The Network Bandit Problem

Alex Jones

October 13, 2016

Abstract

We consider a variation of the multi-arm bandit problem, called the network bandit problem, which generalizes bandit-type play over unknown distributions by playing over a network of arms $\mathcal{A}$. Literature review reveals an unexplored problem space, motivating generalization of bandit-style play to *network bandit-style play*. To explore this, we re-examine the nature of regret for a special class of budgeted online learning problems with limited priors under two assumptions: not all user play sequences are a valid solution to the bandit; and the cost paid to play a sequence depends upon the edge traversal count on the network bandits walk W_T and is charged against the rewards gathered. Analysis demonstrates that the oracle finds *non bandit-looking*, or degenerate, solutions when the play horizon is too small. We also describe a natural sufficient condition for limiting solutions of the oracle's play sequence to be stationary. We provide a simple algorithm with analysis of regret.

1 Preliminaries

The network bandit problem is concerned with maximizing the performance of an agent, or player, who pulls the arms of a bandit over a finite horizon and can switch between arms according to a given network. The arm network $\mathcal{A}$ can be represented by a directed graph $G = (V, E)$, $|V| = N > 1, |E| = M$. The arms of the bandit are in a one-to-one correspondence with the nodes of G, which we use interchangeably where clear. For $(i, j) \in E$, there is a non-negative switching cost $c(i, j) \in \mathbb{R}_{\geq 0}$ in $\mathcal{A}$, equivalent to edge weight $w_{i,j} = c(i, j)$. This cost must be paid, discounted against the reward of the network bandit, in order to play arm j after node i. These costs represent the known overhead cost which the player must pay in order to play the bandit.

Play happens as follows: initially, the bandit player is assigned a starting node i_0 at time $t = 0$. At the beginning of the next discrete time step t, the player moves along some edge $(i_{t-1}, i_t) \in E$ and the plays arm i_t. The reward received for that decision is drawn i.i.d. from a distribution X_{i_t}, subtracted by the cost to move to i_t, which is $c(i_{t-1}, i_t)$. This procedure repeats until $t = T$, allowing T plays and incurring the cost of T edges. This procedure is reminiscent of the finite-horizon multi-armed bandit problem with non-uniform switching costs, with reward-affecting constraints on the order in which the actions are played.

A network bandit is given a possibly non-informative prior distribution of the arm reward distributions $\{X_i\}_{i \in V}$, denoted $\{\pi_0^i\}$ for $i \in V$. This setting

sets up bandit-play which simultaneously: update all posterior distributions π_n^i while sampling the optimal arm increasingly more often.

The required output from a player is to construct the next node on walk W_t, given a past walk history $W_{t-1} = (i_0, ..., i_{t-1})$ and one sample from X_i for each instance of $i \in W_{t-1}$. We use W_T to denote that the walk is constructed over a finite-horizon of length T. Informally, the player's goal is to generate W_t that maximizes the sum of the rewards of the plays subtracted by sum of the switching costs, with knowledge of the reward distributions available from W_{t-1}, for $t \leq T$.

Notation: We denote the expectation of random variable X as $\bar{X}$. $V(G)$ is the vertex set of graph G. Any set of samples drawn from distributions is i.i.d. unless stated otherwise.

2 Overview

We propose the network bandit setting to generalize the bandit setting when the action sequence of a player is restricted and incurs expense according to what order the actions are played. This work hopes to motivate study of bandit-style algorithms to explore online policy-space expoloration. In natural spaces of policies, the order in which the policies are played will affect how much reward is gained, as well as the total number of time each policy is played. In fact, we can view such policy spaces as switching-restricted bandits (i.e. arms in out-neighborhood $\mathcal{N}_i^{out}$ can be played after node i). To include switching costs, we can include the cost into the expected reward *rate* over continuous plays, which becomes non-decreasing as a function of the contiguous play count. Formally, if an arm i is played positive k times in a row after playing arm j, the player receives expected reward rate of $\rho_{j,i}(k) = \frac{(\sum_{1 \leq p \leq k} \bar{X}_i) - w_{j,i} - (k-1)w_{i,i}}{k}$, which is $o(\bar{X}_i - w_{i,i})$.

See how this subsumes the multi-armed bandit: put $G = K_N$, the directed complete graph with self-loops, with $w_{i,j} = 0, \forall i, j \in V$. This setting is highly exploration-friendly, like the MAB, as any sequence of arm plays is valid and does not affect the expected (or actual) total regret since $\rho_{i,j}(k) = \bar{X}_j, \forall i, j \in V, k \geq 1$.

The focus of this work is the single player setting over any strongly-connected G with self-loops. Without strong connectivity, any policy achieves expected linear regret in the worst case since at some point a decision must be made for the player to leave the current strongly connected component and explore another, without having seen any samples from that component and without the ability to return to the original component. Rather the be limited by corner cases, we want to consider network bandits where deciding to play an arm i does not *eliminate* any arm j from being played again, which always occurs for some pair in the absence of strong connectivity. We use the switching costs to dictate how expensive it is to play these two arms in succession; absence of an edge indicates intermediate arms must be played between plays the two arms. To the best of our knowledge, this is the first time that the following ideas have been proposed simultaneously in a bandit-like setting: heterogeneous and asymmetric switching costs between arms; and constraining future arm plays depending on current arm choice.

2.1 Examples

Like the multi-armed bandit, settings where the network bandit is an effective learning model abound, but solutions in this paper will be most appropriate when the problem observes the following: the action/policy space is finite and relatively small compared to the time horizon; a measurable reward function and cost function; a priori knowledge of switching costs; actions and switching can be cast as discrete events. The following three scenarios demonstrate a wide range of uses of the network bandit, which are followed with discussion of how the network bandit problem and its solutions make progress towards creating theoretically-sound solutions for multi-response problems where the response dynamics incur arbitrary and non-trivial costs.

Emergency Robot Rescue In this scenario the network bandit can be used to model emergency robotic response with little information in a chaotic disaster relief area. In one example, several buildings have been destroyed by a fire, flood, etc. and the emergency robot's goals are to search the remains of each building and look for survivors. The stochastic reward is how many lives a robot can save per time period at each building, conditioned on the estimated survival rate of the accident. Switching costs should be the same units as rewards, so we estimate the lives per time period cost to travel between buildings, estimated from the distance between the buildings and the average number of people the robot can save given the population of the disaster area. Modelling this scenario as network bandit problem will allow the robot to maximize its life saving potential by including the area-specific cost to leave one building and look for survivors in a different area, while maximizing the total amount of lives saved.

Commercial Marketing A store owner, Alice, with limited resources wants to determine her shop setup – how goods are displayed and how the appearance of the store is chosen – in order to maximize daily profits. However, in order to change the shop setup Alice must compensate employees to enact the transition between two setups, which occurs overnight. Consequently, the current shop configuration cannot be transformed into any other configuration overnight, either limited by time or available man-hours. We can model this problem as a network bandit, where the arms are the different store setups, the rewards are the daily profits, and the edge constraints encode the labor cost or impossibility of transforming the store between two setups. By solving the bandit problem, we give Alice the opportunity to maximize her business' profits over time and learn which shop configurations are most profitable, while considering the labor costs to transform her store and the realities of restocking shelves, creating signs, etc.

Non-profit Organization A non-profit organization must organize its limited set of employees to coordinate daily fundraising campaigns. Some employees act as directors, guiding the rest of the employees to canvas for donations throughout a city. As with any team, the effect of a chosen hierarchy and task assignment may have significant and highly non-linear relationship with the overall productivity of the team. In order to tease out the effective of these complex relationships, we model each feasible set of task assignments as an arm

– e.g. some employees are volunteers and may not be directors according to organization rules. Switching task assignments will require retraining of any director who wasnt recently a director, which eats away at fundraising dollars. We can include more constraints if the chair of the organization wants to maintain continuity of leadership – e.g. no more than one director can be reassigned at any time. These constraints can be encoded into a network bandit, which then can be solved to provide donors with an impartial and mathematically sound optimization of donation dollars.

Remarks The network bandit problem allows a general framework for online sequential learning problems, where the sequence of samples is not given nor random, but in fact controllable by the learner. Incorporating switching costs induces a sequential learning cost, removing the exchange properties of many multi-armed bandit analysis but opening up new avenues for theory-backed practical solutions to seemingly disparate and difficult organizational and decision-based online learning problems.

3 Solution Space

This section explores nuances of the network bandit problem regarding the solution space, which we show has a richer structure than that of the traditional multi-armed bandit. Interestingly, generalizing the multi-armed bandit to the network bandit illustrates how much or little information a player needs in order to play the optimal solution with high probability. The machinery developed motivates us to call into question the traditonal notion of regret in a bandit setting.

3.1 Oracle Network Regret

To start, we define oracle network regret, which generalizes the notion of total expected regret used in earlier bandit literature to this constrained setting. Let W_T be the sequence of arms of the bandits which are played over a horizon of length T, which must be a walk in the graph G. Like most MAB settings, our goal is to minimize the expected difference between the reward a reward-maximizing oracle would achieve with *a priori* knowledge of $\{X_i\}_{i \in V}$, and an algorithm we provide. Unlike the standard MAB, we need to decide how the initial arm of the player will be decided in order for the network bandit to be well-defined. Keeping in the spirit of the expected regret measure, we assume the oracle must play from the same initial position as the player – the difference in reward stems from prior distribution accuracy only, not initial configuration of the problem. Using a well-defined procedure to select v_0, the player's initial arm, network regret is well defined if we set it to be the expected difference in reward over the distribution of initial positions. We formalize in the following paragraphs.

Define the expected network reward of a walk $W = (v_0, ..., v_k)$ in G as

$$R_W(v_0) := \mathbb{E}\left[\sum_{0<i\leq k} X_{v_i} - w_{v_{i-1},v_i}\right] \tag{1}$$

This discounts the rewards of all arm plays after the initial node by the cost to reach that node over the edge used by the walk. Let $R_{W^*}(v_0) := R^*(v_0)$ be the maximum expected reward over any walk starting at v_0. Then the expected network regret of a policy π is

$$\mathcal{R}^\pi(v_0) = \mathbb{E}[R^*(v_0) - R_W(v_0)]$$

where expectation is taken over randomness in the walk generated by the policy and the reward samples. So by defining $\mathcal{V}$ as a distribution over the nodes of G (and thus the arms), we arrive at the formal statement of the network bandit problem.

Problem 1 *Stochastic Network Bandit Problem Given a network bandit* $(G, \{\mathcal{X}_i\}_{i\in V}, \mathcal{V}, T)$*, find a policy* π *which minimizes*

$$\mathbb{E}[\mathcal{R}^\pi(v); v \leftarrow \mathcal{V}] \tag{2}$$

4 Network Bandit Solution Space

If we know the means of the arm distributions, then we can provide a recursive solution to an arbitrary finite-horizon network bandit $\beta = \{(G, \{\mathcal{X}_i\}_{i\in V}, \mathcal{V}, T) | T \in \mathbb{N}\}$. Let $L(i,t)$ denote the maximum amount of expected additional reward receivable by a bandit player who has previously played arm i and has t more plays left. Define $L(i,0) = 0$. The following equation for $L(i,t)$ gives the maximum possible reward for a player at $L(i,t)$ and can be calculated via dynamic programming in $O(TM)$ time for general network bandits and $O(min(NM, TM))$ time for IPA bandits with non-degenerate solutions since the last $T - N + 1$ plays at any node are stationary walks if $T > N$ and this can be precomputed.

$$L(i,t) = max_{j\in N_i^{in}} \Big\{ L(j, t-1) + \bar{X}_i - w_{j,i} \Big\} \tag{3}$$

If the player starts at node i_0, then the walk W constructed by iteratively appending $i_{k+1} = argmax_{j\in\mathcal{N}_{i_k}^{out}} L(j, T-k)$ to W has maximum reward over all walks which started at i_0. By definition, this solution is optimal.

4.1 Information Available to the Oracle

The heart of any bandit problem is the difference in information about the arm distributions between an oracle and the player with inaccurate priors. The optimal solution above can be calculate if the means of the arm distributions are known. However, in the MAB setting, the oracle only requires the index of an arm with maximum mean in order to play the optimal solution. Thus, forcing any player to play over a network requires more information in order to play optimally.

From from the discussion above follows the natural question: How much information is required to play optimally when plays are constrained and switching costs exist?

In the network bandit setting, the optimal arm index is not sufficient for optimal play. In fact, even an ordering of the expectations of $X_i, i \in V$ for unknown distributions is not sufficient. Informally, this holds because the relative distance between expectations can be arbitrarily large or small.

We formalize how close a player can get with information available to it in lemma 1 and lemma 2. We say that a policy π is α-optimal with respect to a set of network bandit instances $\mathcal{B}$ with prior samples for each instace B with arms V given by $S_B = \{X_i^1, X_i^2, ..., X_i^k | i \in V\}$ if

$$\alpha \geq inf_{B \in \mathcal{B}} \frac{\mathbb{E}_B[R^\pi(v); v \leftarrow \mathcal{V}]}{\mathbb{E}_B[R^*(v); v \leftarrow \mathcal{V}]} \tag{4}$$

Under this definition, the oracle is 1-optimal with respect to the set of all bandit instances Ω with prior samples $S_B = \{X_i^1, X_i^2, ...| i \in V\}, \forall B \in \Omega$. Also, a player policy may be α-optimal

The following lemmas provides an argument that oracle regret borrowed from the MAB literature may make finding solutions with similar order expected regret difficult in the worst case.

Lemma 1 *For any nonempty set of bandit instances $\mathcal{B}$ with non-degenerate initial arm distribution $\mathcal{V}$ and finite samples $S_B, \forall B \in \mathcal{B}$, there exists an $\alpha < 1$ for which any player is at most α-optimal.*

Proof For the following lemma, we allow the player to have ϵ-close measurements of true means. That is the player is given a set of i.i.d. samples for each random variable X_i such that $(1-\epsilon)\hat{\bar{X}}_i \leq \bar{X}_i \leq (1+\epsilon)\hat{\bar{X}}_i$.

Lemma 2 *For any set of arm distributions $X_i, i \in V$ with unique optimal arm, any $\alpha \in [0, 1)$ there exists a non-degenerate bandit instance B with arm distributions $\{X_i\}_{i \in V}$ with strongly connected G for which any player with ϵ-close estimates of the true means is at most α-optimal when*

Proof Given $X_i, i \in V$ with two nodes with distinct means, we must have $|V| \geq 2$. Assume the player has finite samples S_X.Select arbitrary $\alpha \in [0, 1)$. We want to show that a non-degenerate bandit example exists for which two candidate solutions appear similar enough that any random process used to decide between them will cause the resulting solution to be no more than α-percent of optimal.

W.l.o.g relabel $i \in V$ such that that $\bar{X}_i' \leq \bar{X}_{i'+1}$ for $1 \leq i' \leq N-1$. Let $E(G) = \{(1,2),(2,3),...,(N-3,N-2)\} \cup \{(N-2,N-1),(N-1,1),(N-2,N),(N,1)\} \cup (N-1,N-1),(N,N)$. Clearly, G is strongly connected and $\Delta = \bar{X}_N - \bar{X}_{N-1} > 0$. Let $\mathcal{V}$ have all mass at node $N-1$.

In this setting, if the the player starts at arm N, any adaptive allocation rule (see Lai and Robbins 85) can be split into k phases, each of which the player pulls arms $1, 2, ..., N-2$ and then $N-1$ or N at least once. To prove our result, we reduce this problem to the two-armed bandit and utilize lower bounds on minimax regret from Theorem 1 from Kulkarni and Lugosi (97) to construct edge weights and a time T which guarantees at most α-optimal performance.

Notice that in our bandit, for time horizon T, the set of all adaptive allocation rules Ψ can be partitioned based on the number of times node 1 is played, for k plays on node 1 the subset is $\Psi_k = \{S = \psi_1, ..., \psi_n | S \text{ contains } X_1 \ k \text{ times}\}$. Since arm 1 can only be played after arm $N-1$ or N, $P_1(T) \leq P_{N-1}(T) + P_N(T)$,where $P_i(T)$ is the number of times arm i is played up to and including

time T. Also, arms $N-1$ and N can only be played after arm $N-2$, and arm $N-2$ can be played on the terminal play at most once, so $P_{N-2}(T)+1 \leq P_{N-1}(T)+P_N(T)$. note that arm $1 < i \leq N-2$ can be played only after arm $i-1$ and only one terminal play is allowed, so $\exists s \in \{1,...,N-2\} s.t. P_r(T) = P_s(T), \forall 1 \leq r \leq s$ and $P_s(T) = 1 + P_t(T), \forall s < t \leq N-2$.

If we set $w_{i,j} = \mu_j, 2 \leq j \leq N-2$, then the player receives no expected reward for each play that does not end in node $N-1$ or $N-2$.

4.2 MAB vs Network Bandit Limiting Solutions

Without restrictions on switching costs, we can create solutions which do not resemble oracle solutions to existing MAB problems. As a primary example, if we allow $c(i,j) > c(i,i), i \neq j$, then the oracle in some network bandit instances will choose to not find and exploit the best arm in the network, but repeatedly traverse a *non-stationary* walk. We call the walks with maximal expected per-play reward rate the *limiting solutions*, e.g. if $W_{lim} = (v_0,...,v_k = v_0)$ then if $\mathcal{W}$ is the set of all walks in G then $W_{lim} = \{W | W = argmax_{W \in \mathcal{W}, v \in W}(R_W(v)/|W - 1|)\}$. By convention, we repeat the initial node of the walk at the end to keep reward notation consistent. We call this limiting solution stationary if $v_i = v_j$ for $1 \leq i < j \leq k$, and non-stationary otherwise. Since walks must repeat one node at least twice, then we put the lowest index arm in the walk first so each unique walk with maximum per-step reward has a unique solution.

The full solution of the oracle, given an infinite horizon, will be to reach v_0 after an initialization period and play the limiting solution repeatedly. We will show conditions for which an optimal stationary limiting solution exists; we will use this to prove that the oracle will move along a simple path to the unique node of the limiting solution and play it until time runs out. To guarantee stationary limiting solutions, we invoke three simple assumptions – existence of self-loops, inertial play and a sufficiently large horizon. We show a closed form solution which gives a sufficient value of T for this to hold in the next section. Define the following as inertial play assumptions (IPA): i) $c(j,i) \geq c(i,i), \forall (j,i) \in E$; ii) $\exists (i,i) \in E, \forall i \in V$. These ensure that playing an arm repetitively gives no worse reward than playing an arm immediately after a different arm.

Claim 1 *Any network bandit problem which satisfies IPA has a limiting solution which is stationary. Condition (i) is also necessary for the statement to hold. Also, if the inequality in (i) holds strictly, then any limiting solution is also stationary.*

To show that condition (i) of IPA is necessary, we construct a small example. Let $G = K_2$ and add self-loops. Set $\bar{X}_1, \bar{X}_2 = 1, w_{i,i} = 1, w_{i,j} = 0$ if $i \neq j$ and $i,j \in \{1,2\}$. The walk $W_T = (v_1, v_2, v_1)$ has expected reward of 1 per step, while $W_T^1 = (v_1, v_1)$ and $W_T^2 = (v_2, v_2)$ both have expected reward of 0 per step. Thus, all stationary limiting solutions are strictly suboptimal.

To show that a stationary limiting solution exists and is optimal under the assumptions, assume $W_k = (v_0,...,v_k = v_0)$ is a non-stationary, optimal limiting solution. For some i, $r(v_i) - c(v_{(i-1) mod (k+1)}, v_i) \geq r(v_j) - c(v_{(j-1) mod (k+1)}, v_j) \forall j \in [k], j \neq i$. By using the walk $W = (v_i, v_i)$, each step of the walk gives reward $r(v_i) - c(v_i, v_i) \geq r(v_i) - c(v_{(i-1) mod (k+1)}, v_i)$, so the expected reward rate of

the stationary solution (v_i, v_i) is at least as must as that of the optimal non-stationary solution. Further, if the inequality holds strictly, we have a contradiction that W_k was optimal, which proves the second part of the claim. ■

4.3 Degenerate Oracle Solutions

Although we have a stationary limiting solution under IPA, the oracle's finite horizon walk W_T^* may not include any subset of the limiting solution. We argue these instances represent cost-dominated scenarios, where the best action available is too expensive to reach in the time available to play – the bandit degenerates. For IPA network bandits, this will occur if the edge weight over any paths from the starting node to i^* overwhelms the amount of reward which can be drawn over the remaining plays at i^* and the plays over the corresponding path. For arbitrary bandits, if the oracle never plays the limiting solution as a part of its solution (at least once, possibly with shifted starting point), then we say the instance is degenerate. We state this formally below.

Definition 1 *Let $\mathcal{W}_{lim}$ be the set of limiting solutions for network bandit and $\mathcal{W}_T$ be the set of oracle solutions to the network bandit $\beta = (G, c, \{\mathcal{X}_i\}_{i\in V}, \mathcal{V}, T)$. The instance β is degenerate if $\forall W_T \in \mathcal{W}_T, \forall W_{lim} \in \mathcal{W}_{lim}$, W_T does not contain any sequence $S_{lim} \in W_{lim}$ as a subsequence, and is non-degenerate otherwise.*

As a player, a non-degenerate solution is unfriendly to MAB-style learning, as we are incurring significant regret attempting to look for the best arm. To see such an instance for any time T, construct $G = K_2$,$V = \{1, 2\}$, $\mathcal{V} = 1$ with probability 1, $c(1, 1) = 0, c(1, 2) = 1, c(2, 2) = 0, c(2, 1) = 0$, $\mu_1 = 0, \mu_2 = (1/(T + 1))$. Any policy π^* starting at 1 as given will incur a negative expected reward if it moves to arm 2, but non-negative reward if it remains at arm 1. Since no optimal solution will include the walk $(2, 2)$ as a subsequence, then we have a degenerate instance. Note that this network bandit satisfies IPA, but the time horizon is simply too short to overcome the cost to move and play the limiting solution.

We can partition the set of instances $\mathcal{B} = \{(G, c, \{\mathcal{X}_i\}_{i\in V}, \mathcal{V}, T)|T \in \mathbb{N}\}$ into non-degenerate and degenerate class. Let the non-degenerate subset be $\mathcal{B}_N$. The following claim gives a lower bound for T such that $\mathcal{B}_N \supset \mathcal{B}_{\geq T_0} = \{(G, c, \{\mathcal{X}_i\}_{i\in V}, \mathcal{V}, T)|T \geq T_0\}$. Note, we say a walk or path is empty if it is a single node.

Claim 2 *Let $\mathcal{B} = \{(G, c, \{\mathcal{X}_i\}_{i\in V}, \mathcal{V}, T)|T \in \mathbb{N}\}$ satisfy IPA. Also, assume that there exists a j for which $\mu_j = \bar{X}_j - w_{j,j} > \bar{X}_i - w_{i,i}, \forall i \in V, i \neq j$, and let $\mu_j = \bar{X}_j - w_{j,j}, \mu_{2nd} = max_{s\in W_T\setminus j}\bar{X}_s - w_{s,s}$ and $c_{max} = max_{e\in E}c(e), \bar{X}_{min} = min_{i\in V}\bar{X}_i$. The set of non-degenerate instances $\mathbb{B}_N \supset \mathcal{B}_{\geq T_0} = \{(G, c, \{\mathcal{X}_i\}_{i\in V}, \mathcal{V}, T)|T \geq T_0\}$, if*

$$T_0 = \lceil \frac{(n-1)(2c_{max} + \mu_j - \bar{X}_{min})}{\mu_j - \mu_{2nd}} \rceil \tag{5}$$

Moreover, if $T \geq T_0$ the oracle will never play any arm $i \neq j$ more than once.

Proof Let $W_T = (v_0, ..., v_T)$ be a solution to a network bandit $\mathcal{B}$. Put v^* as the first arm in W_T which maximizes $X_{v_f} - c(v_{f-1}, v_f), 1 \leq f \leq T$. The sub-walk (possibly empty) $W_{f,T} = (v^* = v_f, ..., v_T)$ is equal to $(v^*, ..., v^*) = V$, or else we can replace $W_{f,T}$ with V with at least as high expected reward since $\bar{X}_s - w_{p,s} \leq \bar{X}_s - w_{s,s} \leq \bar{X}_{v^*} - w_{v^*,v^*}$ for $s \in W_T \setminus v^*, p \in \mathcal{N}_s^{in}$ by assumption. Thus, an optimal solution exists for any network bandit which end with and repeats the node with highest stationary mean along the walk. If $v_T = j$, then $v^* = j$ and this solution is strictly optimal since $\bar{X}_s - w_{p,s} \leq \bar{X}_s - w_{s,s} < \bar{X}_{v^*} - w_{v^*,v^*}$

The above argument shows that if $k < T$, then if an optimal solution exists where $v_k \neq v_T$ is played $h > 1$ times, then we can play v_k once and play v_T $h-1$ more times and receive at least as much reward. Additionally, if the solution is non-degenerate, then no optimal solutions may play nodes in $W_T \setminus j$ more than once, which proves the last line of the claim if we show that $T \geq T_0$ ensures non-degeneracy.

Let $\bar{X}_{2nd}$ be the 2nd highest mean of any arm distribution in the bandit, c_{max}, c_{min} be the highest and lowest cost non-self loop edge, resp. and μ_{2nd} be the second highest stationary one-step reward of the bandit. Also, we know a walk of maximum expected reward for any bandit exists which is a (possibly empty) simple path followed by repeatedly playing the final node. If we let the following be true for $1 \leq k, l \leq n$, then the lowest expected reward from a non-degenerate solution of $\mathcal{B}$ is greater than or equal to the highest expected reward from a degenerate solution, also from $\mathcal{B}$.

$$(k-1)(\bar{X}_{min} - c_{max}) + (T-k+1)(\mu_j) \geq (l-1)(\bar{X}_{2nd} - c_{min}) + (T-l+1)(\mu_{2nd}) \quad (6)$$

Solve for T and take the maximum over all k, l to lower bound T_0, a sufficiently large time to ensure the network bandit with $T \geq T_0$ is non-degenerate.

$$\begin{aligned}
T \leq & max_{1 \leq k,l \leq n} \frac{(l-1)(\bar{X}_{2nd} - c_{min} - \mu_{2nd}) - (k-1)(\bar{X}_{min} - c_{max} - \mu_j)}{\mu_j - \mu_{2nd}} \\
\leq & max_{1 \leq k,l \leq n} \frac{(l-1)(-c_{min} + c_{max}) - (k-1)(\bar{X}_{min} - c_{max} - \mu_j)}{\mu_j - \mu_{2nd}} \\
\leq & max_{1 \leq k,l \leq n} \frac{(k-1)(c_{max} + \mu_j - \bar{X}_{min}) - (l-1)(c_{min} - c_{max})}{\mu_j - \mu_{2nd}} \\
\leq & \lceil \frac{(n-1)(2c_{max} + \mu_j - \bar{X}_{min})}{\mu_j - \mu_{2nd}} \rceil
\end{aligned}$$

The last bound is no less than $n-1$, so a walk from i_0 can reach j from any other node. Finally, by setting T_0 as the last upper bound, then all instances with $T \geq T_0$ will have non-degenerate solutions. ■.

5 Policy Design

Generating general policies essentially decides on one of the following at each time step: generate a new walk based on new information gather from playing the bandit; or continue the walk previously constructed. Altering the previous

walk may involve: playing a node repeatedly more times, i.e. extend the walk by repeatedly moving on the self loop, reduce the number of times a walk repeatedly visits a node, or add/eliminate a section of the walk which visits multiple nodes. An inherent difficulty in solving this problem is that the average reward of a walk may vary wildly across the edges of the walk, so each edge-end in the walk

Non-adaptive Batch Network UCB Algorithm

Our first proposed policy is simple, and requires little computational power. We find the proof follows naturally from the UCB1 proof in [1] , and the regret includes an additive logarithmic factor when compared to standard UCB1 algorithm, which depends on the structure of the network bandit. This policy is non-adaptive, as it does not adjust the length of the walks depending on the sampled means of the arms, nor does it adjust the number of samples of arms during a walk. This is a warm-up algorithm with clean analysis of regret. We assume that the reward distributions and costs are normalized by the same constant so that all variables drawn from either set fall in $[0, 1]$,

Notation The following are notation used in the algorithm and analysis of regret. For a walk $W = (w_0, ..., w_k)$, let $\delta_i(W)$ be the number of times i appears at the end of an edge $(w_{t-1}, w_t = i), 1 \leq t \leq k$, which we also call the number of times W visits i. The stationary mean gap $\Delta_i = (\bar{X}_{v^*} - w_{v^*,v^*}) - (\bar{X}_i - w_{i,i}) = \mu_{v^*} - \mu_i$ is the expected difference in reward of playing arm i repeatedly vs the optimal stationary arm repeatedly. A *node-covering walk* from i_0 is a walk which starts at i_0, visits all other nodes at least once and then returns to i_0. We use $\hat{y}_n$ to represent the sample average of a variable y over n samples. When we update this sample mean with a new sample s we set $\hat{y} \leftarrow (n\hat{y} + s)/(n + 1)$. We say a walk finishes when a walker has no more nodes left to traverse, its cost is the sum of the edge costs over the walk, and its length is the number of edges on the walk.

Algorithm 1 BatchUCB1

//INITIALIZATION
Initial node $i_0 \leftarrow \mathcal{V}$
Calculate node-covering walk $\mathcal{W}$ of G with cost C_{apx} and length L
Move to the next arm on the walk, then play that node; repeat until walk reaches and plays i_0 for the final time.
Update $m_i \leftarrow \delta_i$ and $\hat{\mu}_i, \forall i \in V$
$t = L; k = 0$
while $t < T$ **do**
  $k \leftarrow k + 1$
  $c_i \leftarrow \delta_i(\mathcal{W}), \forall i \in V$
  $m_i \leftarrow m_i + \delta_i(\mathcal{W}), \forall i \in V$
  $t \leftarrow t + L$
  while $t < L \cdot (2^k + 1)$ **do**
    Put $i = argmax_{i \in V} \left\{ \hat{\mu}_i + \sqrt{2ln(t)/m_i} \right\}$
    $c_i \leftarrow c_i + 1.$
    $m_i \leftarrow m_i + 1$
    $t \leftarrow t + 1$
  end while
  Put $(p_1, ..., p_{\delta_i}) = (|e_1|, ..., |e_{\delta_i}|)$, where $\{e_i\}_{i \in [\delta_i]}$ is a partition of $\{1, ..., c_i\}$.
  while $\mathcal{W}$ is not finished **do**
    Play ith node on $\mathcal{W}$, p_k time(s) if this is the kth occurrence of i on $\mathcal{W}$
    Update $\hat{\mu}_i$
    Move to the next node in $\mathcal{W}$
  end while
end while

The main idea behind BatchUCB1 policy is to schedule arm pulls in batches of doubling size. Using the arm reward estimates from previous plays, we schedule arms according to UCB *before* playing the arms. To execute a batch of samples, we traverse G according to a low cost walk, stopping to play i p_k times if this is the kth time i occurs on the walk. We can view this policy as a batched scheduling algorithm to be executed over walks of increasing total length. Compared with a non-batch UCB, there is a small decrease in overall reward, since the confidence bound on each sample mean is increasingly "outdated" as the player gets further along in her walk. This policy requires a computation of a minimum node-covering walk, which has a simple 2-approximation for undirected G with self loops: find the MST of G and perform walk according to DFS from root i_0. We do not focus on this sub-problem at the current time, and assume an approximation can be efficiently calculated for arbitrary directed G.

5.1 Regret of BatchUCB1

To bound the network regret over arbitrary $\mathcal{V}$, we first assume the maximum expected network reward of the oracle. This occurs when $\mathcal{V}$ has all mass at the initial node of the highest reward length T walk, $W^* = (v^*, ..., v^*)$ where $\mu^* := \mu_{v^*} = X_{v^*} - w_{v^*,v^*}$ is maximum in the network. We bound the regret by by counting the expected number of plays at all other nodes, scaling each count by the stationary reward mean gap $\Delta_i \equiv \mu^* - \mu_i$, and then add the total edge cost of the walk.

Define C_{apx} as the cost of the minimum cost node-covering walk $\mathcal{W}$ which can be calculated in polynomial time and L the number of edges on the walk. We now present our first theorem.

Theorem 1 *The expected network regret of BatchUCB1 on IPA network bandit with strongly-connected G, unknown but finitely supported $\{X_i\}_{i\in V}$ and arbitrary $\mathcal{V}$ is at most*

$$\Big(\sum_{i\in V:\mu_i<\mu^*} \Delta_i\Big(\frac{8lnT}{\Delta_i^2} + 1 + \pi^2/3\Big)\Big) + (\lceil log(T/L)\rceil + 1)\Big(C_{apx} + \sum_{i\in V}(\Delta_i\delta_i - w_{i,i})\Big) \tag{7}$$

when δ_i is the number of times the node-covering walk $\mathcal{W}$ of cost C_{apx} visits i, and L is the number of edges in $\mathcal{W}$.

Corollary 1 *The maximum increase in regret bound for BatchUCB1 compared to UCB1 for network bandit $\beta = (G, c, \{X_i\}_{i\in V}, \mathcal{V})$ compared to $\{X_i\}_{i\in V}$ is*

$$(\lceil log(T/L)\rceil + 1)\Big(C_{apx} + \sum_{i\in V}(\Delta_i\delta_i - w_{i,i})\Big) \tag{8}$$

To prove this, we state a version of the Chernoff-Hoeffding Bound from [2]

Lemma 3 *$X_1, ..., X_n$ are random variables with range $[0, 1]$ where $\mathbb{E}(X_t|X_{t-1}, ..., X_1) = \mu$ for all $1 \le t \le n$. Denote $S_n = \sum_{1\le t\le n} X_t$. Then for all $a \ge 0$*

$$\mathbb{P}(S_n \ge n\mu + a) \le e^{-2a^2/n}$$
$$\mathbb{P}(S_n \le n\mu - a) \le e^{-2a^2/n}$$

Proof of Theorem 1 Since node-covering walk $\mathcal{W}$ is fixed in the algorithm, put $\delta_i = \delta_i(\mathcal{W}$. Let the number of times an arm i is played after τ total arm plays to be be $P_i(\tau)$. The total number of times that i is scheduled to be played is tracked by m_i, while the number of times that i is scheduled to be played on the kth walk is c_i. By scheduling multiple plays of each arm before playing any arms, we create a "batch" of plays, where each batch is associated with a single node-covering walk.

We always charge δ_i plays to arm i for each batch by setting $c_i = \delta_i$ at the beginning of every batching, and never decrease it. So $c_i \geq \delta_i$, and $\sum_{k\in 1,\ldots,\delta_i} p_k = c_i$ by definition so the number of batched samples attributed to i will be no less than the number of times i will be visited on $\mathcal{W}$. Since we want to play i c_i times on a node-covering walk, we simply play i a total of $c_i - \delta_i$ more times than is given by the node-covering walk. To maintain the node-covering property of the kth walk these additional scheduled plays will be played as subsequent plays when the bandit is already at i.

Let K be the smallest integer for which $T \leq L(2^{K+1}-1)$, so $K \leq \lceil log(T/L) \rceil$ and let $Q[\Gamma]$ be the binary function equal to 1 iff the event Γ is true. Observe the first play of the kth node-covering walk occurs at time $t_k^s = L(2^k - 1) + 1$ and the last play is at time $t_k^e = L(2^{k+1} - 1)$ if $0 \leq k \leq K$, excepting $t_K^e = T$. These intervals encode the time to perform an initialization walk plus walks of doubling length thereafter. When arm plays during the kth node-covering walk for $k \geq 0$ are being batched, δ_i samples are always charged to be played. To schedule remaining arm plays, we iterate t over $t \in [t_k^s + L, t_k^e]$ and select the arm with maximum UCB, using the sample time and scheduled sample count rather than real time and real arm play count. This scheme ensures that the length of the kth node-covering walk is always $2^k L$ for $k \geq 0$.

During arm play scheduling, one of two things happens: an optimal arm is scheduled, or a non-optimal arm is scheduled. Because the kth walk is a valid sequence of bandit plays and the player will complete at most $K + 1$ node-covering walks, then the total number of plays of all arms is no more than the total number of scheduled plays of all arms.

Linearity of expectation gives us

$$\sum_{i\in V} \mathbb{E}[P_i(T)] \leq \sum_{i\in V} \mathbb{E}[m_i(T)] \tag{9}$$

Define by $I_i(t)$ the indicator function that is equal to i if arm i is scheduled to be played when sample time is t. Then

$$m_i(T) \leq \sum_{k=0}^{K} \Big\{ \delta_i + \sum_{t=t_k^s+L}^{t_k^e} Q[I_i(t) = 1]) \Big\} \tag{10}$$

Let l_i be an integer no less than δ_i. So

$$m_i(T) \leq l_i + \sum_{k=1}^{K} \Big\{ \delta_i + \sum_{t=t_k^s+L}^{t_k^e} Q[I_i(t) = 1, m_i(t-1) \geq l_i]) \Big\}$$

Denote C_{t,m_i} as $\sqrt{\alpha ln(t)/m_i(t)}$, where $m_i(s)$ is the largest value that the variable m_i takes when $t \leq s$. At each sample time $t \in [t_k^s + L, t_k^e]$, the algorithm

uses the sample means of stationary reward from all previously played arms, namely $\hat{\mu}_{i,t^e_{k-1}} = \hat{\bar{X}}_{i,t^e_{k-1}} - w_{i,i}$. So the indicator $I_i(t) = 1$ over these times only if arm i has higher UCB value than the optimal arm at time t. Hence,

$$m_i(T) \leq l_i + \sum_{k=1}^{K} \Big\{ \delta_i + \sum_{t=t^s_k+L}^{t^e_k} Q[\hat{\mu^*}_{m^*(t^e_{k-1})} + C_{t-1,m^*(t-1)} \leq \hat{\mu}_{i,m_i(t^e_{k-1})} + C_{t-1,m_i(t-1)}, \\ m_i(t-1) \geq l_i]) \Big\}$$

Rather than bounding the event inside $Q[\cdot]$, we bound using a higher probability event: the inequality is true while assuming the following is true inside the double sum: $m_i(t-1) \geq l_i$. This implies that $m_i(t-1) \geq l_i \geq \delta_i$ inside the double sum and the following is true.

$$m_i(T) \leq l_i + \sum_{k=1}^{K} \Big\{ \delta_i + \sum_{t=t^s_k+L}^{t^e_k} Q[\hat{\mu^*}_{m^*(t^e_{k-1})} + C_{t-1,m^*(t-1)} \leq \hat{\mu}_{i,m_i(t^e_{k-1})} + C_{t-1,m_i(t-1)}) \Big\}$$

From standard UCB arguments, we can see that one of the following must be true for $Q[\cdot] = 1$ for a fixed k

$$\hat{\mu^*}_{m^*(t^e_{k-1})} \leq \mu^* - C_{t-1,m^*(t-1)} \tag{11}$$

$$\hat{\mu}_{i,m_i(t^e_{k-1})} \geq \mu_i + C_{t-1,m_i(t-1)} \tag{12}$$

$$\mu^* < \mu_i + 2C_{t-1,m_i(t-1)} \tag{13}$$

To bound the probability of the first inequality, we use

$$\begin{aligned}
&\mathbb{P}[\hat{\mu}^*_{m^*(t^e_{k-1})} \leq \mu^* - C_{t-1,m^*(t-1)}] \\
&= \mathbb{P}[\hat{\bar{X}}^*_{m^*(t^e_{k-1})} \leq \bar{X}^* - C_{t-1,m^*(t-1)}] \\
&= \mathbb{P}[m^*(t^e_{k-1})\hat{\bar{X}}^*_{m^*(t^e_{k-1})} \leq m^*(t^e_{k-1})\bar{X}^* - m^*(t^e_{k-1})C_{t-1,m^*(t-1)}] \\
&\leq exp\Big(-2 \cdot (m^*(t^e_{k-1}))^2 \frac{\alpha \cdot ln(t-1)}{m^*(t-1)} \cdot \frac{1}{m^*(t^e_{k-1})} \Big) \\
&= exp\Big(-2 \cdot m^*(t^e_{k-1}) \frac{\alpha \cdot ln(t-1)}{m^*(t-1)} \Big) \\
&\leq exp(-\alpha ln(t-1)) \\
&= (t-1)^{-\alpha}
\end{aligned}$$

The second line follows since the sample mean of μ_i and true mean of μ_i both include the additive term $w_{i,i}$. The third line follows from lemma 3, which can be applied since X_i takes values in $[0, 1]$. Since for any event $Q[\cdot]$ and fixed k we have $2t^e_{k-1} = 2L(2^k - 1) \geq L(2^{k+1} - 1) = t^e_k$ and $t^e_{k-1} + L + 1 \leq t \leq t^e_k$, implying

$m^*(t-1) \leq 2m^*(t^e_{k-1})$ since we can schedule the optimal arm at most once for each sample time, which gives us the fifth line. We can bound the second UCB inequality in the same manner, with same probability bound, since all the above arguments are independent of arm choice, and the Chernoff-Hoeffding bound probability still holds.

Recall that we know inside the double sum that $m_i(t-1) \geq l_i$. Now, we force the third UCB inequality to be false for a fixed k by setting $l_i \geq max(\delta_i, \lceil 4\alpha ln(T)/\Delta_i^2 \rceil)$ which satisfies $l_i \geq \delta_i$ as required, giving

$$\begin{aligned}\mu^* - \mu_i - 2\sqrt{\frac{\alpha \cdot ln(t-1)}{m_i(t-1)}} &\geq \mu^* - \mu_i - 2\sqrt{\frac{\alpha \cdot ln(T)}{l_i}} \\ &\geq \mu^* - \mu_i - \Delta_i = 0\end{aligned}$$

Thus we get for $\alpha = max(2, \frac{\delta_i \cdot \Delta_i^2}{4lnT})$ that $l_i = \lceil 4\alpha ln(T)/\Delta_i^2 \rceil \geq max(\delta_i, \lceil 4\alpha ln(T)/\Delta_i^2 \rceil)$ makes the third UCB bound false. We finally bound $m_i(T)$ as follows

$$\begin{aligned}\mathbb{E}[m_i(T)] &\leq \lceil 4(2 + \frac{\delta_i \cdot \Delta_i^2}{4lnT}) ln(T)/\Delta_i^2 \rceil \\ &\quad + \sum_{k=1}^{K} \Big\{ \delta_i + \sum_{t=t_k^s+L}^{t_k^e} (\mathbb{P}[\hat{\mu^*}_{m^*(t^e_{k-1})} \leq \mu^* - C_{t-1,m^*(t-1)}] + \mathbb{P}[\hat{\mu}_{i,m_i(t^e_{k-1})} \leq \mu_i - C_{t-1,m_i(t-1)}]) \Big\} \\ &\leq \lceil 8ln(T)/\Delta_i^2 + \delta_i \rceil + K\delta_i + \sum_{t=1}^{\infty} 2t^{-2} \\ &\leq 8ln(T)/\Delta_i^2 + 1 + (K+1)\delta_i + \pi^2/3\end{aligned}$$

For arbitrary initial arm distribution $\mathcal{V}$, the expected regret can be upper bounded by assuming the oracle does not play any non-optimal arms or pay any edge costs (besides the cost of the self loop at v^*). The expected network regret then is no more than a sum of expectations of the following: the edge cost incurred over all $K+1$ node-covering walks; the arm play counts $P_i(T)$ scaled by the mean gap Δ_i; the negative of the self-loop cost of arm i multiplied by the number of times we double counted it. This last term comes from the fact that we are counting the number of times i is sampled and scaling by the stationary mean gap, but δ_i of these sample rewards are not discounted by the cost of the self loop at i since we played i after playing another arm; this edge cost is included in the node-covering walk cost. We also use the fact $K \leq \lceil log_2(T/L) \rceil$ and IPA ensures $C_{apx} \geq \sum_{i \in V} w_{i,i}$. Thus the regret for our policy π described by BatchUCB1 is

$$\mathbb{E}_{v\leftarrow\mathcal{V}}[\mathcal{R}^{\pi}(v)] \leq \sum_{i:\mu_i<\mu^*} \mathbb{E}[P_i(T)] + (K+1)(C_{apx} - \sum_{i\in V} w_{i,i})$$
$$\leq \sum_{i:\mu_i<\mu^*} \mathbb{E}[m_i(T)] + (K+1)(C_{apx} - \sum_{i\in V} w_{i,i})$$
$$\leq \sum_{i\in V:\mu_i<\mu^*} \Delta_i(\frac{8lnT}{\Delta_i^2}) + (\lceil log(T/L)\rceil + 1)\delta_i + 1 + \pi^2/3) + (\lceil log(T/L)\rceil + 1)(C_{apx} - \sum_{i\in V} w_{i,i})$$
$$\leq \Big(\sum_{i\in V:\mu_i<\mu^*} \Delta_i\Big(\frac{8lnT}{\Delta_i^2} + 1 + \pi^2/3\Big)\Big) + (\lceil log(T/L)\rceil + 1)\Big(C_{apx} + \sum_{i\in V}(\Delta_i\delta_i - w_{i,i})\Big)$$

which proves Theorem 1. The corollary follows by subtracting the UCB1 bound from this bound. ■

Comments In this bound we can see the original UCB bound, with an additional cost of $(\lceil log(T/L)\rceil + 1)\Big(C_{apx} + \sum_{i\in V}(\Delta_i\delta_i - w_{i,i})\Big)$. Intuitively, this accounts for all network-constraints which are observed during BatchUCB1. In practice, we can use the best known estimates from the values of δ_i to recalculate a node-covering walk which minimizes this value. Additionally, with some small modifications, we can calculate the kth batch of samples and then calculate the minimum cost walk which visits all nodes included in the batch at least once. In instances when low reward arms are found on the periphery of the network bandit, this modification will concentrate the sampling time of our tours on the central, more profitable nodes and reduce the length and cost of switching overall.

5.2 UCB1 Equivalence

We can employ a trick in BatchUCB1 which by inspection gives a matching regret bound for UCB1. The discussion is informal, as the details are tedious but uncomplicated.

To perform, for each maximal clique S' in G with total edge cost of 0, add a super-node S to multi-graph G^+. Now, for all super-nodes in G^+ add the lowest cost edge which existed between a node in each clique. This may in fact be a self-edge. Then BatchUCB1 changes as follows: follow the Eulerian tour of the MST of G^+ instead of G; when first moving to a super-node S, we play the arm which is incident to the edge between super-nodes. To find the number of samples to be played on the walking tour, use the same method as BatchUCB1 but over the value $\max(2U_S, \sum_{i\in S} c_i)$ where U_S is the number of unique nodes in S with an incident edge which was chosen in in the MST of G^+, always less than the degree of super-node in the MST, δ_S. If the partition indicated that p_k samples of the super node S during the kth visit, then perform UCB1 sampling over $\{s \in S\}$, $max(0, p_k - 2)$ times, updating $m_i, \hat{\bar{\theta}}_i$ whenever $i \in S$ is played. To advance the tour, we play the node adjacent to the next super-node in the tour and then move to the next super-node.

We can see that if our network bandit is equivalent to the standard MAB, then $G^+ = (\{S\}, \emptyset)$, where S is a super-node containing all arms. The regret analysis of this algorithm changes very little from the previous proof. We use

δ_S instead of δ_i and C^+_{apx} is the min cost tour on G^+ and our expected regret bound resembles [prove this formally in an appendix]

6 BatchUCB1 With Pruning Commitments

This section covers a more adaptive strategy for scheduling our samples in increasingly sized walk. We can eliminate increasingly larger sections of our walk in order to balance the cost of switching with the expected sampling regret. If the previous samples indicate that the stationary mean gap for a node is larger than the extra switchings needed to explore that node for the remaining time with a high probability, then we could eliminate that node from further walks and increase our total reward with high probability. This we call a pruning commitment, since we will not explore this node at all in the future.

6.1 Lower Bounds on Mean Gaps

For our algorithm, we want to develop intuition on how good our estimates can get for $\Delta_i = \mu^* - \mu_i$, the true mean gap for arm i. Define the sample mean gap $\Delta_i(n_1, n_2) = (1/n_1)\sum_{i\in[n_1]} X^* - (1/n_2)\sum_{i\in[n_2]} X_i$. We want to bound the probability that the sample mean gap is at least γ. We will use this bound to consider when to ignore certain arms in our graph. Define $S^* = (1/n^*)\sum_{i\in[n_1]} X^*$ and $S_i = (1/n_2)\sum_{i\in[n_2]} X_i$. We will also assume for this section that for each arm distribution X_i the sample avergage $(1/n)\sum_{i\in[n]} X_i$ is σ_i/n-subgaussian. The following lemma from [**?**, ?]llows us to bound tail probabilities for subgaussian random variables.

Lemma 4 *Concentration Inequality for Subgaussian Random Variables*
Let X be a σ-subgaussian random variable and let $t > 0$, then the following holds

$$\mathbb{P}[X > t] \leq exp(\frac{-t^2}{2\sigma^2})$$
$$\mathbb{P}[X < -t] \leq exp(\frac{-t^2}{2\sigma^2})$$

We will use this to prove our own lemma regarding the sample mean gap.

Lemma 5 *Sample Mean Gap Lemma*
Let X_i, X_j be random variables, and $X_k - \mathbb{E}[X_k]$ a σ_k^2-subgaussian random variable $k \in \{i, j\}$. Let $\Delta_{i,j} = \mathbb{E}[X_i] - \mathbb{E}[X_j]$ and Then for any $\gamma > 0$ we have

$$\mathbb{P}[\Delta(n_i, n_j) - \Delta_{i,j} > \gamma] \leq exp\Big(\frac{-\gamma n_i n_j}{2\sigma_i^2 n_j + 2\sigma_j^2 n_i}\Big)$$

Proof
Let $Y = \Delta(n_i, n_j)$. Then $Y - \Delta_{i,j}$ is a $\sigma_i^2/n_i + \sigma_j^2/n_j$ subgaussian random variable. Using the suitable inequality from lemma 4 and a straightforward calculation we get the desired bound. ■.

6.2 Pruning

7 Ideas

If we are not forced to sample on our walk, how does this change the problem? If we are forced to sample at a specfic set of nodes, how does this change the problem? If we prune nodes so that we don't need to sample, then we can create edges between boundary set of pruned nodes in G with weight the sp distance (may not change edges).

Forced Samples vs Nonforced Samples: Forced samples means deciding to play some arms will eliminate a section of the possible adaptive allocation rules.

Difference with Contextual Bandits: Contexts are decided by an adversary in contextual bandit. Not an offensive problem, a defensive problem.

8 Batched Bandit Linear Optimization

In this section, we will utilize the Bandit Online Linear Optimization (BOLO) algorithm of Abernethy et al [**?**] as a subroutine of a batched network bandit algorithm with provable regret bounds. Their result solves a related by more general problem than the network bandit, so the primary difficulty in using their algorithm as a subroutine is finding a bandit linear optimization problem.

The BOLO algorithm solves an *online linear optimization* problem, defined in [**?**] as the following repeated loss-minimization game:

At each step $t = 1$ to T,

- Player chooses action $\mathbf{x}_t \in \mathcal{K}$
- Adversary indepdently chooses $\mathbf{f}_t \in \mathbb{R}^n$
- Player suffers loss $\mathbf{f}_t^T \mathbf{x}_t$ and observes feedback $\mathcal{F}_t$

In the bandit form of this game, the feedback is only the linear loss function evaluated at the players choice, namely $\mathcal{F}_t = \mathbf{f}_t^T \mathbf{x}_t$. The player's goal is to minimize his regret

$$R_T := \sum_{t=1}^{T} \mathbf{f}_t^T \mathbf{x}_t - min_{\mathbf{x}* \in \mathcal{K}} \sum_{t=1}^{T} \mathbf{f}_t^T \mathbf{x}_t$$

As noted in [**?**], the player only need compete against the best fixed point in $\mathcal{K}$, a compact convex subset of $\mathbb{R}^n$.

8.1 A Sketch: Equivalence to the Fully Stochastic Network Bandit

As a primary challenge in developing this algorithm we need to design a convex set $\mathcal{K}$ whose solutions can be interpreted as valid network bandit play(s). From [**?**], we know we can associate a vector in $\mathbb{R}^n$ with each discrete possibile prediction of the player, form $\mathcal{K}$ from the convex hull of these possible prediction vectors. We then work with the convex hull of the discrete set while predicting the original set.

The network bandit problem introduces a wrinkle, which we must be careful to smooth out: the linear optimization problem allows any action to be chosen at each t, while the available actions to a network bandit player depend entirely on the previous action. In order to reconcile these two concepts, we take advantage of the fact that the linear functions $\{\mathbf{f}_t\}_{t=1}^T$ can be arbitrarily selected a priori, and the regret bound of the BOLO algorithm still holds. This guarantee guarantees BOLO to solve the adversarial problem with reget of order $O(poly(n)\sqrt{T})$. To transform the problems and achieve $O(log(T))$ regret, we can solve $O(log(T)$ linear optimization problems using BOLO algorithm, where the loss for each optimization problem is close to the loss from one batch of an algorithm using BOLO as a subroutine. As a bonus, this allows us to solve network bandits where switching costs and rewards are coupled random variables. This is formalized below.

8.2 Setup

In order to reduce unnecessary analysis, we will solve the loss version of the network bandit problem, where traversing an edge $(i,j) \in E$ reveals an i.i.d. sample of $l_{i,j} = w_{i,j} - X_j$. Minimizing loss in this setting is equivalent to maximizing reward of the network bandit, but where we can only get (noisy) differential estimates of switching costs and arm rewards after sufficient exploration.

Horizon DAG In order to fix the optimization problem, we will construct a short-length DAG which includes all paths through our graph. For a directed graph $G = (V, E)$, denote $Hor(G)$ over horizon h as the graph constructed by making h copies of $V(G)$, where the ith copy of v is v_i. Add edges $(v_i, w_{i+1}) \in Hor(G)$ if $(v, w) \in E$ and $i < h$. Add a source node s and sink node t and edges $\Cup_{v \in V}(s, v_1) \cup (v_k, t)$. There is a obvious bijection between each walk with h edges in G a walk with $h+2$ edges in $Hor(G)$.Denote the number of edges in the DAG $|E(Hor(G))| = \mathcal{E} = |E|(h-1) + 2|V|$. In vector form, each s,t-path in $Hor(G)$ can be represented by a vector $\mathbf{x} \in \{0,1\}^{\mathcal{E}}$.

8.3 Online Linear Optimization

By taking $\mathcal{K}$ as the convex hull of all such path vectors in $Hor(G)$, which is known to be the set of flows in $Hor(G)$, we now have the convex set over which to optimize an unknown linear function sequence $\{\mathbf{f}_t\}_{t=1}^T$. Like section 7 in [**?**], we will be repeated solving the bandit shortest s,t-path problem on $Hor(G)$.

The following lemma indicates that we can construct such a linear function sequence which will relate the optimal solution of the linear optimization problem on $Hor(G)$ to the reward of the oracle in the original network bandit graph G.

Lemma 6 *Assume a network bandit B is non-degenerate and satisfies IPA with initial node v_0, and $l_{i,j} \in [-1,1], \forall (i,j) \in E$. Then there exists a sequence of linear functions $\{\mathbf{f}_t\}_{t=1}^T$, a convex set $\mathcal{K}$ such that*

$$min_{\mathbf{x}^* \in \mathcal{K}} \sum_{t=1}^{T} \mathbf{f}_t^T \mathbf{x}_t = g(R^*(v_0))$$

for linear g which depends only $|V|$ and $R^*(v_0)$ the expected network reward of the oracle in B.

The proof can be found in the appendix.

9 Appendix

9.1 Proof of Lemma 6

Fix v_0. The oracle will play arms over the walk W^* starting at v_0. Let $S^* = \sum_{e \in W^*, e \neq e^*} l_e$, the total of the oracle losses over any edges played which are not the optimal edge.

Let $\mathcal{K}$ be the convex hull of the set of s,t-paths in $Hor(G)$, which is the set of flows in *graph*.

References

2]*P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," Mach. Learn., vol. 47, no. 2–3, pp. 235–256, 2002. D. Pollard, Convergence of Stochastic Processes. Berlin, Germany: Springer, 1984. Missing https://ocw.mit.edu/courses/mathematics/18-s997-high-dimensional-statistics-spring-2015/lecture-notes/MIT18$_S$997S15$_C$hapter1.pdf*