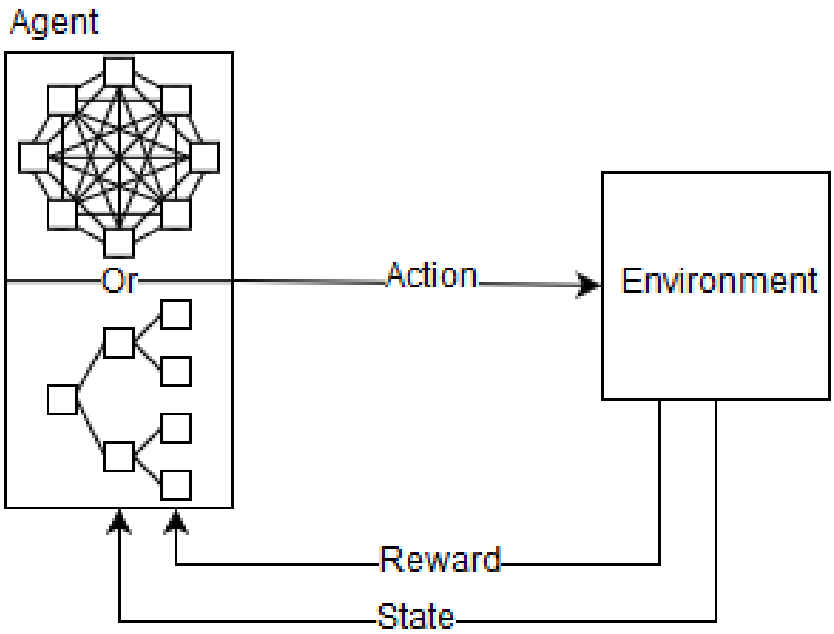

Figure 10: Online Learning Network

combining the problem specifics of the MAB problem with the potential useful properties of model solutions for other domains, such as the binary tree. We end up with a set of general network topologies that seems promising to build upon.

5 Implementation

5.1 Tools

5.1.1 Julia(https://julialang.org/)

Julia is a high-performance programming language ment for numerical computation, which can boast it self with being near to C level speed. While its syntax might resemble the ease of Python.

Julia was selected after attempts in languages such as R and Python. Al-

though, these too are a lot more used in the types of programming this thesis concerns itself with. They both have their downsides. R which was primarily used before the switch to Julia, is incredible slow. Especially so with loops. For the most part, specialised vectorized function are used instead of loops. These functions are essentially typical higher order functional programming substitutes for Map, and Fold. I found it clunky to use these as most of the program is easiest written by the use of indexing, which is also more intuitive with loops.

Python shares the slowness of R. Although there are packages and workaround solutions for this, it is teadesom and too much out of scope of this thesis. I saw promise in Julia, as I did not need R or Pythons strongest card, namely, their packages. Since I am sampling relatively large numbers to create an average, the speed of Julia comes in handy. Last but not least, Julia seeks to solve the two language problem, where algorithmic solutions are often written first in a easy to write language, such as R or Python, but then has to be rewritten in a faster language later, like C or Fortran. Since Julia is meant to cover both these needs, it means that refactoring this thesis module is a easy step, should the Module produce something worth using in production.

5.2 Module Implementation

The whole practical implementation of this thesis were conducted through the creation of a module in Julia, after initial experiments where done in R. The different aspects were separated in functions which can be called through each other in order to create the plots in this paper. These aspects are: sampling, measure, regret, algorithm and bandits.

Sampling, are functions that take large numbers of parameters to submerge the agent in an environment, and repeat the whole process N amount of times. In effect these functions are made to compose a simulation and return performance of an algorithm, encapsulating all other aspects.

Measure, is class of functions especially designed to record the results of

a specific given network algorithm. Since these networks often store data in seperate result matrixes, and in different data structures(e.g, trees, flat networks etc). This function has to collect the data and transform it to a single result matrix in order to be able to measure the performance of the networks.

Regret, are functions that calculate the instantaneous and cumulative regret given the aggregated results of an algorithm in a specific environment.

Algorithm, contains all implementations of classical algorithms(UCB,E-greedy, TS), as well as the different network structures and network building functions. This is done in order so that classical algorithms can run as functions in the networks.

Bandits, these are stochastic functions that simulate the environment composed by a given bandit type.

Some parts of the code has generalized functions(e.g: regret) that are easy to reuse, but a large part of this project has involved coding network structures with obscure rules that they have to abide by, and dividing data in several data structures. This means that in many cases specialised wrapping code has to be made on top of the networked algorithm.

5.2.1 Binary/Bernoulli Bandit

A Binary Bandit is simply put, a weighted coin flip. A bandit that either returns a value of say 1, or does not. In the module, I made this by having a function draw a standard distributed random number. Then compare this number to a cut off threshold symbolizing the probability of success. E.g: If the drawn number is lesser or equal to the probability threshold, return 1. Otherwise return 0.

5.2.2 Symbiotic Bandit

A Symbiotic Bandit is a Binary Bandit, but its probability threshold is altered depending on the state of its linked bandits. E.g: Bandit 1. has a independent probability of 20%, but if its linked bandits are played before it, then it its probability is increased by 10%. The Symbiotic Bandit is represented by Binary Bandits that has 2 arrays of context information. The first array represents which bandit carries a bonus probability. The later array states which bandit the bonus probability should be passed along to.

5.3 Hierarchical Network

This network is at its core a tree structure. When a choice is being made we start at the top node and work our way down to a leaf(Bandit) then record the results and passes it upwards again. This approach is heavily inspired by Monte Carlo Tree Search which has recently been effective in playing games with deep neural nets, games such as Go and Chess (Silver, Schrittwieser, et al., 2017). All tree structures used in this paper, mimics the backpropagation step of the MCTS.

Its implementation relies on a recursive function call through branches of MAB algorithm result matrixes. Until it reaches the bottom leaf, which is a bandit. It then runs the bandit, records the result and propagates the success or failure up to the top node.

Overlapping Tree(OLTree)
The initial approach was to have architectures allowing as many overlapping branches as possible. This is highly inspired by a typical Artificial Neural Network, however, in difference to a feed forward structures, this problem does not have the same input and output scenario, and the results of a choice has to be recorded and used as a prior for further choices.

The main inspiration was the aspect of capturing relationships between bandits in the layers of the tree. As each layer captures a certain level of detail

or sub-problem. In our case that could be different MAB algorithms being selected depending on which prevailed in the scenario. Also, with all nodes being the same algorithm, there's a high likelihood that one branch outperforms the others by chance, and would then be selected for in future sequences. While in more complex bandit scenarios, where certain relationships occurs between the bandits. Such structures might encapsulate dependencies between arms better than an traditional MAB algorithm.

In figure 11 each box represents a MAB algorithm and a result matrix. While a circle is a bandit with some associated success probability.

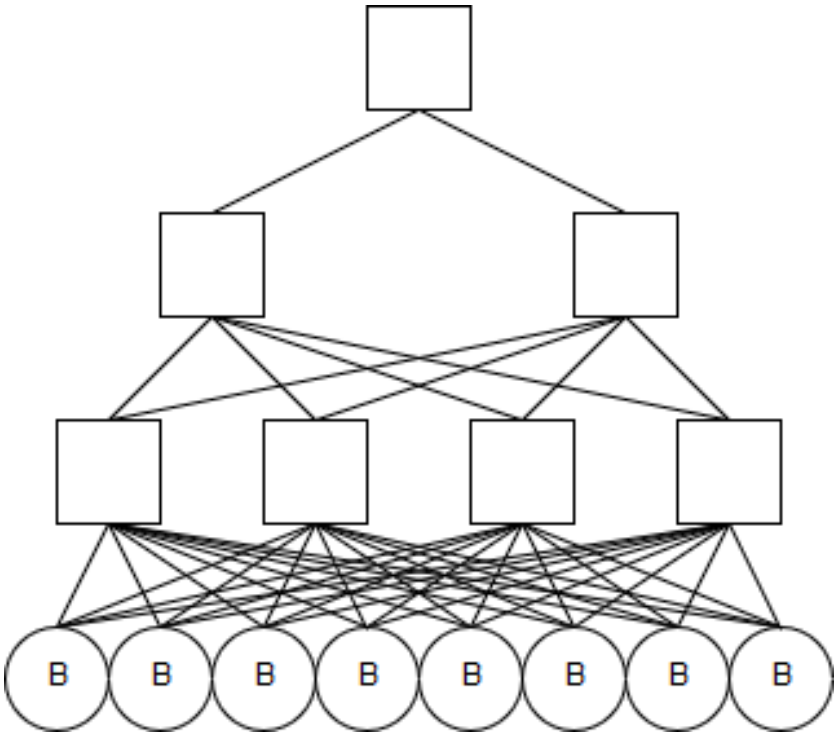

Figure 11: Overlapping Tree

One imminent issue with such a open structure is that this takes a long time to cumulate reliable information. Something of which is key in the MAB problem. Each new layer in the hierarchical obstructs the certainty of the optimal choice. In one way this is a overly explorative approach.

results, is a 2-by-n matrix containing all prior data recorded, where each

column(n) is a specific bandit. The first row(m) contains successful trials, while the second holds failed trials.
level, is an array representing an horizontal section, containing all results matrixes at the current level.
tree, a tree is an array containing all levels. **RecordReward**, pulls an arm, records the reward and updates the path of choices through backpropagation of the reward.
Policy, is the explore-exploit strategy used to make a choice over at set of arms. In this thesis I use Thompson Sampling in all experiments.

$$arms_n = \begin{bmatrix} bandit_1 & \cdots & bandit_n \end{bmatrix}$$

$$result_{m,n} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ l_{2,1} & \cdots & l_{2,n} \end{bmatrix}$$

$$level_n = \begin{bmatrix} result_1 & \cdots & result_n \end{bmatrix}$$

$$tree_n = \begin{bmatrix} level_1 & \cdots & level_n \end{bmatrix}$$

```
1: function OverLappingTree(tree, arms)
2:     if bottom level then
3:         Policy selects any bandit, at current level
4:         updatedTree ← RecordReward
5:     else
6:         Policy selects any arm, at current level
7:         run Overlapping(tree,arms), over the selected arm
8:     end if
9:     return updatedTree
10: end function
```

At each level a policy is used to select the next node at the level below, by using its current result matrix. It does this until it reaches a bandit where it plays the bandit, records the reward, and backpropagation the reward through the reward metrics used in its path.

Binary Tree(BiTree)

On the other side, with absolute limitation in mind, we have the binary MAB net. This structure is partly inspired by the notion that is made by other studies that algorithms tend to have a easier time in scenarios with small amounts of arms and scenarios with a large probability variance between the arms (Kuleshov & Precup, 2010). Then what would happen if the whole process is a series of easier choices?

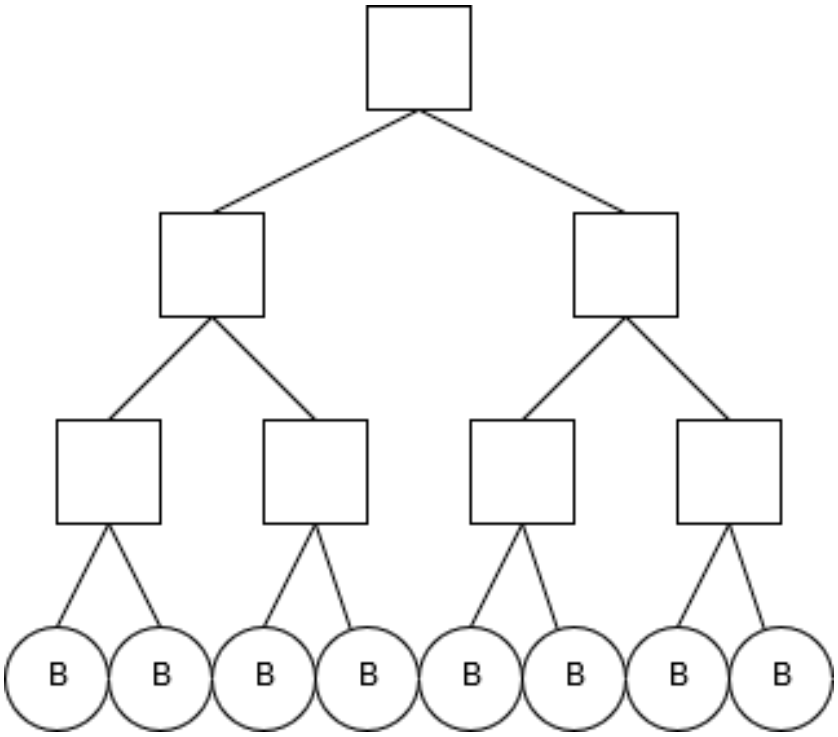

Figure 12: Binary Tree

In a way it functions as a probabilistic binary tree search, which becomes more and more certain. However, this tree is not sorted by size, and the initial order of each bandit has a significant impact on its likelihood of being played. The probability of choosing one branch becomes the average of the bandits contained in that branches. If by accident we capture the best and the worst bandit at the lowest paring. Than there is a good chance that the coupling of suboptimal arms will be preferred, as a result the optimal regret bound would never be reached.

$$arms_n = \begin{bmatrix} bandit_1 & \cdots & bandit_n \end{bmatrix}$$

$$result_{m,n} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ l_{2,1} & \cdots & l_{2,n} \end{bmatrix}$$

$$level_n = \begin{bmatrix} result_1 & \cdots & result_n \end{bmatrix}$$

$$tree_n = \begin{bmatrix} level_1 & \cdots & level_n \end{bmatrix}$$

```
1: function BinaryTree(tree, arms)
2:     if bottom level then
3:         Policy selects between 2 bandits, at current level
4:         updatedTree ← RecordReward
5:     else
6:         Policy selects between 2 arms, at current level
7:         run Overlapping(tree,arms), over the selected arm
8:     end if
9:     return updatedTree
10: end function
```

Binary Thompson Sample Sorted Tree(BiTSSortTree)

This configuration attempts to counteract the importance of the initial bandit arrangement in the binary tree. It does this by at some interval, sort the bandits by the current lowest level nodes assessment, with a Thompson Sampling algorithm. After the bandits are rearranged, information recorded by the bottom layer algorithms are traded accordingly and then propagated back up the tree.

```
1: function BiTSSortTree(tree, arms)
2:     Sort Bandits and Tree by Thompson Sampling
3:     if bottom level then
4:         Policy selects between 2 bandits, at current level
5:         updatedTree ← RecordReward
6:     else
```

7: Policy selects between 2 arms, at current level
8: run Overlapping(tree,arms), over the selected arm
9: **end if**
10: **return** updatedTree
11: **end function**

5.4 Flat Network

In flat network the algorithm operates with a current position over the bandits. This position is changed depending on the arm the algorithm chooses to pull.

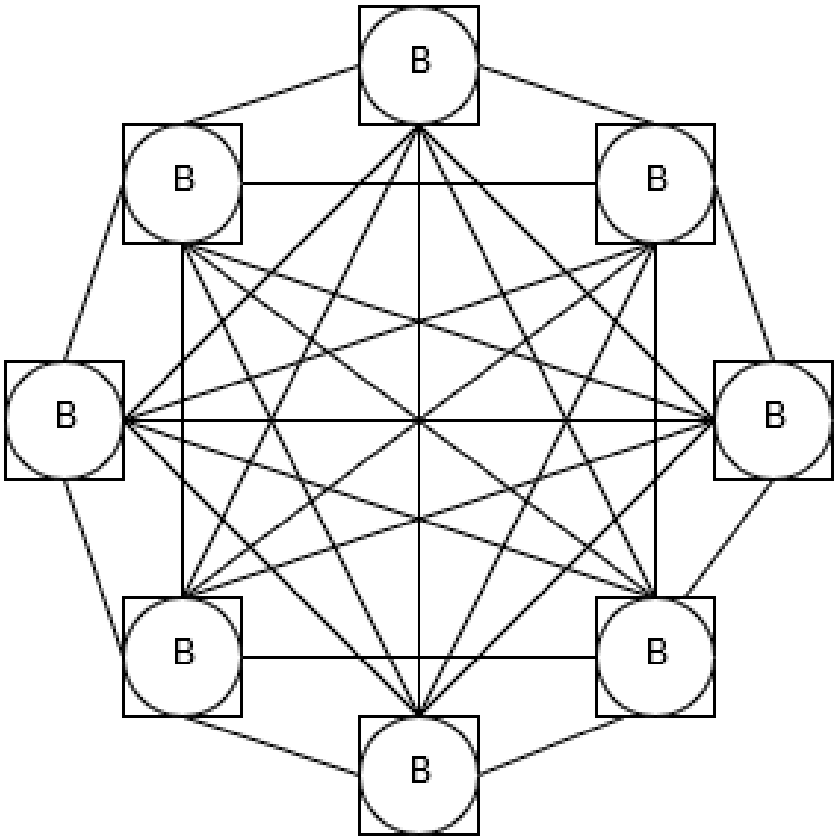

Figure 13: Clique Net

A prominent attribute here is an acquirement of local information. This

means that we now record information attributed to one specific arm, and each choice is reflected as, what is my best choice given my current position. This approach is highly similar to a markov chain as each choice is on the bases of the current position. Where the probabilities are discovered on the go. However, the more advanced versions include different ways of supplementing the current positional result matrix with global knowledge(the sum of all local result matrices). Global knowledge is essentially the same as the result matrix a single classical MAB algorithm would work on. While local knowledge is the current positional result matrix. The reason this has to be specified is that with local knowledge matrices we also gain a positional perspective of the data which is a subset of the global knowledge matrix.

In figure 13 each box is a result matrix, with an associated explore-exploit algorithm. While each circle is a Bandit. The Global knowledge Matrix is the sum of all the result matrices represented with the boxes.

Clique net

This version only considers the local information at each timestep. Each algorithmic nodes choice is independent. A common theme is that the more free of a choice or connected these networks are, the less effective they are in the short term. In effect this approach is similar to having N amount of MAB algorithms play by them self, but counting each trail as if it were one algorithm.

results, is a 2-by-n matrix containing all prior data recorded, where each column(n) is a specific bandit. The first row(m) contains successful trials, while the second holds failed trials.
resultArray, is an array containing each result matrix. Where its index is the position of the result matrixes.
indmax, return the index of an given array which hold the largest value.
position, is a integer holding the current positional state of algorithm. This number dictates which result matrix is the local knowledge matrix.
algos, is an array containing each explore-exploit algorithm associated with a result matrix.
updatedResultArray, simplified way of recording the rewards from play-

ing a Bandit.

$$results_{m,n} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ l_{2,1} & \cdots & l_{2,n} \end{bmatrix}$$

$$resultArray_m = \begin{bmatrix} results_1 & \cdots & results_n \end{bmatrix}$$

1: **function** $CliqueNet(resultArray, algos, position)$
2: $nextposition \leftarrow algos[position](weighted)$
3: **return** nextposition, updatedResultArray
4: **end function**

Local Global Network Weighted(LocGlobNetWeight)
This approach aggregates each local knowledge in to one matrix, and uses that as the baseline of choice, followed by supplementing with the current local information. The aggregation of each matrix, makes it similar to a normal single node MAB algorithm. As a result it performs accordingly, but we also have the capability to influence the choice by the dependent local matrixes. In this cases simply by adding the local and the global matrix together.

results, is a 2-by-n matrix containing all prior data recorded, where each column(n) is a specific bandit. The first row(m) contains successful trials, while the second holds failed trials.
resultArray, is an array containing each result matrix. Where its index is the position of the result matrixes.
indmax, return the index of an given array which hold the largest value.
position, is a integer holding the current positional state of algorithm. This number dictates which result matrix is the local knowledge matrix.
algos, is an array containing each explore-exploit algorithm associated with a result matrix.
updatedResultArray, simplified way of recording the rewards from playing a Bandit.

$$results_{m,n} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ l_{2,1} & \cdots & l_{2,n} \end{bmatrix}$$

$$resultArray_m = \begin{bmatrix} results_1 & \cdots & results_n \end{bmatrix}$$

1: **function** $LocGlobWeight(resultArray, algos, position)$
2: $collective \leftarrow sum(resultArray) - length(resultArray) + 1$
3: $weighted \leftarrow collective + resultArray[position]$
4: $nextposition \leftarrow algos[position](weighted)$
5: **return** nextposition, updatedResultArray
6: **end function**

Local Global Net Relative Weight(LocGlobNetRelW)
Relative weight is meant by the entrywise product of the global knowledge matrix and the current local one. In this way the current local matrix has a significant impact on with action is selected next. While still maintaining a relative scale between the arms.

results, is a 2-by-n matrix containing all prior data recorded, where each column(n) is a specific bandit. The first row(m) contains successful trials, while the second holds failed trials.
resultArray, is an array containing each result matrix. Where its index is the position of the result matrixes.
indmax, return the index of an given array which hold the largest value.
position, is a integer holding the current positional state of algorithm. This number dictates which result matrix is the local knowledge matrix.
algos, is an array containing each explore-exploit algorithm associated with a result matrix.
updatedResultArray, simplified way of recording the rewards from playing a Bandit.

$$results_{m,n} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ l_{2,1} & \cdots & l_{2,n} \end{bmatrix}$$

$$resultArray_m = \begin{bmatrix} results_1 & \cdots & results_n \end{bmatrix}$$

1: **function** $LocGlobRelWeight(resultArray, algos, position)$
2: $collective \leftarrow sum(resultArray) - length(resultArray) + 1$
3: $weighted \leftarrow collective. * resultArray[position]$
4: $nextposition \leftarrow algos[position](weighted)$
5: **return** nextposition, updatedResultArray
6: **end function**

Local Global Net Thompson Sampling(LocGlobNetTS)
Although Thompson Sampling is the main explore-exploit algorithm used in all of the compared networked algorithms, this version uses it slightly differently. This version draws a random sample from the beta curve form both the global knowledge matrix as well as the current local knowledge matrix. Then creates a entrywise product of each corresponding arm.

results, is a 2-by-n matrix containing all prior data recorded, where each column(n) is a specific bandit. The first row(m) contains successful trials, while the second holds failed trials.
resultArray, is an array containing each result matrix. Where its index is the position of the result matrixes.
indmax, return the index of an given array which hold the largest value.
position, is a integer holding the current positional state of algorithm. This number dictates which result matrix is the local knowledge matrix.
algos, is an array containing each explore-exploit algorithm associated with a result matrix.
updatedResultArray, simplified way of recording the rewards from playing a Bandit.
randomSamples, the act of drawing random samples form a beta curve made from its result matrix input. Especially this thompson Sampling without selecting the maximum output.

$$results_{m,n} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ l_{2,1} & \cdots & l_{2,n} \end{bmatrix}$$

$$resultArray_m = \begin{bmatrix} results_1 & \cdots & results_n \end{bmatrix}$$

1: **function** $LocGlobTS(resultArray, algos, position)$
2: $collective \leftarrow randomSamples(sum(resultArray) - length(resultArray) + 1)$
3: $weighted \leftarrow randomSamples(resultArray[position])$
4: $nextposition \leftarrow indmax(weighted. * collective)$
5: **return** nextposition,updatedResultArray
6: **end function**

6 Results

The results are divided in two categories, the Bernoulli Bandit, and its extension the Symbiotic Bandit. Initially all algorithms are shown in 3 different scales: 8, 32 and 128 arms, to showcase how the performance of the algorithms scale in the lights of increasing number of arms.

Under the comparison section, I dive in more depth with regards to trial length and differences of property between the more promising algorithms. The plots are all instantaneous regret measurements unless expressed otherwise, where the general goal is to reach zero regret. In all name labels of each algorithm, you will see a number reflecting the total cumulative regret over the same interval displayed in the plot. The lower this number is, the better the algorithm has performed.

6.1 Bernoulli Bandit

The figure 14 shows how Bayes(Thompson Sampling) can be considered objectively better than the two other policies(Epsilon Greedy and Upper Confidence Bound). The reason it can be considered generally better is because of the combination of the two following metrics, keeping in mind that the length of the experiment is an arbitrary pick, purly to show the lower end

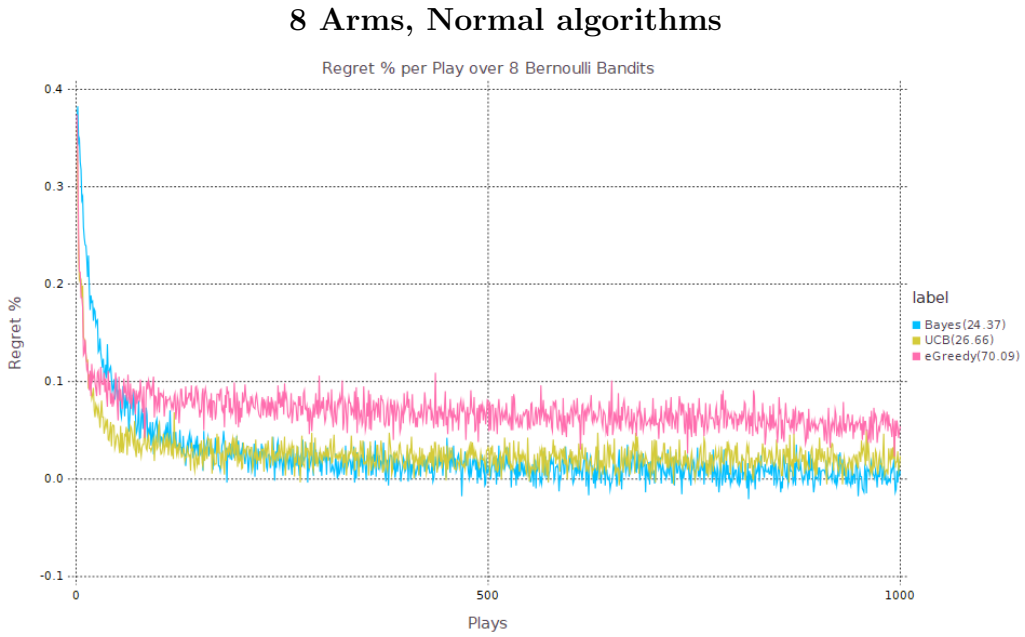

Figure 14: 8 Arms, Normal Algorithms

of the algorithms. The first metrics, is that it has the lowest cumulative regret at the end of the trial interval(24.37), meaning that in the case of 1000 timesteps its policy achieved the highest reward over investment. Secondly, it reaches a zero regret state first, which means that any future timesteps is purely beneficial, and will only grant a better cumulative lead relative to the others.

When we compare MAB algorithms, the typical general notion is that we want a zero regret policy, as they can be guaranteed to find the optimal arm, if given enough trials. However, given the right scenario, this might not be necessarily the best solution. In more practical applications, the amount of trials might be much shorter, and one might be more interested in finding a relatively good arm fast rather than the best arm overall. What follows is that one might just be interested in more greedy algorithms rather than algorithms with a better balance. This means that when we consider how good an algorithm performs, it can be from a contextual vantage point which considers the length of the interval.

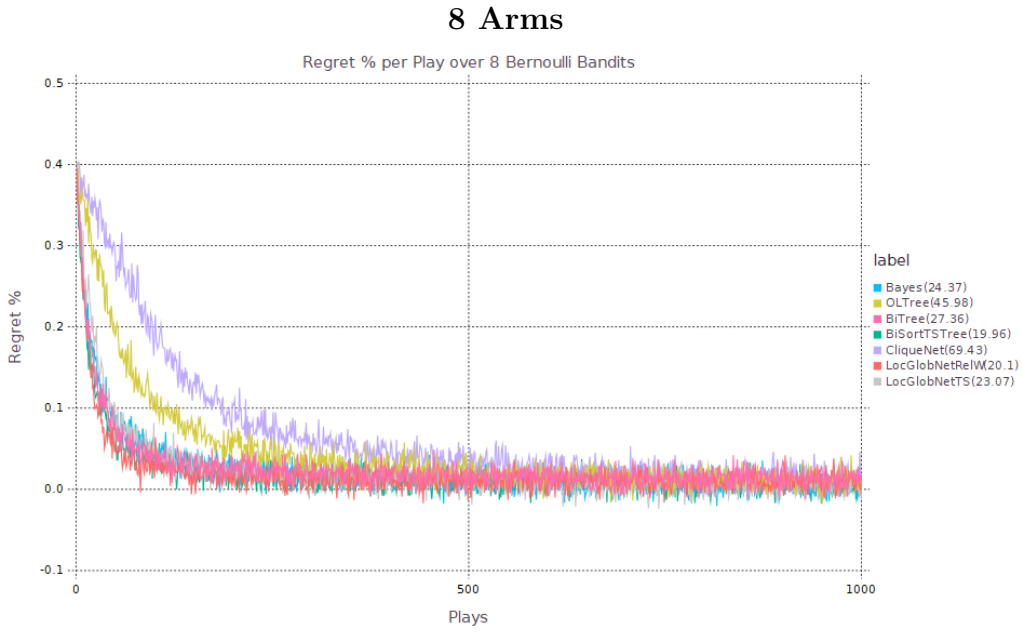

Figure 15: 8 Arms, Bernoulli Bandit

In the plots 17, 16 and 15 OverLappingTree and CliqueNet, are easily seen as outliers as they are too explorative to compete. Something which becomes even more clear at larger scales. On the other hand we see the two greediest policies are the BiTree, and its sorted version BiSortTSTree. In the middle we find LocGlobNetRelW and LocGlobNetTS, both with a slight regret variation from the single bayesian(TS) algorithm. One noticeable trait of the LocGlobNetRelW configuration is that it seems to always score among the lowest regret wise regardless of arms scale.

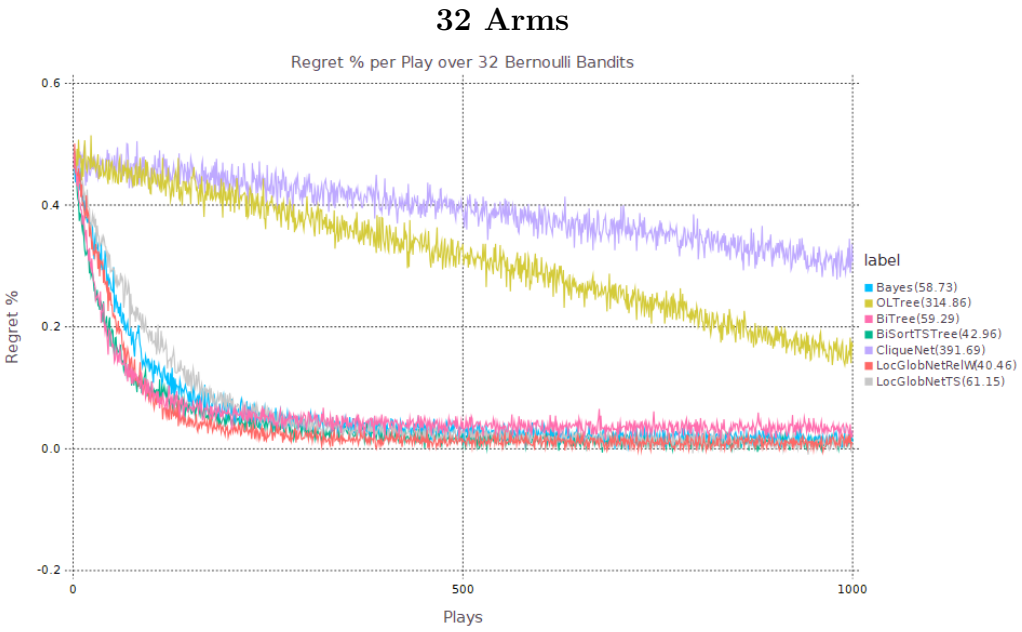

Figure 16: 32 Arms, Bernoulli Bandit

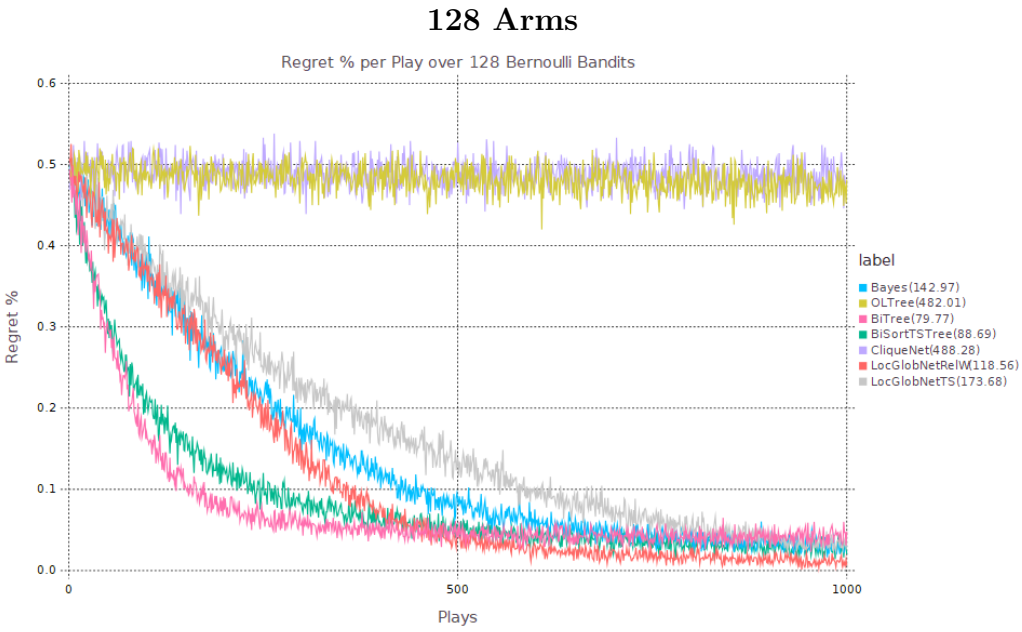

Figure 17: 128 Arms, Bernoulli Bandit

6.2 Symbiotic Bandit

In the Symbiotic Bandit, algorithms can score lower than 0 regret. This is apparent in the following plots 18, 19 and 20, were the independently best arm is taken as a faux optimal choice. It should be noted that the regret could just as well be calculated from the theoretical optimal dependent sequence of arms. The reason this is not done, is to showcase how much the algorithms that does not take advantage of elusive feature exploitation(EFE), still by luck benefit from a dependent environment. What becomes even more clear then, is the discrepancy between algorithms that randomly pass by elusive features, and the ones that actively benefiting from them. The down side of this displayal is that we cannot see how far away the EFE algorithms are from reaching the theoretical optimal regret through the dependent arms. However interesting this knowledge could have been, it is somewhat besides the point of this thesis, as the EFE algorithms are simply supposed to be normal algorithms with a slight advantage in dependent scenarios. It should also be reiterated that the goal of this thesis is simply to put down cornerstones for further advancements.

In low arm scales CliqueNet can find optimal symbiotic solutions, while OverLappingTree will not. However, at larger scales it is also obvious that both of these methods alone are too explorative. In the middle ground we find the two tree structures. They are not intended to find symbiotic relationships. And seems to perform along the lines of a single node bayesian(TS) algorithm, if not slightly more greedy. It should be noted that on 128 arms scale, there are too many options, to reliable find the symbiotic link within a 1000 trials. As we can see some algorithms are still improving at that point. Overall LocGlobRelW seems to perform best, in that it reaches a low instantatiouse regret quickly by taking advantage of the dependent arms. However, it appears that LocGlobTS might have reach a better instantaneous regret if the experiment goes on long enough.

8 Arms

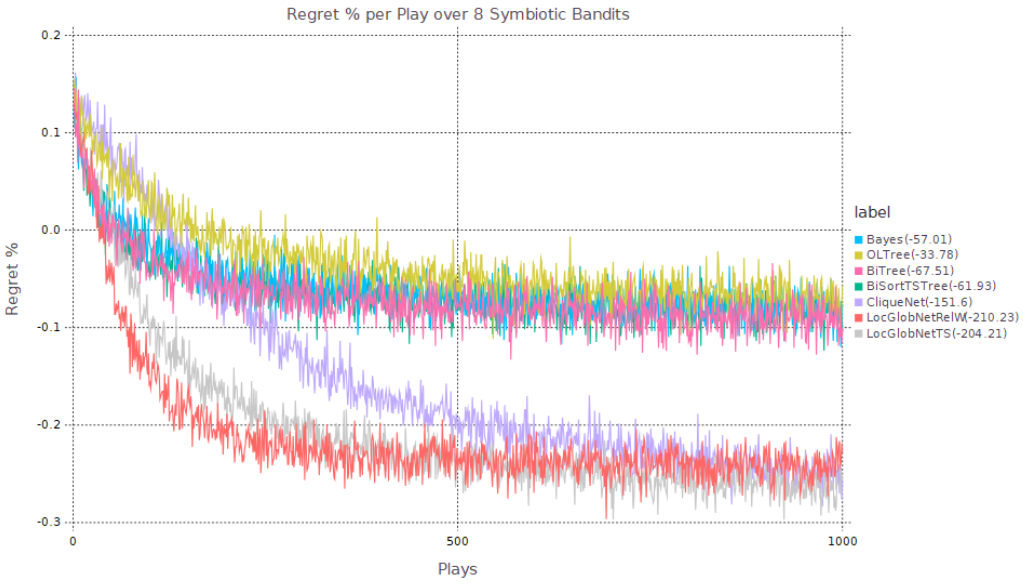

Figure 18: 8 Arms, Symbiotic Bandit

32 Arms

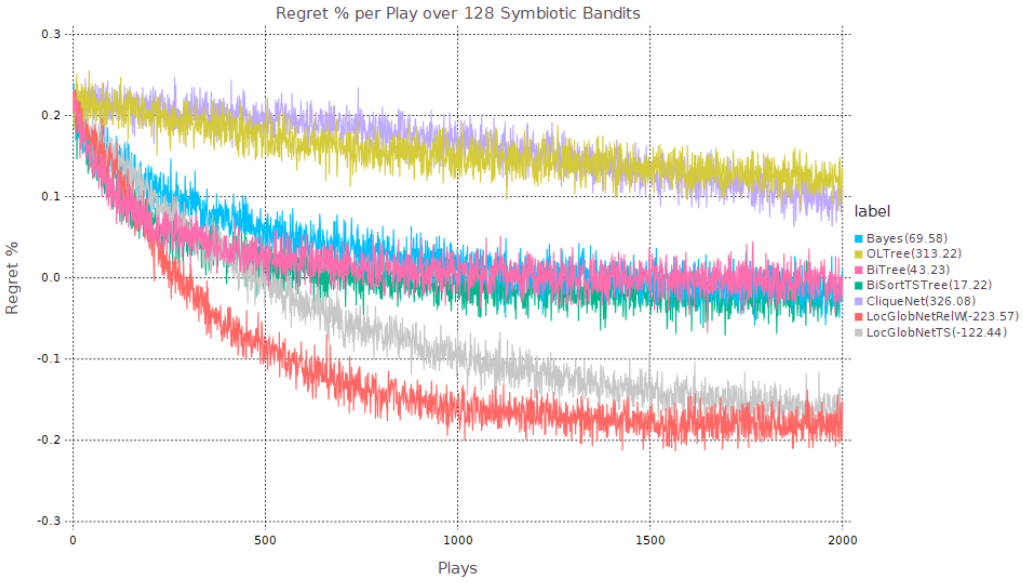

Figure 19: 32 Arms, Symbiotic Bandit

Figure 20: 128 Arm, Symbiotic Bandit

6.3 Comparison

Some of initial plots does not justify, or simply makes it hard to tell the differences between some of the algorithms. The following comparisons are meant to shed light on traits not captured in the previous viewed general plots.

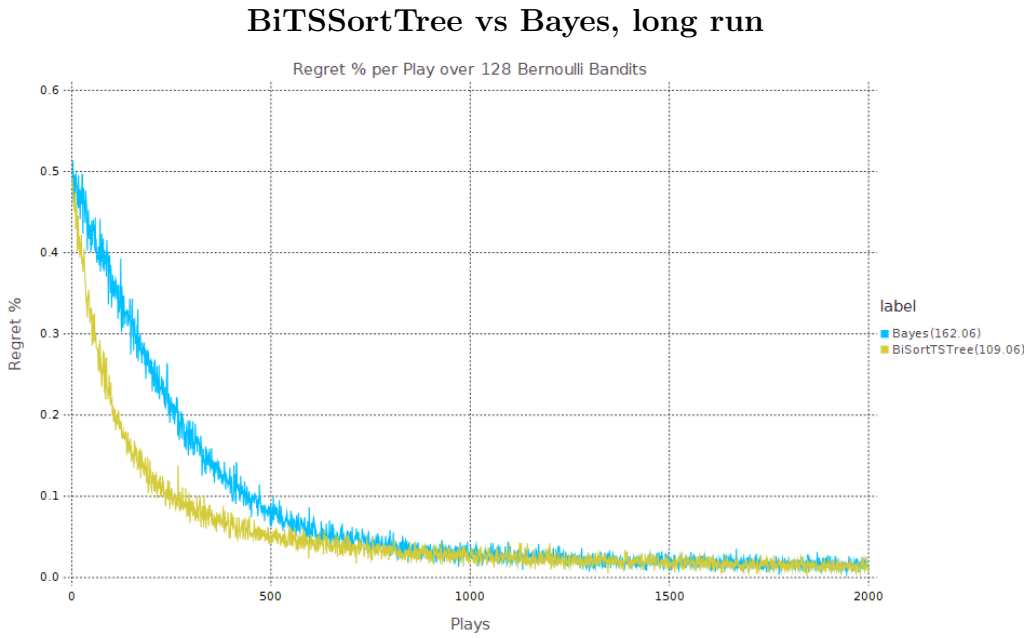

Figure 21: BiTSSortTree vs Bayes, long run

Binary Thompson Sampled Sorted Tree seems to have on average a better performance on all arm scales, but it become especially significant on large amounts of arms. Seen in this figure 21, the difference is mostly in the initial stages of the run. Where it has a much greedier approach then a single Thompson Sampling algorithm, at later timesteps they both seems to converge to the same regret floor.

As seen in figure 23, in the Symbiotic Bandit scenario LocGlobNetTS has a better instantaneous regret in the end, while at that point LocGlobNetRelW does not seem to improve at any significant rate.

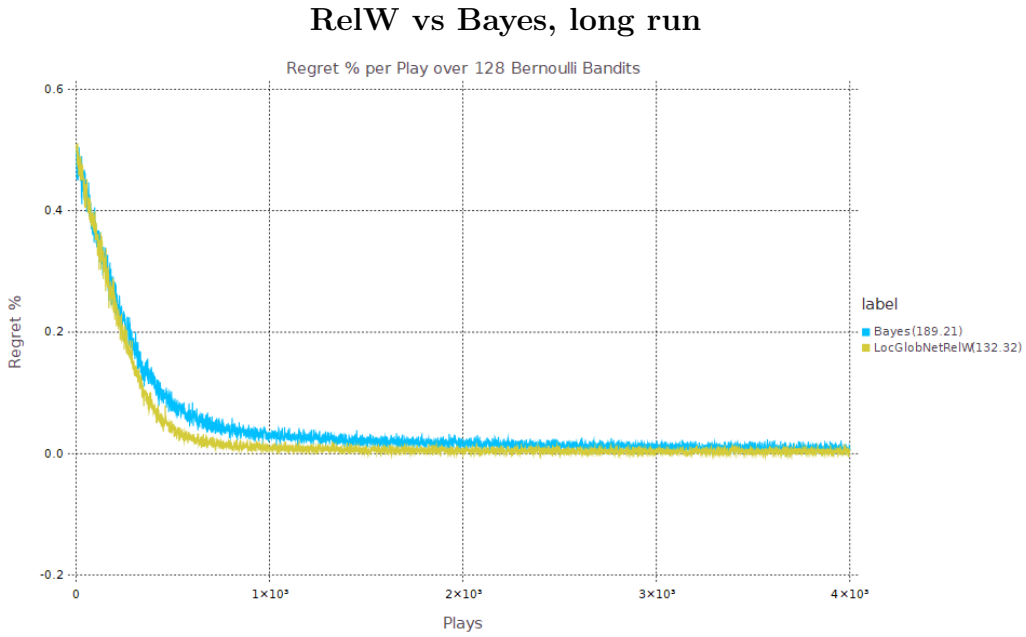

Figure 22: RelW vs Bayes, long run

6.3.1 Other notes

Square Bayes

It should be noted that there are other ways of tuning off-policy explore-exploit parameters of an algorithm, on a more general level that does not use networks. One way is to square the data before the algorithm run over it. The result is that any minor difference of data is exaggerate to the scale of an exponent, something which seems to do well in the Thompson Sampling policy. This is also what we do to some degree in the networks that use global and local result matrices. It is not exactly exponential, but when a global result matrix is entrywise multiplied with a local one, the weight of the local matrix dictates the differences between the arms.

Its approach seems to be somewhat similar to that of the binary TS sorted tree regret wise. Where both approaches are rather greedy initially, and seems to reach the same regret floor as normal Thompson Sampling.

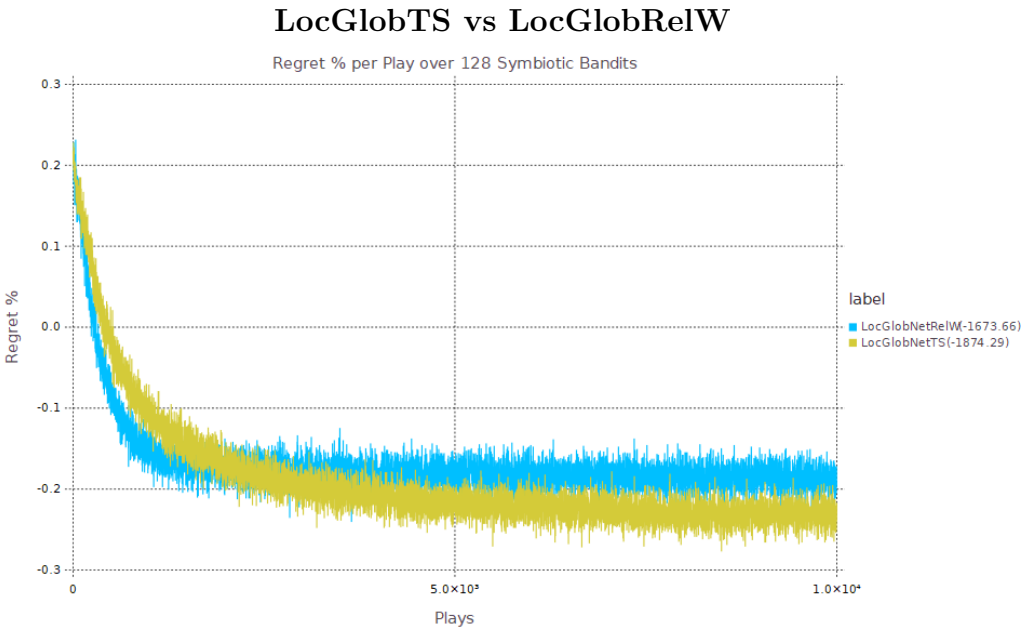

Figure 23: LocGlobTS vs LocGlobRelW

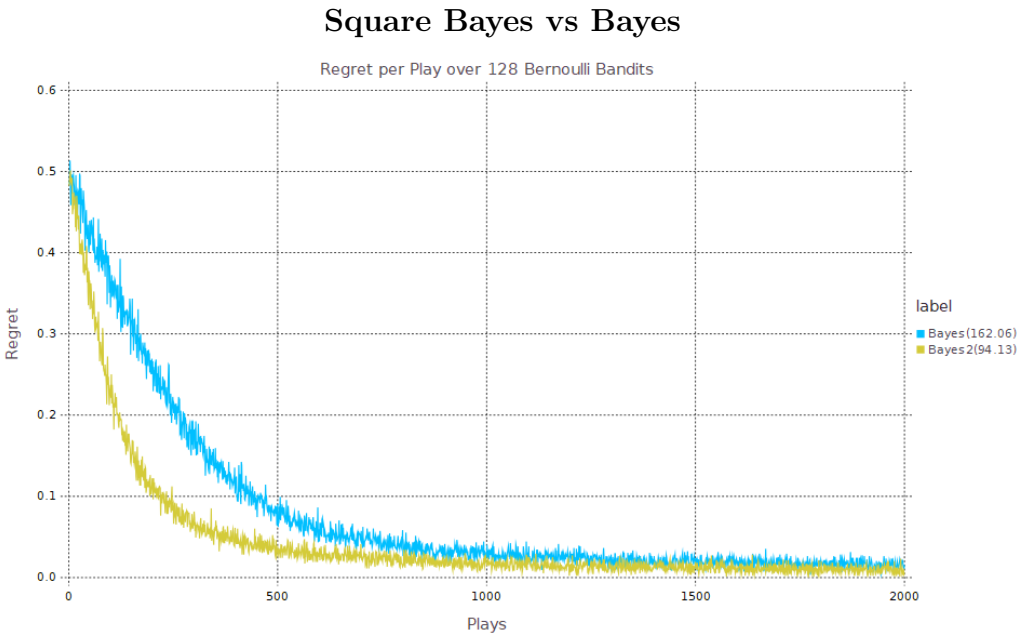

Figure 24: Square Bayes vs Bayes

7 Discussion, Conclusion, Future Work

7.1 Discussion

7.1.1 Explore vs Exploit revisited

In many ways a networked structure is simply put just a more complicated way of tuning a hypothetical explorative/exploitative parameter on the go. Either by focusing on certain subset of the total data at the right time(LocGlobRelW), or dividing the total amount of comparative data in to smaller and smaller branches(BiTSSortTree). Fundamentally, what these structures are doing, is to exaggerate the differences between current data by a set scale.

This can be done by a lot simpler measures, such as squaring the input if you want the algorithm to behave exponentially more greedy. Which is shown in figure 24 that its behavior, regret wise, is somewhat similar to a binary sorted tree. But it should be noted that this is a very general way of looking at the input data. Which means that that it can be applied to any algorithm. One proverbial flag has to be raised as follows when considering networked structures regarding what the purpose of such an endeavor is. The fact is that there are more efficient ways of scaling explore-exploit differences throughout the run. That relies on more fundamental mathematical simplicity, and is often what is attempted in most explore-exploit algorithms.

7.1.2 Improvement over large pools of arms

Generally it appears that some networked MAB algorithms can achieve a better choice of exploration. In other words, the explorative approach is conducted with more nuance, such that rather then randomly selecting from any arm within some probability distribution. This distribution segregates good explorative candidates from those who are not so much so. The result is as we have seen more impact full on scenarios of many arms, but the more

arms are introduced, the more memory and processing power is necessary. Especially, so in the case of networked algorithms.

7.1.3 Information Leveraging, Branching Factors

I experimented mostly with branching factors in the tree based approaches, and information leveraging mostly in the flat net approaches. It should be noted that experimenting with both in both approaches could have given much more interesting behaviors. Where the tree based structures could assimilate more complex behaviors motivated by connectionist approaches, rather than binary search and decision trees alone. This also suggests that the flat networks could have possibly achieved ways of reducing the search field.

7.2 Conclusion

Research question conclusions:

1. Can network based MAB models achieve lower regret than traditional solutions(TS) in the Bernoulli Bandit(independent arms)?
2. Can network based MAB models achieve lower regret than its single node counterpart on dependent arm MAB scenarios(Symbiotic Bandit).
3. Are there networked solutions that can reduce regret compared to a single node, in both scenarios, Bernoulli and Symbiotic Bandit.

To answer **RQ1**, we will have to look at networks that has a lower cumulative regret then a normal TS, as well as the capability to reach zero regret. It turns out that there are actually topologies that can improve the normal Thompson Sampling method, where both tree and flat networks has candidates. In the results we see that both BiSortTSTree and LocGlobNetRelW scores a lower cumulative regret, as well as converges to zero regret either before or at the same rate as normal TS.

RQ2, of the topologies presented in this paper, there are only a few from the flat network approach, that can be considered elusive feature exploiting networks. The networks are the LocGlobNetRelW and LocGlobNetTS, which both are able to separate them self from the other algorithms. CliqueNet also will eventually reach less than zero regret, but it does so at such a slow rate that in can not be considered competitive.

RQ3, to this question there is only one network topology capable of answering. That network is the LocGlobNetRelW, which is capable of outperforming normal TS in the Bernoulli Bandit in all arms scales, as well as taking advantage of the Symbiotic Bandit scenario.

In this thesis I have proposed a set of networked structures built out of explore-exploit algorithms, in an attempt to introduce methods to further tweak explore-exploit parameters and enquire traits overlooked by assumptions made in standard algorithms. Many of the explored structures seems to perform better than its single node counterpart, especially on scenarios with large numbers of arms, but it should also be noted that they are significantly heavier to run, both memorywise and computationally.

As an extension of prior bandit scenarios, I have also introduced a bandit type called Symbiotic Bandit, in an attempt to mimic a scenario where some bandits share some common abstract feature, so that when put in the same scope it effects their presumed probabilities of success. The results suggests that, in a theoretical setting, certain algorithms can take advantage of these relationships, while still maintaining a state of the art performance in the Bernoulli Bandit scenario.

It is important to emphasize that the different topological networks proposed, are in no way meant to be optimal, but rather to showcase suggestions for topological properties through information leveraging or branching factors. The best results using information leveraging is found in the LocGlobRelW configuration, where small subsets of the total data associated with a single node, is used as weights by multiplying it with the networks cumulative total data, before it is passed through a normal Thompson Sampler.

As for branching factors, we have seen in the BiTSSortTree that a sequence of binary arm selections can drastically improve early gains through a more delicate ordered exploration. This limitation introduced by binary search does however also introduce an assumed order of the arms, of course this is problematic because a core part of the problem is that the agents know nothing about which arm is better. BiTSSortTree has as a result a Thompson Sampling sort function that iteratively tries to estimate the correct order of the arms.

7.3 Future Work

The process of exploratory studies should open a field of interesting questions, which one could have only come to bare with through the initial exploratory research. This can in many ways be considered as actual results of the research.

7.3.1 Speed and Applicability

Although some of these structures can perform better than a single node MAB algorithm, regret wise. A flag should be raised in concern with how applicable they are in terms of performance. This thesis has not laid any emphasis on computational complexity, or how feasible these structures are in a real world scenario. The practical design of the code was intended to be as malleable as possible rather than speed orientated. So that different network topologies could easily be tested. The result of this lead to a code that is far from optimized, and in many ways probably exhibits unnecessary memory use. One such trait is the capability of using a heterogeneous pool of explor-exploit algorithm. This trait was not explored too extensively as early attempt indicated that some algorithms where simply better suited. As the networked versions usually just amplifies the traits of the single nodes.

It should be noted, however that although the general notion is that networked algorithms, if configured correct, can significantly outperform single

node algorithms when one increase the amount of arms of the scenario. It also significantly increase the computational efforts. Further investigation in this area would be necessary to prove the usefulness of networked MAB algorithms in real world scenarios.

It could be that although the networked algorithms are more hardware requiring, as noted by the differences in run time speed in the experiments depending on arm scale. However, it could be that these difference are not that significant as the algorithms might not be run as often in real world scenarios, when considering batch updating. Nonetheless, the algorithms are far from optimized, and in some cases there might exist simpler methods to achieve the same results, we can consider Squared TS vs binary TS sorted tree as an example(figure 24 21).

7.3.2 Batch updating, test

Although I have argued that Thompson Sampling fits well as it is suited for batch updated bandit scenarios. As well as the fact that networked versions might be too heavy duty for online(continuous) updating scenarios. I did not perform any experiment with batch updating. Due to the fact that this only becoming apparent in hindsight after the design of the experiments and the code. Most of the algorithms are designed so that they relay on a instantaneous feedback before it proceeds the experiments. Seeing as this version of the problem is likely very applicable I would have liked to proceed with testing differences between the behaviours of the networked algorithms vs normal Thompson Sampling in a batch updated scenario. The initial advantage of a probabilistic policy such as Thompson Sampling in a batch pull updating scenario is that the distribution of arm pull will match its current certainty. While a deterministic approach such as Upper Confidence Bound will invest all in one arm unless further tricks are made. The question is then how is this distribution of pulls in a batch update performed under a tree based method, or a flat network?

7.3.3 Dynamic Branching

Some structures has been constrained to ease the implementation process. The archetype for this problem is the binary tree structures, which in this study has been limited to arms scales with a binary nature(2,4,8,16...). This means that it can not be used in any situation where the total number of arms does not scale binary. Further constraints involve the fact that adding new arms would imply some logical way of extending the tree, which in this case is not done. There is also a concern regarding whether this can be done while still maintaining properties that would make these kinds of structures worthwhile.

One approach could be to have a binary structure at the top of the tree. Where the bottom branches can contain many leafs(bandit arms). One would still achieve the efficiency of a binary tree. And since we are sorting back-propagating rewards up the tree, there should be little difference, other than a slightly less efficient search at the bottom layer. The rest of the tree would still cut the search space in half at each layer.

7.3.4 Different scales of greediness, e.g: binary tree

Initially the idea was to have different MAB algorithms at different nodes of a network. Much like some Artificial Neural Networks has different layers of different activation functions. This idea was dismissed as given a specific MAB scenario with a certain length one MAB algorithm tends to be better than others, without any special properties worth considering. Therefor a heterogeneous mix did only reduce its global potential. However, one version that might have been interesting in hindsight. Would have been a tree based structure where greediness is altered between the layers. E.g: The top node is excessively explorative while the bottom layers are increasingly exploitive, or vice versa.

7.3.5 MCTS MAB Networks between layers

Some of these version could be extended to function between the layers of the Monte Carlo Tree Search, where the arms are fixed. Seeing as its performance in areas of large branching factor could be potentially further improved. Results from this thesis suggest that there are better solutions of the MAB algorithms for high branching factors. And would suggest Binary TS sorted Tree for cases where arms can be split binary, and quick instentainouse regret is important. Whereas the best cumulative version over long runs would be LocGlobRelW. As it acts accordingly to a single node Thompson Sampling method up to a certain point where it improves. Overall scoring a better cumulative regret. Seeing as these are relatively heavy to run. Simply running Squared TS seems to be an improvement in cases of many branches.

7.3.6 MCTS Backpropagation

The flat networks can be considered as an altered version of the MCTS, with only one step backpropagation in a recurrent state space. What would happen if this backpropagation where to go further back then simply the initial play step? This would probably have to contain som way of reducing the value being propagated between each node, and would probably also have to have a set max depth. Sending waves of information back through the network. However this would probably also increase the time complexity, and online learning would be even less feasible. Future implementations should also investigate in a more direct implementation of the MCTS in dependent arm problems. Especially with the Symbiotic Bandit in mind, or other dependent arm bandit scenarios, where sequences between arms can be explored.

7.3.7 Symbiotic Bandit Algorithms in real world

Any bandit algorithm is build upon an assumption about its environment, this goes for the solutions to the Symbiotic Bandit also. Even though the

networked approaches was intended to perform on the Bernoulli Bandit as well, but the question still stands; how would they act in the real world? Would its inbuilt assumptions prevail single node Thompson Sampling in a web based news page? Assuming that it can be computationally feasible, would the assumptions regarding relationships between arms make for spurious predictions in a highly stochastic environment?

In the thesis experiments I only showed an example where 2 bandits are linked together, but what inspired this altered version of the Bernoulli Bandit, namly a news page with articles as arms, would likely have unthinkable many relationships affecting there probability of getting played. There would also be a lot of potential insignificant noise, and other structures of the webpage that could affect these probabilities. It could also be that these symbiotic relationships are not as influential as this theses experiments assumes.

7.3.8 Reward Variance

Bernoulli Bandit is a very specific type of bandit scenario, which is especially limited in comparison to most other versions. It should be noted that networked structures might not work as well in other cases. In the Bernoulli Bandit reward variance is omitted, as there is no distribution governing the size of the reward. There is simply a stationary reward or no reward at all. (Kuleshov & Precup, 2010), stresses the importance of the difference in reward variance distributions. This could have an effect on the efficiency of the networked structures. As there are more uncertainty at play.

References

Agrawal, S., & Goyal, N. (2011). Analysis of Thompson Sampling for the multi-armed bandit problem. *arXiv preprint arXiv:1111.1797*, 1–21. Retrieved from http://arxiv.org/abs/1111.1797 doi: arXiv:1111.1797

Allesiardo, R., Feraud, R., & Bouneffouf, D. (2014). A Neural Networks Committee for the Contextual Bandit Problem. doi: 10.1007/978-3-319-12637-1

Bai, A., Wu, F., Zhang, Z., & Chen. (2014). Thompson Sampling based Monte-Carlo Planning in POMDPs. *Icaps*, 29–37.

Carroll, S., & Goodstein, D. (2009). Defining the scientific method. *Nature Methods*, *6*(4), 237. Retrieved from https://www.nature.com/articles/nmeth0409-237 doi: 10.1038/nmeth0409-237

Chapelle, O., & Li, L. (2011). An Empirical Evaluation of Thompson Sampling. *Advances in Neural Information Processing Systems*, 2249—-2257. Retrieved from https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/thompson.pdf

Chaslot, G., Bakkes, S., Szita, I., & Spronck, P. (2008). Monte-Carlo Tree Search: A New Framework for Game AI. *Aiide*, 216–217. Retrieved from http://www.aaai.org/Papers/AIIDE/2008/AIIDE08-036.pdf

Féraud, R., Allesiardo, R., Urvoy, T., & Clérot, F. (2015). Decision Tree Algorithms for the Contextual Bandit Problem. *arXiv:1504.06952 [cs]*. Retrieved from http://arxiv.org/abs/1504.06952{\%}5Cnhttp://www.arxiv.org/pdf/1504.06952.pdf

Hellevik, O. (1999). *Forskningmetode i sosiologi og statsvitenskap*. Universitetsforlaget AS 1999.

Jamieson, K., & Nowak, R. (2014). Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *2014 48th annual conference on information sciences and systems, ciss 2014.* doi: 10.1109/CISS.2014.6814096

Kuleshov, V., & Precup, D. (2010). Algorithms for the multi-armed ban-

dit problem. *Journal of Machine Learning*, *1*, 1–32. Retrieved from https://arxiv.org/abs/1402.6028

Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource Management with Deep Reinforcement Learning. *HotNets*, 50–56. Retrieved from http://people.csail.mit.edu/hongzi/content/publications/DeepRM-HotNets16.pdf doi: 10.1145/3005745.3005750

May, B. C., Korda, N., Lee, A., & Leslie, D. S. (2012). Optimistic bayesian sampling in contextual-bandit problems. *Journal of Machine Learning Research*, *13*, 2069–2106. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84864939787{\&}partnerID=tZOtx3y1

Pandey, S., Chakrabarti, D., & Agarwal, D. (2007). Multi-armed bandit problems with dependent arms. In *Proceedings of the 24th international conference on machine learning - icml '07* (pp. 721–728). Retrieved from http://portal.acm.org/citation.cfm?doid=1273496.1273587 doi: 10.1145/1273496.1273587

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, *58*(5), 527–536. Retrieved from http://www.ams.org/journals/bull/1952-58-05/S0002-9904-1952-09620-8/home.html doi: 10.1090/S0002-9904-1952-09620-8

Rumelhart, D. E., Hinton, G. E., & Mcclelland, J. L. (1986). *A General framework for Parallel Distributed Processing.*

Scott, S. L. (2010). A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, *26*(6), 639–658. Retrieved from http://www.economics.uci.edu/{~}ivan/asmb.874.pdf doi: 10.1002/asmb.874

Shahrampour, S., Noshad, M., & Tarokh, V. (2017). On Sequential Elimination Algorithms for Best-Arm Identification in Multi-Armed Bandits. *IEEE Transactions on Signal Processing*, *65*(16), 4281–4292. doi: 10.1109/TSP.2017.2706192

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A.,

... Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. Retrieved from https://arxiv.org/abs/1712.01815 doi: 10.1002/acn3.501

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. Retrieved from http://www.nature.com/doifinder/10.1038/nature24270 doi: 10.1038/nature24270

Sutton, R. S., & Barto, A. G. (1998). Introduction to Reinforcement Learning. *Learning*, *4*(1996), 1–5. Retrieved from http://dl.acm.org/citation.cfm?id=551283 doi: 10.1.1.32.7692

Thompson, W. R. (1933). On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, *25*(3/4), 285. Retrieved from http://www.jstor.org/stable/2332286?origin=crossref doi: 10.2307/2332286

Wang, Q., Zeng, C., Zhou, W., Li, T., Shwartz, L., & Grabarnik, G. Y. (2017). Online Interactive Collaborative Filtering Using Multi-Armed Bandit with Dependent Arms. Retrieved from http://arxiv.org/abs/1708.03058

Whittle, P. (1988). Restless bandits: activity allocation in a changing world. *Journal of Applied Probability*, *25*(A), 287–298. Retrieved from https://www.cambridge.org/core/product/identifier/S0021900200040420/type/journal{_}article doi: 10.2307/3214163