Scalable Model-Free Algorithms for Influencer Marketing

Alexandra Iacob

université PARIS-SACLAY

Scalable Model-Free Algorithms for Influencer Marketing

Apprentissage séquentiel pour la diffusion d'information

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Graduate School : Informatique et sciences du numérique. Référent : Faculté des sciences d'Orsay

Thèse préparée dans la unité de recherche
Laboratoire interdisciplinaire des sciences du numérique (Université Paris-Saclay, CNRS),
sous la direction de **Bogdan CAUTIS**, Professeur,
le co-encadrement de **Silviu MANIU**, Maître de Conférences

Thèse soutenue à Paris-Saclay, le 23 février 2023, par

Alexandra IACOB

Composition du jury

Membres du jury avec voix délibérative

Fatiha SAÏS Professeur, Université Paris-Saclay, LISN	Présidente
Stratis IOANNIDIS Professeur, Northeastern University (Boston, MA, USA), SPIRAL	Rapporteur & Examinateur
Michalis VAZIRGIANNIS Professeur, École Polytechnique, LIX	Rapporteur & Examinateur
Pablo PIANTANIDA Professeur, Université Paris-Saclay, ILLS, CNRS	Examinateur
Vincent TAN Professeur, National University of Singapore, IDS	Examinateur

THESE DE DOCTORAT
NNT : 2023UPASG012

To my parents

Acknowledgements

First, I want to thank my supervisors, Bogdan and Silviu, for the chance to work on this project. The knowledge they shared with me, their guidance, and their advice were invaluable and made this experience unique and enjoyable. They taught me how to approach research and always had ideas or literature to refer me to when I felt the solution I was searching for was not in sight.
I cannot thank my family and Guilhem enough for their love, constant motivation, support, and patience when it was difficult.
Many thanks to Olivier and Yoan for the interesting and fruitful discussions that contributed to the first solution presented in this thesis. Thanks to Benoît and Taha for collaborating with me on the deep reinforcement learning research direction.
I am grateful to have met wonderful people in the laboratory, whose professionalism and character I admire. I was fortunate to form friendships with Armita, Yuting, Lucas, and Taha, which I hope will last forever.

Contents

List of Figures

List of Tables

Résumé long

Motivés par les scénarios de diffusion de l'information et de publicité dans le les réseaux sociaux, nous étudions un problème de maximisation de l'influence (MI) dans lequel on suppose que l'on en sait peu sur le réseau de diffusion ou sur le modèle qui détermine comment l'information peut se propager. Dans un tel environnement incertain, on peut se concentrer sur des campagnes de diffusion à plusieurs tours, avec l'objectif de maximiser le nombre d'utilisateurs distincts qui sont influencés ou activés, à partir d'une base de nœuds influents. Au cours d'une campagne, les graines de propagation sont sélectionnées séquentiellement lors de tours consécutifs, et les commentaires sont collectés sous la forme des nœuds activés à chaque tour. L'impact (récompense) d'un tour est alors quantifié par le nombre de nœuds nouvellement activés. En général, il faut maximiser la propagation totale de la campagne, comme la somme des récompenses des tours. Nous considérons deux sous-classes de d'IM, *Contextual Influence Maximization with Persistence* (CIMP) et *Episodic Contextual Influence Maximization with Persistence* (ECIMP), où (i) la récompense d'un tour d'une campagne en cours consiste uniquement en de nouvelles activations (non observées lors des tours précédents de cette campagne), (ii) le contexte du tour et les données historiques des tours précédents peuvent être exploités pour apprendre la meilleure politique, et (iii) ECIMP est CIMP répété plusieurs fois, ce qui permet d'apprendre également des campagnes précédentes. Ce problème est directement motivé par les scénarios du monde réel de la diffusion de l'information dans le marketing d'influence, où (i) seule la première / unique activation d'un utilisateur cible présente un intérêt (et cette activation persistera comme une activation acquise, latente, tout au long de la campagne). (ii) de précieuses informations secondaires sont disponibles pour l'agent d'apprentissage. Dans ce contexte, une approche d'exploration-exploitation pourrait être utilisée pour apprendre les principaux paramètres de diffusion sous-jacents, tout en exécutant les campagnes. Pour CIMP, nous décrivons et comparons deux méthodes de bandits à bras multiples contextuels, avec des limites supérieures de confiance sur le potentiel restant des influenceurs, l'une utilisant un modèle linéaire généralisé et l'estimateur de Good-Turing pour le potentiel restant (`GLM-GT-UCB`), et l'autre adaptant directement l'algorithme LinUCB à notre cadre (`LogNorm-LinUCB`).

Afin d'estimer le potentiel restant des influenceurs dans un contexte, `GLM-GT-UCB` modifie l'estimateur Good-Turing avec une fonction externe du contexte de la ronde, du vecteur caractéristique de l'influenceur et du nombre de sélections dans la campagne. Chaque vecteur est a priori inconnu et estimé à partir des feedback reçus à la fin de chaque tour. Les intervalles de confiance des estimateurs reposent sur l'hypothèse que la distribution sous-jacente du nombre de nœuds influencés est une distribution de Poisson. Selon l'hypothèse log-normale, les paramètres inconnus sont estimés par rapport à l'échelle des récompenses et des garanties théoriques pour les estimateurs à l'échelle logarithmique. Pour ECIMP, nous proposons l'algorithme `LSVI-GT-UCB` qui implémente le principe d'optimisme face à l'incertitude pour l'apprentissage par renforcement, avec approximation linéaire. L'agent d'apprentissage estime pour chaque nœud de départ son potentiel restant avec un estimateur de Good-Turing, modifié par une fonction Q estimée. Nous avons également présenté une solution Deep Reinforcement Learning pour ECIMP, qui estime la fonction Q de chaque influenceur à l'aide d'intégration de nœuds. Nous montrons qu'ils surpassent les performances des méthodes de base utilisant les idées les plus récentes, sur des données synthétiques et réelles, tout en présentant un comportement différent et complémentaire, selon les scénarios dans lesquels ils sont déployés.

Chapter 1

Introduction

Social media advertising is a booming domain, gradually replacing advertising over traditional channels (TV, radio, print, mail). It is enabled by the highly effective word-of-mouth mechanisms that are embedded in social applications, such as likes, shares, reposts, or notifications. Social networking applications are therefore an unprecedented medium for advertising, be it with commercial intent or not, as products, news, ideas, political manifests, etc., can propagate easily to a large yet well-targeted audience.

But the interest of social media for marketers is not only that it enables them to easily and rapidly reach a large user base, but also, and foremost that it brings credibility to the messages that are being conveyed. Indeed, many studies show that people are more inclined to pay attention to a message or referral from a known individual, e.g., a friend or an influencer whom she follows [Ewing (2019); Buchin (2015)].

Motivated by advertising in social media, the class of algorithmic problems under the generic name of *Influence Maximization* [Kempe et al. (2003)] encompasses all scenarios that aim to maximize the spread of information in a diffusion network, by identifying the most influential nodes from which the diffusion of a specific message should start. Influence Maximization mirrors an increasingly used and highly effective form of marketing in social media, targeting a sub-population of *influential people*, instead of all users of interest, known as *influencer marketing* [Brown and Fiorella (2013)]. It is one of the most studied problems in the literature due to its applicability not

only to viral marketing [Chen et al. (2010)], but also to ad placement [Tang and Yuan (2016)], or personalized recommendation [Song et al. (2006); Guo et al. (2013)].

Influence Maximization usually has as objective the *expected spread* under a stochastic diffusion model, which describes diffusion as a probabilistic process. From a simple perspective, given a diffusion network represented as a directed, weighted (probabilistic) graph $G = (V, E, p)$, the Influence Maximization problem to solve is that of finding L seed nodes (influencers), from which to initiate an information diffusion process, with the objective of maximizing the number of influenced (activated) nodes, i.e., *the reward*. The seminal work of Kempe et al. (2003) proposed two models for the information diffusion process, the Independent Cascade model and the Linear Threshold one. Under the former model, the process develops in discrete steps, starting with the selection of L seed nodes, where at each step the newly activated nodes attempt to influence and activate their neighbors, succeeding with a probability $p_{ij}, \forall (i, j) \in E$. In the Linear Threshold model, at any step in the diffusion process, a node becomes active if the sum of the weights of its active incoming neighbors is above that node's own activation threshold. Both models have been adopted by most of the literature, see the survey of Li et al. (2018). However, under both diffusion models, the Influence Maximization problem has been shown to be NP-hard, reducing to the Set-Cover problem and the Vertex-Cover problem respectively [Karp (1972)], see Theorems 2.4. and 2.7 of Kempe et al. (2003).

Approximation algorithms that exploit the objective's monotonicity and sub-modularity have been studied extensively, yet scaling Influence Maximization to realistic graphs remains difficult. While most of the Influence Maximization literature focuses on improving efficiency and scalability – see the benchmarks Arora et al. (2019, 2017), other major obstacles have limited the practical impact of this research. First, it is hard to obtain meaningful influence probabilities, as it is hard and data-intensive to learn them from past diffusions [Hu et al. (2019); Gomez-Rodriguez et al. (2012); Du et al. (2013)]. Also, the effectiveness of most Influence Maximization algorithms depends on diffusion models and their key parameters – whether known or learned in an online manner – aspects that are most often hard to align with real-life diffusion dynamics. It is commonly agreed that such parametric diffusion models represent elegant yet coarse interpretations

of a reality that is complex and uncertain.

For these reasons, the focus of the Influence Maximization literature has shifted recently towards *online* and *diffusion-model independent* methods [Lagrée et al. (2018); Vaswani et al. (2017b); Wen et al. (2017); Lei et al. (2015a)] where, during a *multi-round influence campaign*, a learning agent sequentially selects at each round seeds from which a new diffusion of the campaign's message is initiated and observed in the network. A round's feedback is then used to update the learning agent's knowledge. To balance between exploration (of uncertain aspects of the diffusion environment) and exploitation (e.g., focusing on the most promising seeds), such methods rely on sequential learning, either without state information – *Multi-Armed Bandits* [Thompson (1933)], or with state – *Reinforcement Learning*. In this way, *Online Influence Maximization* (OIM) over multiple rounds allows dealing with problem settings having (partially) unknown diffusion specifications (i.e., network and/or diffusion model). Therein, starting from a known base of few influential nodes, one can discover and *learn* the diffusion environment while maximizing spread, over multiple rounds of a learn & spread campaign. Consequently, the Influence Maximization objective shifts from a per-diffusion (round) one to a per-campaign one, e.g., by maximizing the total number of activations or, alternatively, the total number of *distinct* activations.

Lagrée et al. (2018) introduced the *Online Influencer Marketing (or Influence Maximization) with Persistence (OIMP)* problem, with the particularity that the reward consists of only the *new/distinct activations*; a (basic) user activated in one round remains activated for the rest of the rounds. In other words, only a target user's first activation is of interest and, once acquired, it will *persist*; e.g., as in political endorsements or subscriptions to a media service, hence the persistence notion. This variation of the OIM problem reflects the real-world scenarios where even though a user's activations may influence the spread of information, it is only its first activation that is counted as part of the reward; e.g., an influencer Bob may sway a user Alice to sign a petition (activate) and share its link, the signature (activation) is counted only once, but the link may be shared multiple times by Alice at the influencer's impel, enhancing the chances for other new activations.

This type of reward exhibits the *diminishing returns* property, i.e., the expected number of new activations for an influencer decreases with each of its selections as the seed node in a diffusion

process. The expected spread is not constant over the entire campaign as in most OIM problems, thus the learning agent must make decisions based on the influencer's expected *remaining potential* of activating new users. Because the diffusion environment is unknown, this quantity is also unknown a priori, so the learning agent must estimate it from the feedback received from the environment. The Good-Turing estimator [Good (1953)] has been proven to be efficient for unseen unique entities [McAllester and Schapire (2000); Bubeck et al. (2013b); Lagrée et al. (2018)]. More in-depth details of how this estimator may be used in an OIMP context are provided in Chapter 2.

Besides the OIM problems where the seeds are selected by an online approach [Lei et al. (2015a)], instead of the classical select-then-spread offline one, or where the diffusion process may be repeated over multiple rounds (a diffusion campaign), with the objective being the cumulative new activations [Lagrée et al. (2018)], many other instances of the Influence Maximization problem have been considered in recent years, for diverse problem settings, application scenarios, or performance objectives. E.g., the diffusion network may be a bipartite graph, modeling any-path diffusion from influencers to target nodes [Alon et al. (2012)], or the diffusions may be topic-aware [Aslay et al. (2014); Chen et al. (2015)].

Inspired by these works, we study in Chapter 3 an OIMP problem in which the diffusion topology, the influence probabilities, and the model that determines how information may spread are all assumed to be unknown. Instead, what is known are the potential spread initiators, a set of few influential nodes called hereafter the *influencers*. In such a highly uncertain environment, under *budget limitations* (number of seedings and number of rounds), the diffusion campaign aims to maximize the number of *distinct* users that are influenced or activated, starting from the influencers. Seeds are selected sequentially (at each round) among the influencers, and an influencer may be re-seeded, i.e., selected at multiple rounds. After a round's diffusion, the assumed feedback is all the activated nodes from that round, i.e., only the diffusion's effects are observed (the *who*), but not their causes (the *why*). Generically, this feedback is used to refine the estimations for the influencers' remaining spread potential, which will guide future seeding decisions. Matching the overall objective, a round's *reward* is the number of *newly activated nodes*, i.e., those that

were not already activated at previous rounds. The campaign's objective is to maximize the sum of rounds' rewards. Regarding the influencers' aptitude to spread information and their remaining potential, we are assuming that contextual information is known and exploitable, as features of influencers or of the information being diffused. The intuition is that within a campaign, whose overall goal is to get a specific "message" to as many users as possible, the ways in which that message may be formulated, presented, or diffused may vary from round to round, and such contextual variations will lead to different propagation dynamics. E.g., the campaign's message may be a political manifesto, while the to-be-diffused items may pertain to different aspects thereof, to connections with societal issues, or it may be framed in news, op-eds, data analysis, multimedia content, etc. Thus, we consider an OIMP problem setting that connects many of the recently considered Influence Maximization assumptions for practical purposes, namely (i) an unknown diffusion medium and an influencer-target node bipartite graph abstraction thereof, (ii) diffusion model-independent spread over multiple rounds of an influence campaign, (iii) topic/context dependent diffusions, and finally (iv) a spread objective given by the campaign's total number of *distinct* activations.

We formally define this problem, *Contextual Influence Maximization with Persistence* (CIMP), in Chapter 3. In short, CIMP is the optimization problem of choosing a number of $1 \leq L \leq K$ influencers from a given set to be activated in each of the budgeted T rounds, with the objective of maximizing the cumulative spread at the end of all rounds, assuming that the probability that influencer k activates basic node j depends on the round's context, the number of k's selections within that campaign, and the node's j unknown inner probability of activating itself. We stress that this is directly motivated by (but not limited to) the real-world scenarios of information diffusion in *influencer marketing*, where (i) the diffusion medium is highly uncertain and only a few influencer nodes may be known in advance, (ii) only a target user's *first* activation is of interest and, once acquired, it will *persist*, and (iii) valuable side-information may be available to the learning agent.

To solve the CIMP problem, we extend the recent state-of-the-art research which has considered Topic-Aware / Contextual Influence Maximization in uncertain environments, by sequential

learning approaches formulated in terms of Contextual Multi-Armed Bandits [Chu et al. (2011); Agarwal et al. (2014); Slivkins (2019); Lattimore and Szepesvári (2020)], assuming that contextual information is known and exploitable in the learning process. Contextual Multi-Armed Bandits are rather versatile approaches, providing both formal guarantees and effective approximation algorithms. Our solutions to the CIMP problem, presented in detail in Chapter 3, are based on them and proven theoretically and empirically to be effective.

However, Contextual Multi-Armed Bandits fail to capture many realistic scenarios, where actions are influenced by other factors than the context and the previously sampled i.i.d. reward random variables. In particular, for information diffusion with persistent activations – i.e., where only the new activations are rewarded – repeatedly selecting the same seemingly optimal seed node(s) may lead to non-optimal policies for seed selection. This important aspect may be captured by a modifier to the estimated remaining potential for Contextual Multi-Armed Bandits, or by Reinforcement Learning states, thus preserving the i.i.d. requirement for the rewards. To enable decision-making based on the seed's number of selections, in Chapter 4 we propose a sequential learning method based on *Episodic Reinforcement Learning*, called *Episodic Contextual Influence Maximization with Persistence* (ECIMP).

In short, *Episodic Contextual Influence Maximization with Persistence* (ECIMP) is the optimization problem extending CIMP to multi-campaign diffusions for a budget of T campaigns, each consisting of H rounds, the objective being to maximize the cumulative spread at the end of all the campaigns. The novelty of an activation and the selections of an influencer are reset at the beginning of each campaign, which means that an ECIMP campaign proceeds just like a CIMP one, but can benefit from information learned during the previous campaigns.

Contribution. For the CIMP problem, we propose two algorithms based on Upper Confidence Bound, `GLM-GT-UCB` and `LogNorm-LinUCB`, and one algorithm `LSVI-GT-UCB` for the ECIMP problem. Both problems address selecting influencers in advertising campaigns, where newly activated nodes make up the reward. The algorithms follow *optimism in the face of uncertainty* in sequential learning [Bubeck and Cesa-Bianchi (2012)], deriving an Upper Confidence Bound on the estimator of the remaining spread potential of each influencer. This enables us to alternate

in a principled way between exploration and exploitation steps when taking seeding decisions at the campaign's rounds. Our solutions are diffusion-model agnostic and follow different assumptions on the rewards distribution: Poisson for `GLM-GT-UCB` and `LSVI-GT-UCB`, log-normal for `LogNorm-LinUCB`.

In Chapter 3, published in Iacob et al. (2022), we formally introduce the CIMP problem, the solution based on Upper Confidence Bound using Poisson distribution assumption, i.e., `GLM-GT-UCB`, and the LinUCB-based solution that assumes a log-normal distribution, i.e., `LogNorm-LinUCB`. `GLM-GT-UCB` uses a Good-Turing estimator [Good (1953); Bubeck et al. (2013a)] for new activations, to which it applies an external factor function modeling an influencer's fatigue (diminishing returns) and potential in a given context. The parameter of the external factor is assumed to be a linear combination of the context and an unknown feature vector learned through linear regression. Theoretical guarantees are provided under the Poisson distribution of rewards assumption. `LogNorm-LinUCB` assumes a linear structure for the scale of rewards, estimated by the inner product of the context and the influencer's learned feature vector, and it has theoretical guarantees at a logarithmic scale.

In Chapter 4, recently submitted for publication, we formally introduce the ECIMP problem, which allows a solution based on Reinforcement Learning and Upper Confidence Bound – `LSVI-GT-UCB`. With respect to CIMP, the focus moves here from one of sequential learning during a single campaign with multiple rounds to one of learning from *multiple campaigns* with multiple rounds. Not only each influencer's number of selections can inform the decision-making process, but also the historical data from previous campaigns can be exploited for the learned policy. In the Reinforcement Learning terminology, a diffusion *episode* will be the equivalent of a diffusion campaign, and its *horizon* will be the equivalent of the number of rounds.

Due to the contextual information provided by the environment at the beginning of each round, the state space may be extremely large. For such cases, the recent works of [Wang et al. (2019, 2020); Jin et al. (2020)] have successfully used (generalized) linear function approximations to estimate the value function or the policy, a direction we also adopt here.

If we assign to each potential seed node its own episodic Markov Decision Process, then the

value function in each state represents the respective seed's remaining potential. Assuming that an activation's novelty is relative to one campaign, and knowing that the Good-Turing estimator is just an average of new activations, we use the historical data from previous campaigns for computing the average Good-Turing estimator for the round over the episodes.

In summary, in this thesis, we formally describe two Online Influence Maximization problems under uncertainty, namely the CIMP and ECIMP problems, jointly drawing motivation from several up-to-now disjoint recent studies in the area of Online Influence Maximization, for learning seed selection policies in unknown diffusion environments. We propose the algorithms `GLM-GT-UCB`, `LogNorm-LinUCB`, and `LSVI-GT-UCB`, which implement the *optimism in the face of uncertainty* principle for Multi-Armed Bandits, and Episodic Reinforcement Learning with linear approximation, respectively.

We experimented with synthetic and real-world data, comparing with state-of-the-art solutions adapted to our problem. The experiments show that our methods successfully learn from the available side information and achieve higher cumulative rewards. These results are complemented by theoretical *regret* guarantees for a `LogNorm-LinUCB` variant that learns from independent samples.

We believe that our studies can help in bridging the gap between Influence Maximization research and practical scenarios for information diffusion and advertising in social media, by proposing effective solutions for scenarios of information diffusion where (i) the diffusion medium is highly uncertain and only a few influencer nodes may be known in advance, (ii) activated users (instead of activations) represent the objective to be maximized, and (iii) side-information (contextual information) about the influencers' aptitude to spread information can be exploited by the learning agent.

The relevant papers are:

1. *Contextual bandits for advertising campaigns: A diffusion-model independent approach.* In Proceedings of the 2022 SIAM International Conference on Data Mining, SDM 2022.
2. *Sequential Learning Algorithms for Contextual Model-Free Influence Maximization.* In Proceedings of the 29th ACM SIGKDD Conference on knowledge discovery and data mining, KDD 2023.

Chapter 2

Preliminaries

In this chapter, we establish the basic theoretical foundations for our contributions presented in Chapters 3 and 4 and in Appendix A. In Section 2.1 we give a general definition for graphs, the Influence Maximization and Online Influence Maximization problems, and two well-known diffusion processes, and justify our choice for diffusion-model independent approaches. In Section 2.2 we present a short history of Multi-Armed Bandits, the most known application scenarios, and proceed with introducing the most common classes of problems. We explain how an Online Influence Maximization problem is an instantiation of the Multi-Armed Bandit problem. We continue to introduce the type of reward we use in our work, i.e., *new activations*, inspired by Online Influence Maximization with Persistence. Before presenting the Upper Confidence Bound algorithms in Section 2.3, we mention a few simple heuristics and Thompson Sampling. We then present the intuition behind the Upper Confidence Bound and its advantageous handling of the exploration-exploitation trade-off, along with solutions based on Upper Confidence Bound for the Multi-Armed Bandit and Influence Maximization problems introduced previously. The persistence in the reward used by CIMP and ECIMP can be intuitively captured by Good-Turing estimators. Therefore, in Section 2.4 we present the estimator in its original design context and then in the various ways it has been adapted for different problems, including the Influence Maximization problem. We conclude this chapter with an introduction to Reinforcement Learning, which is necessary for solving the ECIMP problem. After an overview of Reinforcement Learning

and Markov Decision Processes, we identify the challenge of a very large state space for Markov Decision Processes due to the large variety of possible contexts in ECIMP and choose the (generalized) linear function approximation for the Q-function. We also present Deep Reinforcement Learning, a different approach to solving ECIMP with Reinforcement Learning.

2.1 Diffusion Process Models

The *Influence Maximization* [Kempe et al. (2003)] problem aims to maximize the spread of information in a diffusion network, by identifying the most influential nodes from which the diffusion of a specific message should start. The diffusion network can be formally represented by a graph.

We use the definitions of Trudeau (2013) for graphs, as follows: a graph G is an object of two sets called its *nodes set* V and its *edge set* E; a set is a collection of distinct objects, none of which is the set itself. The node set is a finite nonempty set. The edge set may be empty, but otherwise, its elements are two-element subsets of the nodes set.

The Influence Maximization problem is thus formalized as the optimization problem of selecting the subset $I \subseteq V$ of seed nodes which maximizes the expected number of activated nodes at the end of the diffusion process started from the selected seed nodes:

Problem 1 (Influence Maximization)**.** *Given a graph $G = (V, E)$ with the nodes V and the edges E, the objective is to solve the following optimization problem:*

$$\underset{I \subseteq V, |I| = L}{\operatorname{argmax}} \mathbb{E}\left|S(I)\right|,$$

where $S(I)$ is the spread started from the seed nodes in set I of size L.

Online Influence Maximization (OIM) extends Influence Maximization by repeating the diffusion process for T rounds, in each round t choosing the seed nodes $I_t \subseteq V$ and observing the spread $S(I_t)$ started from them.

Problem 2 (Online Influence Maximization)**.** *Given a graph $G = (V, E)$ with the nodes V and the*

edges E, a budget of T rounds, the objective is to solve the following optimization problem:

$$\underset{I_t \subseteq V, |I_t|=L, 1 \leq t \leq T}{\operatorname{argmax}} \mathbb{E} \left| \bigcup_{1 \leq t \leq T} S(I_t) \right|,$$

where $S(I_t)$ is the spread, i.e., number of activations, of the chosen seed nodes for round t.

Kempe et al. (2003) introduced two stochastic diffusion models generally assumed in the literature when designing solutions for the Influence Maximization problem Li et al. (2018), the Linear Threshold and the Independent Cascade models. The activation is carried out in discrete steps.

The Linear Threshold model assigns an activation threshold $\theta_v, \forall v \in V$ to every node in the graph, and an influence weight $b_{v,w}, \forall v, w \in V$ of node w over node v. The threshold is chosen uniformly at random from $[0, 1]$, and the influence weights must satisfy $\sum_{\text{w active neighbor of v}} b_{v,w} \leq 1$. The process starts with only the chosen seed nodes being activated. At every subsequent step t in the diffusion process, for every node v it is checked if the sum of all the weights of its active neighbors is larger than the threshold θ_v. The node becomes active when this condition is met:

$$\sum_{\text{w active neighbor of v}} b_{v,w} \geq \theta_v.$$

The process stops when no more nodes get activated.

The Independent Cascade model assigns an activation probability $p_{v,w}$ to every node in the graph. The diffusion process starts with only the chosen seed nodes being activated. At every step t, if a node v becomes active for the first time, it also has its only chance of attempting to activate its inactive neighbor w with a probability of $p_{v,w}$. If several neighbors of node w are active in step t, the model does not specify any order of their attempts to activate node w. The process stops when no more nodes get activated.

These diffusion process models are coarse generalizations of real diffusion processes, leaving out details, e.g., Independent Cascade does not specify the order in which newly activated nodes attempt to activate their neighbors. They are based on certain assumptions, e.g., the activation probabilities are uniformly distributed or all known a priori, which may not necessarily be true.

Both models assume that the structure of the graph is known to the learning agent. This is not necessarily the case in many settings.

The Influence Maximization problem remains NP-hard under either Independent Cascade or Linear Threshold diffusion process model. Under the Linear Threshold model it can be reduced to the vertex cover problem, while under the Independent Cascade model, it can be reduced to the set cover problem. Heuristics and approximation algorithms are used instead to solve this type of problem.

In our work, we simplify the solutions even more assuming that the diffusion network can still be represented by a graph, but without knowing its structure, and knowing only the set of potential seed nodes, i.e., influential, and another set of activated basic nodes, i.e., users. A bipartite graph is a graph whose nodes are separated into two disjoint and independent sets, and every edge connects a node in one set to one in the other set. A digraph is a graph whose edges have direction. The diffusion network can be viewed as a bipartite digraph, with the direction of edges following the direction of information spread, from influencers to basic users. The diffusion process model simply assumes that the influencers could activate the basic users and the learning agent only sees the end result. With this in mind, our solutions are actually independent of the diffusion model.

The formulation of the (Online) Influence Maximization problem in terms of Multi-Armed Bandits, with the influencers being the arms and the number of activations being the reward, offers the possibility of solving it with approximation algorithms based on Upper Confidence Bound, both of which are presented in the next two sections.

2.2 Multi-Armed Bandits

The Multi-Armed Bandit problem is a commonly used formalization of real-world scenarios where there is a learning agent that sequentially interacts with an environment. The decision maker, i.e., the learning agent, observes the results of each action over time, learns which action is more effective, and adjusts its decisions based on all the data available up to that point; the idea pre-

sented abstractly in Figure 2.1. Thompson (1933) introduced the Multi-Armed Bandit problem to study the exploration/exploitation trade-off in sequential decision problems, that is, the trade-off between choosing to explore with new actions and learning more about the environment or choosing to exploit the best action known so far to get the best result/reward, e.g., the number of cured patients in a clinical trial. Greedily choosing the action known to offer the best reward could be beneficial for a short period of time, while choosing to explore new actions could be beneficial in the long term as it increases the chances of discovering better actions. All this is due to the lack of complete knowledge about the environment.

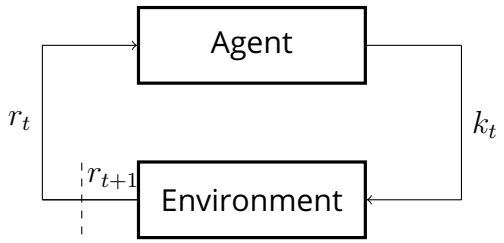

Figure 2.1: Multi-Armed Bandit.

Most known application scenarios for Multi-Armed Bandits are clinical trials [Thompson (1933); Berry (1978); Villar et al. (2015); Durand et al. (2018); Bastani and Bayati (2020)], recommender systems [Mary et al. (2015); Tang et al. (2014); Lagrée et al. (2016); Li et al. (2016, 2017); Wang et al. (2017); Herbster et al. (2021); Xie et al. (2021)], influence maximization [Lei et al. (2015b); Vaswani et al. (2017a); Wen et al. (2017); Lagrée et al. (2018); Li et al. (2020); Dong et al. (2022)]. The Multi-Armed Bandit problem can be applied to virtually any discrete-step sequential decision-making problem where a learning agent must choose the best action.

In more formal terms, the Multi-Armed Bandit problem involves sequential interactions of a learning agent with the environment for a horizon $T \in \mathbb{N}^+$, each interaction taking place in discrete steps called rounds. In each round t the agent chooses an action/arm $k_t \in \mathcal{K}, |\mathcal{K}| = K$, and the environment returns a reward $r_t \in \mathbb{R}$. Both the environment and the agent may randomize their decisions. Typically, the goal of the learning agent is to maximize the cumulative reward at the end of all rounds $\sum_{t=1}^{T} r_t$. To this end, the agent tries to learn the optimal policy $\pi^* \in \Pi, \pi^* : \{k_1, r_1, \ldots, k_{t-1}, r_{t-1}\} \rightarrow \mathcal{K}$ for choosing the most rewarding arm $k_t \in \mathcal{K}$ in each round $t \leq T$, where Π is the competitor class, i.e., a set of policies which includes the optimal

policy for all the environments in an environment class $\mathcal{E}$.

The cumulative reward for a policy $\pi \in \Pi$ can be converted into the learning agent's regret with respect to the optimal policy π^*. The regret $\mathcal{R}_T$ at the end of all T rounds is the difference between the expected cumulative reward for the optimal policy and the expected cumulative reward for the policy π used by the learning agent:

$$\mathcal{R}_T = \mathbb{E}\left[\sum_{t=1}^{T} \max_{k \in \mathcal{K}} r_{k,t}\right] - \mathbb{E}\left[\sum_{t=1}^{T} r_t\right],$$

where $r_{k,t}$ is the reward for choosing k in round t, and r_t is the reward for the arm chosen by policy π in round t.

Thus, the objective is either to maximize cumulative reward or, similarly, to minimize regret. The optimal policy depends on the environment with which the learning agent interacts. The environment is unknown, but assumptions can be made about its true nature to facilitate the learning process. Usually, the environment is assumed to belong to a certain class $\mathcal{E}$.

The *Adversarial* Bandits are a general type of bandit that makes very few assumptions about the reward generation process. The environment can be seen as an adversary because it can choose rewards based on the learning agent's algorithm and past decisions, to which it can also add independent randomization. The only way for the learning agent to minimize the cumulative regret $\mathcal{R}_T$ is to randomize its strategy such that the action k_t and the reward r_t are random variables, and to aim to not perform worse than an assumed set of constant policies Π. If the adversary doesn't use independent randomization, then the only source of randomness in the regret $\mathcal{R}_T$ is the one from the learner's actions. Auer et al. (2002b) obtain a regret of $O(T^{-1/2})$ with their algorithm *Exp3* for the adversarial bandit problem.

The *Stochastic* Multi-Armed Bandits work with environment classes that assume that the reward for any arm $k_t \in \mathcal{K}$ is distributed according to the same distribution law, e.g., Bernoulli, Uniform, Gaussian, etc., with each of these distribution types defining the respective environment class. For an assumed class, each arm k_t has its own reward distribution ν_{k_t} with its mean reward μ_{k_t}. If the bandit is also assumed to be *stationary*, then the reward distribution is defined for the chosen arm, regardless of the history of actions and rewards so far. In this thesis, we do

not study the *non-stationary* environments, i.e., environments that change over time (c.f. Chapter 31 from Lattimore and Szepesvári (2020)). Thus, the reward function can be assumed to be $r(k_t) = \mu_{k_t} + \epsilon_t, r : \mathcal{K} \to \mathbb{R}$, where ϵ_t is some normally distributed noise. However, the learning agent has no knowledge of the reward distributions ν_{k_t} and all it observes is each instantiation of the reward random variable $r(k_t)$ of the chosen arm k_t in round t. The regret for the Stochastic Multi-Armed Bandit is defined as follows:

$$\mathcal{R}_T = T \max_{k \in \mathcal{K}} \mu_k - \mathbb{E}\left[\sum_{t=1}^{T} r(k_t)\right] = T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} r(k_t)\right],$$

or

$$\mathcal{R}_T = \sum_{k \in \mathcal{K}} \Delta_k \mathbb{E}[n_k(T)],$$

where $\Delta_k = \mu_{k^*} - \mu_k$ is the sub-optimality gap, and $n_k(T) = \sum_{t=1}^{T} \mathbb{I}(k_t = k)$ is the number of arm k's selections. Since the agent learns from the observed rewards for its previous actions and adapts its decision-making process accordingly, the number of selections for each arm is also a random quantity.

The *Stochastic Linear* Multi-Armed Bandits assume that the expected reward is given by a linear function $r(k_t) = \langle \mathbf{k}_t, \theta \rangle + \epsilon_t$, where $\theta \in \mathbb{R}^d$ is an unknown parameter which gives the true reward in combination with the arm k's feature vector $\mathbf{k}_t \in \mathcal{K}^d \subset \mathbb{R}^d$, and ϵ_t is some noise. The learning agent tries to learn the unknown parameter θ so as to minimize regret:

$$\mathcal{R}_T = \mathbb{E}\left[\sum_{t=1}^{T} \max_{k \in \mathcal{K}^d} \langle \mathbf{k} - \mathbf{k}_t, \theta \rangle\right]$$

As with the more general case of *Stochastic* Multi-Armed Bandits, the *Linear* ones also assume that the reward is well captured by a function, but the learning agent only observes the instantiations of the reward random variable $r(k_t)$, which in turn brings randomness in the formula of regret.

The *Stochastic Contextual* Multi-Armed Bandits assume that the environment provides side-information $Y_t \in \mathbb{R}^d$ at the beginning of each round. This side-information, combined with each arm's feature vector forms the round's context $\phi(Y_t, k), \phi : \mathbb{R}^d \times \mathcal{K} \to \mathbb{R}^d$. The learning agent must

now choose the arm that best fits the context, i.e., maximizing the reward. The reward can be linearly represented by the function $r(Y_t, k) = \langle \phi(Y_t, k), \theta \rangle, \forall (Y_t, k) \in \mathbb{R}^d \times \mathcal{K}, \theta \in \mathbb{R}^d$. The agent's policy can be evaluated with the regret formula in the equation:

$$\mathcal{R}_T = \mathbb{E}\left[\sum_{t=1}^{T} \max_{k \in \mathcal{K}} (r(Y_t, k) - r(Y_t, k_t))\right].$$

The Stochastic Linear/Contextual Multi-Armed Bandit assumes that the reward is normally distributed. If there are other distributions that may fit better, then the *Generalized Linear Model* could be a better representation [see Filippi et al. (2010b)]. The reward is assumed to follow the rule $r(Y_t, k) = \mu(\langle \phi(Y_t, k), \theta \rangle) + \epsilon_t$, where μ is the inverse link function. Its definition depends on the presumed distribution, e.g., for Poisson the inverse link function is $\mu = e^{\langle \phi(Y_t, k), \theta \rangle}, \forall k \in \mathcal{K}, \theta \in \mathbb{R}^d, Y_t \in \mathbb{R}^d$.

The Influence Maximization problem 1 and the *Online* Influence Maximization problem 2 are instantiations of the Multi-Armed Bandit problem, since the set of arms $\mathcal{K}$ can be the potential seed nodes, and the reward can be the spread/number of activations.

The learning agent may not receive the reward directly, but feedback which can be used to determine the reward. Assumptions on the feedback provided by the environment can be as relaxed as expecting only the count of the activated nodes, i.e., *full-bandit* feedback, or stronger, as in the case of *semi-bandit* feedback. The *node semi-bandit* feedback is the set of activated node IDs, and the *edge semi-bandit* one is when the influence path is also known.

In the case of OIM, the reward is the number of activations, or the number of *new* activations for the Online Influence Maximization with Persistence (OIMP) problem [Lagrée et al. (2018)]. Hence, one can assume the reward to be normally distributed and defined as a linear function, respectively distributed according to a Poisson or Log-Normal distribution and defined according to the generalized linear model with the appropriate inverse link function. Furthermore, if side information is assumed to be available, then the problem falls into the category of Stochastic Contextual Multi-Armed Bandits. These, all linked, form the tools for our problem CIMP, detailed

in Section 3.

The Influence Maximization optimization problems of finding the optimal policy π^* are NP-hard. More feasible solutions involve approximating the expected cumulative reward for a policy π and choosing the best-known one. Expressing these problems as Multi-Armed Bandits enables the utilization of various known heuristics and approximation algorithms.

A simple heuristic is ϵ-greedy, i.e., choosing to explore with a probability ϵ by selecting the arm at random, otherwise exploiting the best arm known up to that round. Another heuristic is the Boltzmann exploration [Sutton et al. (1998)], i.e., the probability of employing influencer k in round t is proportional to an exponential function of the empirical mean of the reward of that influencer $p_{k,t} \propto e^{\eta_t \hat{\mu}_{k,t}}$, where $\hat{\mu}_{k,t}$ is the estimated expected reward for k in round t, and $\eta_t > 0$ is the learning rate. Cesa-Bianchi et al. (2017) proposed a variant that uses different learning rates for different influencers, with a distribution-dependent regret guaranteed to be of order $\frac{K \log^2 T}{\Delta}$, where $K = |\mathcal{K}|$ is the size of the set of seed nodes and Δ is the sub-optimality gap, and another distribution independent one of order $\sqrt{KT} \log K$. One of the basic approximation algorithms is Thompson Sampling, proposed by Thompson (1933), i.e., playing the arm with the highest estimated probability to be the best arm. Agarwal et al. (2017) proved Thompson Sampling to have a logarithmic expected regret for the stochastic Multi-Armed Bandit problem. Finally, we use algorithms based on Upper Confidence Bound [Auer et al. (2002a)], which implement *the principle of optimism in the face of uncertainty*, relying on theoretically determined confidence bounds for the estimated expected reward, and able to achieve a regret of $O(\log T)$.

2.3 The Upper Confidence Bound Algorithms

As previously explained, the learning agent begins its sequential interactions with the environment without much knowledge about the potential expected reward of each arm. With each interaction, it can better estimate the expected reward. However, each round, it is faced with the challenge of choosing between an arm that it knows has performed well so far, and other arms that it may not have played enough to have good insight into their performance. This is known

Algorithm 1 Upper Confidence Bound (UCB)

1: **Input:** arms $\mathcal{K}$, confidence δ.
2: **Initialization:** play each arm $k \in \mathcal{K}$ once.
3: **for** $t = K + 1, \ldots, T$ **do**
4: Play arm $k_t = \text{argmax}_{k \in \mathcal{K}}$ UCB$_k(t-1, \delta)$.
5: **end for**

as *Exploration - Exploitation trade-off*.

An elegant solution for balancing the exploration of new arms with the exploitation of the ones known to perform well is to build each arm's *Upper Confidence Bound* (UCB) index computed from its estimated expected reward and the corresponding estimator's confidence width (also known as the exploration bonus); with small probability $\delta \in (0, 1)$, the true expected reward is not within the confidence interval. The former implements the exploitation factor, while the latter implements the exploration one. Algorithm 1 presents the basis for most UCB algorithms. Typically, the learning agent starts with an initialization phase where it plays each arm once to get an initial estimate of their potential and then, for the rest of the rounds, chooses the arm with the highest UCB index.

Choosing the arm with the highest UCB index implements the principle of *optimism in the face of uncertainty*. This principle performs well in practice because an initial overestimation of the UCB of a suboptimal arm is corrected by playing that arm, gathering more data, and computing a progressively more accurate estimate. The confidence interval gets tighter around the estimated expected reward which in turn gets closer to the true expected reward of the arm. Thus, eventually, the optimal arm has an increased chance of being discovered.

In Figure 2.2 we present an example of an initial underestimation for the expected reward of an arm, k_2 here, but with a large enough confidence interval such that a UCB-based algorithm would choose this arm in order to explore it more; LCB stands for Lower Confidence Bound. Let's assume that k_1 and k_3 have been chosen in previous steps, and their expected reward is better estimated. Upon further exploration of k_2, it may be discovered that this arm's true potential is higher than initially estimated, e.g., Figure 2.3. In this case, the UCB of arm k_2 remains larger than the UCBs of the other arms, so the algorithm would continue to play the same arm, but this time

exploits the better-known one.

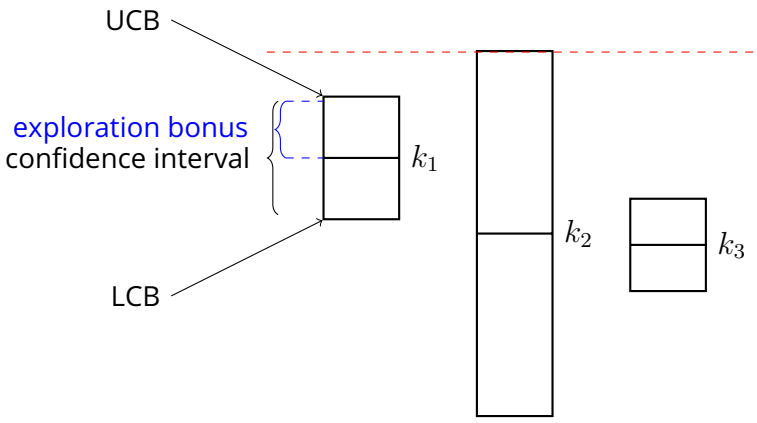

Figure 2.2: UCB example – under-explored a_2.

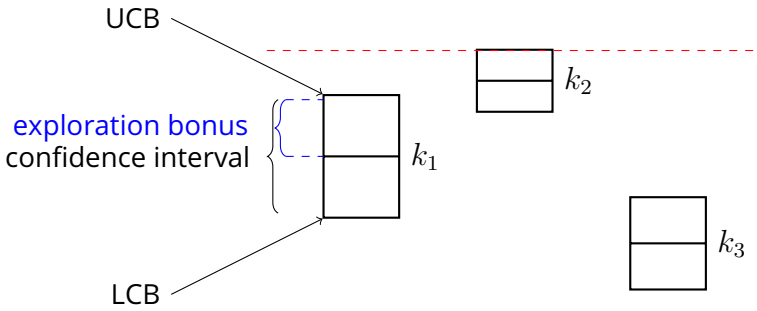

Figure 2.3: UCB example – high confidence for the expected reward from a_2.

Considering the case of Stochastic Multi-Armed Bandits, Auer et al. (2002a) in Theorem 1 and Lattimore and Szepesvári (2020) in Theorem 7.1 prove that the regret for an UCB algorithm is sublinear:

$$\mathcal{R}_T \leq \sum_{k:\mu_k<\mu^*} \frac{8\ln(T)}{\Delta_k} + \left(1+\frac{\pi^2}{3}\right)\left(\sum_{k\in\mathcal{K}}\Delta_k\right)$$

$$\mathcal{R}_T \leq \sum_{k:\mu_k>0} \frac{16\ln(T)}{\Delta_k} + 3\left(\sum_{k\in\mathcal{K}}\Delta_k\right)$$

Having a sublinear regret in the number of rounds T implies that the agent learns at a fast rate, and the distance from the optimal policy is gradually smaller because the increase in regret is smaller over time.

Stochastic Linear/Contextual Multi-Armed Bandits have `LinUCB` [Auer (2002)] as a base algorithm, with a regret of only $\tilde{O}(\sqrt{T})$. Inspired by it, other solutions were developed, Li et al. (2010a);

Chu et al. (2011); Li et al. (2017). Wen et al. (2017) solves an OIM problem with an UCB-type algorithm `IMLinUCB`, with regret polynomial in all quantities of interest.

2.4 The Good-Turing Estimator

Some problems solved by UCB-type algorithms need to estimate the potential to encounter *new* entities. This is also the case for our two problems, CIMP and ECIMP, as we are interested in estimating the potential of each influencer to activate new nodes. For a different problem but with similar requirements, Good (1953), in a groundbreaking work on population frequency estimators that still influences state-of-the-art work today, introduced an estimator for the total probability mass of the entities that are not in a sample. Among the proposed solutions are estimators for the population frequencies of each species being represented r times in a sample of size N, denoted by q_r. Its expected value is estimated by $\mathbb{E}[q_r] = \frac{(r+1)n_{r+1}}{Nn_r}$, where n_r is the number of distinct species that are each represented r times in the sample. More importantly, the total probability mass M_0 of the entities not in the sample is proved to be well estimated by the proportion of entities seen exactly once in the sample:

$$G_0 = \frac{n_1}{N}.$$

In linguistics, the elements seen exactly once are called *hapaxes*. This probability that the next sample belongs to a new species is also known as the missing mass.

McAllester and Schapire (2000) studied the convergence rate for the Good-Turing estimator G_0 for the missing mass, and computed a confidence interval for the actual missing mass:

$$P\left\{M_0 \leq G_0 + 2(\sqrt{2}+\sqrt{3})\sqrt{\frac{\ln 3/\delta}{N}}\right\} \leq \delta$$

Bubeck et al. (2013b) use the Good-Turing missing mass estimator for estimating the expert's probability of identifying *new* interesting items from an underlying population A. In terms of a Multi-Armed Bandit problem, the experts are the arms, and the reward is the number of new

items discovered by the chosen arm $k \in \mathcal{K}$. The goal is to maximize the cumulative count of such *hapaxes* $F(T)$:

$$F(T) = \sum_{x \in A} \mathbb{I}\left\{x \in \{X_{1,1}, \ldots, X_{1,n_{l,T}}, \ldots, X_{K,1}, \ldots, X_{K,n_{K,T}}\}\right\},$$

where $X_{k,n_{k,t}}$ is an i.i.d random variable for the interesting item given by the expert $k \in \mathcal{K}$ at its $n_{k,t}$'s selection in round t.

The proposed algorithm `Good-UCB` estimates in each round t the probability of each expert k to identify *new* items $\hat{R}_{k,n_{k,t-1}}$, calculates the UCB based on the actual probability confidence interval, and chooses the expert with the highest UCB index:

$$\begin{aligned} UCB_{k,t} &= \hat{R}_{k,n_{k,t-1}} + C\sqrt{\frac{\log 4t}{n_{k,t-1}}} \\ &= \frac{U_{k,n_{k,t-1}}}{n_{k,t-1}} + C\sqrt{\frac{\log 4t}{n_{k,t-1}}} \\ &= \frac{1}{n_{k,t-1}} \sum_{x \in A} \mathbb{I}\left\{\sum_{t=1}^{n_{k,t-1}} \mathbb{I}\{X_{k,t} = x\} = 1\right\} + C\sqrt{\frac{\log 4t}{n_{k,t-1}}}, \end{aligned}$$

where C is a tuning parameter.

The regret is proved to be of an order $\sqrt{T}$ the assumption of non-overlapping support for the arms [see Theorem 4 of Bubeck et al. (2013b)]. Under the same assumption, `Good-UCB` is close to an omniscient oracle strategy in terms of the waiting time $T(\lambda), \lambda \in (0, 1)$, which is the time at which every expert has the missing mass of interesting items less than λ for every expert [see Theorem 5 of Bubeck et al. (2013b)]. Empirically, the algorithm performs well without this assumption.

Lagrée et al. (2018) have recently used the Good-Turing estimator for the missing mass to estimate an influencer's remaining potential for activating new nodes in an influence diffusion campaign. Contrary to the problem treated in Bubeck et al. (2013b), an influencer can activate more than one new basic node in a round. The remaining potential is given by the equation:

$$R_k(t) = \sum_{u \in A_k} \mathbb{I}\left\{u \notin \bigcup_{s=1}^{t} S(s)\right\} p_k(u),$$

where A_k is the set of basic nodes reachable from influencer k, $S(s)$ is the spread observed at the end of round s, and $p_k(u)$ is the probability of basic node u to be influenced by k.

The Good Turing estimator of the remaining potential is given by the equation:

$$\begin{aligned}\hat{R}_{k,n_{k,t-1}} &= \frac{1}{n_{k,t-1}} \sum_{u \in A_k} U_{k,n_{k,t-1}}(u) \prod_{l \neq k} Z_{l,t-1}(u) \\ &= \frac{1}{n_{k,t-1}} \sum_{u \in A_k} \mathbb{I}\{X_{k,1}(u) = \cdots = X_{k,n_{k,t-1}}(u) = 0, X_{k,s}(u) = 1\} \\ &\cdot \prod_{l \neq k} \mathbb{I}\{X_{l,1}(u) = \cdots = X_{l,n_{l,t-1}}(u) = 0\}\end{aligned}$$

Given the real-world scenario where selecting too many times the same influencer might result in poor performance, Lagrée et al. (2018) propose also a Good-Turing estimator which captures the influencer's fatigue. This characteristic is implemented by applying a time-dependent decreasing function $\gamma(n_{k,t})$ to the fraction of hapaxes. Theoretically guaranteed confidence intervals are provided for both estimators [see Theorem 4.2 and C.4 Lagrée et al. (2018)]. The solutions perform well under various diffusion models compared to multiple baselines, on both synthetic and real-world datasets.

2.5 Reinforcement Learning

Reinforcement Learning (RL) is one of the basic types of machine learning, along with supervised and unsupervised learning. Reinforcement Learning is useful for real-world scenarios where the agent makes decisions through trial and error, and learns from its experience. The agent repeatedly interacts with an unknown environment with the objective of maximizing the cumulative reward received from that environment. We have seen in Section 2.2 that Multi-Armed Bandits fit the same application scenarios. In fact, Multi-Armed Bandits are a special case of Reinforcement Learning with horizon $H = 1$. In Reinforcement Learning, the agent interacts with the

environment in episodes with a certain horizon H.

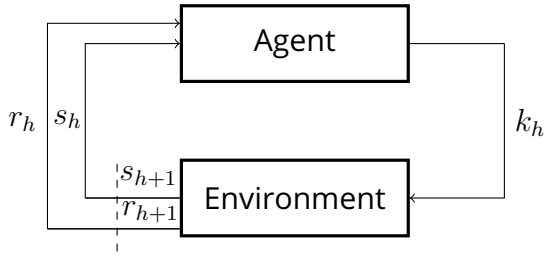

Figure 2.4: Reinforcement Learning.

Sutton and Barto (2018) define the environment as a finite automata entity that is in a certain state at a given time and changes its state upon receiving an action from a learning agent, while also returning a reward to that agent. Figure 2.4 is a rough representation of the dynamics in Reinforcement Learning. At each step h in the horizon, the agent is in a different state s_h. Assuming that it previously received the reward r_t, the agent makes the decision, e.g., choosing k_h, based on both the current state s_h and the last seen reward r_h. After this action, the environment moves to the next state s_{h+1} and returns the reward r_{h+1}; the process continues until the horizon is reached. The long-term reward is the sum of the immediate reward and the prospective reward from the new state. Thus, each state's value depends on the values of the states which can be reached from it. The learning agent takes an action for a state of the environment according to a policy it attempts to improve with each feedback/reward it receives from the environment. In contrast, the bandits have no state transitions. While bandits have the advantage of simplicity, Reinforcement Learning with state transitions can be more effective due to the possibility of using more information to improve the action selection policy. This motivates the definition of the ECIMP problem, and its solution `RL-GLM-GT-UCB`, with more details in Section 4.

The problem is usually represented as a *Markov Decision Process* (MDP) $M(\mathcal{S}, \mathcal{K}, \mathbb{P}, r)$. For episodic MDPs, an agent interacts with an environment in episodes of length $H \in \mathbb{Z}_+$, and for each episode the initial states s_1 are drawn from an initial distribution $\mu \in \mathcal{S}$, where $\mathcal{S}$ is the state space. The action space is $\mathcal{K}$, $\mathbb{P} = \{\mathbb{P}_h\}_{h=1}^H, \mathbb{P}_h : \mathcal{S} \times \mathcal{K} \to \Delta(\mathcal{S})$ are the state transition probabilities with $\Delta(\cdot)$ being the probability simplex, and $r = \{r_h\}_{h=1}^H, r_h : \mathcal{S} \times \mathcal{K} \to [0, 1]$ are the reward functions. MDPs assume that the current state and the taken action are sufficient

information to determine the next state.

The learning agent searches for the optimal policy. For an episodic MDP, the decision rule in a stationary (stochastic) policy is a function $\pi : \mathcal{S} \times [H] \to \Delta(\mathcal{K})$. The agent learns the best policy from the state values or state-action values. However, these values depend on the agent's decisions. Thus, the algorithm must compute and aims to improve at each step both the value function and the optimal policy.

For a finite horizon, the state value function is:

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{\tau=h}^{H} r_h(s_\tau, \pi(s_\tau, \tau))|s_h = s; \pi\right], \text{where } V_h^\pi : \mathcal{S} \to \mathbb{R},$$

and the optimal state value is $V_h^* = \max_\pi V_h^\pi$.

The state-action value function, also known as the Q-function, is:

$$Q_h^\pi(s,k) = \mathbb{E}\left[\sum_{\tau=h}^{H} r(s_\tau, \pi(s_\tau, \tau))|s_h = s, k_h = k; k_{h:H} \sim \pi\right].$$

The Bellman equations for a stationary policy $\pi, \forall s \in \mathcal{S}$:

$$V_h^\pi(s) = r_h(s, \pi(s)) + \sum_{\tau=h}^{H} \mathbb{P}_h(s_\tau|s, \pi(s,h)) V_{h+1}^\pi(s_\tau) \tag{2.1}$$

$$Q_h^\pi(s,k) = r_h(s,k) + \sum_{\tau=h}^{H} \mathbb{P}_h(s_\tau|s, \pi(s,h)) Q_{h+1}^\pi(s_\tau, \pi(s_\tau, \tau) \tag{2.2}$$

The Optimal Bellman equation is:

$$Q_h^*(s,k) = r_h(s,k) + \sum_{\tau=h}^{H} \mathbb{P}_h(s_\tau|s,k) \max_{k' \in \mathcal{K}} Q_h^*(s_\tau, k') \tag{2.3}$$

Equation 2.3 implies that the optimal policy π^* can be expressed as a greedy policy over the optimal state-value function:

$$\pi^*(s,h) = \operatorname*{argmax}_{k' \in \mathcal{K}} \left[r_h(s,k') + \sum_{\tau=h}^{H} \mathbb{P}_h(s_\tau|s,k')) Q_h^*(s,k') \right].$$

Thus, by estimating the optimal action-state value function the learning agent is able to estimate the optimal policy as well.

Equation 2.3 can be rewritten as the Optimal Bellman (update) Operator:

$$\mathcal{T}(Q_h)(s,k) = Q'_h(s,k) = r_h(s,k) + \sum_{\tau=h}^{H} \mathbb{P}_h(s_\tau | s_h = s, k_h = k) \max_{k' \in \mathcal{K}} Q_{h+1}(s_\tau, k'),$$

where $\mathcal{T} : (\mathcal{S} \times \mathcal{K} \to \mathbb{R}) \to (\mathcal{S} \times \mathcal{K} \to \mathbb{R})$.

This transformation enables the estimation of the state-action value function by applying Q-iteration, i.e., starting from a random Q-function, repeatedly updating it with $Q'_h = \mathcal{T}(Q_h)$, and using a greedy policy based on it. Another dynamic programming solution is to estimate the policy by applying policy iteration, i.e., starting from a random policy, computing the Q-function based on it, and repeatedly improving the policy based on the new values. Either approach has the shortcomings of requiring the transition probabilities and the reward function and being applicable only to tabular MDPs.

In this thesis, the ECIMP problem 4 in Section 4.2.1, poses the challenge of very large state space due to the nature of the context – the message diffused in a round can be virtually anything. The solution is approximate Reinforcement Learning. The options are either to approximate the value function, Equation 2.1 or 2.2, or to approximate the policy, $\pi : \mathcal{S} \times [H] \to \Delta(\mathcal{K})$. Estimating the Q-function requires more parameters to learn, but enables an easier computation of the value function $V_h^*(s) = \max_a Q_h^*(s,k)$ and of the policy $\pi^*(s,h) = \operatorname{argmax}_a Q_h^*(s,k)$.

There are different methods to approximate the Q-function. However, each of them requires certain assumptions for the MDP.

In order to approximate the Q-function with a linear function, the MDP is assumed to be linear in a feature map $\phi : \mathcal{S} \times \mathcal{K} \to \mathbb{R}^d$ [Bradtke and Barto (1996); Melo and Ribeiro (2007); Jin et al.

(2020); Yang and Wang (2020)]:

$$\begin{aligned} \mathbb{P}_h(\cdot|s,k) &= \langle \phi(s,k), \mu_h(\cdot) \rangle, \\ r_h(s,k) &= \langle \phi(s,k), \theta_h \rangle, \\ Q_h^\pi(s,k) &= \langle \phi(s,k), w_h^\pi \rangle, \end{aligned} \tag{2.4}$$

where d is the dimension of the feature space, $\mu_h \in \mathcal{S}^d, \theta_h \in \mathbb{R}^d$, and $w_h^\pi = \theta_h + \int_{\mathcal{S}} V_h^\pi(s) d\mu(s)$, $\forall h \in [1, H]$.

When the linearity assumption is too restrictive, the Q-function can be assumed to belong to a class of generalized linear models $\mathcal{G} = \{(s,k) \longmapsto f(\langle \phi(s,k), \theta_h \rangle) : \theta_h \in \mathbb{R}_d, \forall h \in [H]\}$, where $f : [-1,1] \longmapsto [-1,1]$ is the chosen inverse link function [Filippi et al. (2010b); Li et al. (2017); Wang et al. (2019)].

A specific branch of Reinforcement Learning of particular interest for us is *Deep Reinforcement Learning*. Deep learning is itself a type of machine learning, like Reinforcement Learning, this one using neural networks to transform an input, e.g., state and reward, into an output, e.g., next action. Deep Reinforcement Learning is well suited to our challenge of having a prohibitively large state space, often learning to represent the policy π as a neural network. Deep networks are able to generalize by the learning sub-structure. E.g., Andrychowicz et al. (2016) leveraged this power in a meta-learning problem, learning how to design optimization algorithms, i.e., learning an update rule for gradient descent. Khalil et al. (2017) proposed a framework for learning algorithms for different classes of problems, problems from the same class sharing the same structure. This framework is also applicable to the OIM problem.

Chapter 3

Contextual Influence Maximization with Persistence

In this chapter we introduce the *Contextual Influence Maximization with Persistence* (CIMP) problem which leverages available side-information for maximazing the cumulative count of new activations in a multi-round campaign starting from a set of seed nodes. We begin by setting the scene in Section 3.1 by presenting the state-of-the-art work related to each aspect of this problem. We then present in Section 3.2 the formal definition of the CIMP problem. Since the objective of the problem is cumulative new activations, we consider two separate assumptions about the reward distribution, Poisson and Log-Normal. We describe in Section 3.3 the solution based on the first assumption, `GLM-GT-UCB`, which estimates each influencer's remaining potential with a Good-Turing estimator and it adjusts it with an external factor which is a function of the round's context and the influencer's number of selections. `GLM-GT-UCB` is an UCB algorithm, with theoretically guaranteed upper confidence bound presented in short in Section 3.3.4, and extensively in Appendix A. In Section 3.4 we present the solution based on the assumption that the scale of the reward is normally distributed, `LogNorm-LinUCB`, with log-scale regret logarithmic in the number of rounds. The algorithms are also empirically proven to perform better than state-of-the-art methods on two real-world datasets and a synthetically generated one. The setup of the experiments and their results are discussed in Section 3.5.

3.1 Main Related Work

Dye (2000) highlights the importance of the choice of people to spread the information, as well as the potential of word-of-mouth in a successful marketing campaign. Domingos and Richardson (2001) further refine this type of viral marketing by exploiting the network value of the customers, i.e. the expected profit from further influencing potentially new customers. Their solution is to model together the inner expected profit from activating itself and the influence it has on others, which proved to be more efficient than traditional direct marketing in the domain of marketing motion pictures.

The work of Li et al. (2017) proposed a solution for the generalized linear contextual bandit problem, earlier considered also in a practical scenario of news recommendation [Li et al. (2010b)]. The solution is based on the work of Filippi et al. (2010b) – considering non-linear rewards for the MAB problem – and it improves it by adapting the algorithm of Auer (2002) to use Maximum Likelihood Estimation for estimating the unknown parameters, and uses the same approach to create the independent samples.

In Wen et al. (2017), the authors proposed an UCB-based algorithm, `IMLinUCB` for the online influence maximization problem in social networks. They assume that the diffusion of information follows the Independent Cascade (IC) edge semi-bandit model. The algorithm selects multiple influencers per round without suffering from an exponential increase in the combinatorial action space due to the cardinality of the source node set. Efficiency is obtained through the linear generalization of a probability weight function that yields the activation probabilities.

The cumulative regret bounds for `IMLinUCB` are topology dependent; this is also confirmed by the experiments performed on different types of graph topologies.

In Levine et al. (2017), the authors consider the weariness of an influencer's effectiveness over time and introduce the so-called *rotting bandits*. It assumes that the expected reward decays as a function of the number of times an arm has been selected, thus the optimal policy being one of choosing different arms. Our problem bears similarities to the non-parametric rotting bandit problem of Levine et al. (2017), as we also do not make assumptions about the structure of the

reward, but only about its non-increasing nature in the number of selections. To this end, Levine et al. (2017) proposed the `Sliding-Window Average (SWA)` algorithm. In the initialization phase, each arm is chosen a fixed number of times, and for the rest of the "campaign", their empirical average reward is adjusted by a given quantity. `SWA` is thus able to detect early the significantly sub-optimal arms while preserving theoretical guarantees.

The work that is most related to ours is Lagrée et al. (2018). Placed in a similar setting, it focuses on Online Influence Maximization with Persistence (OIMP). Lagrée et al. (2018) has a similar objective formulation, and proposes an algorithm called `GT-UCB` (for Good-Turing Upper Confidence Bound). The approach is inspired by the work of Bubeck et al. (2013a), which used the Good-Turing estimator in a setting where a learning agent needs to sequentially select experts that only sample one of their potential elements at each round. Similar to rotting bandits, an adaptation of `GT-UCB` (called `FAT-GT-UCB`) is considered for scenarios where influencers may experience fatigue, i.e., a diminishing tendency to activate their user base as they are re-seeded during a campaign. The key aspect that distinguishes our study from the one of Lagrée et al. (2018) is that *we assume contextual information is known and exploitable in the sequential learning process*, as features of the influencers or of the information being diffused. In doing so, we provide solutions that are no longer agnostic to the information being diffused nor to the profiles of influencers, as was the case in Lagrée et al. (2018). The contextual assumption leads to entirely different theoretical and algorithmic constructions and is supported by our empirical evaluation. `FAT-GT-UCB` is one of our experimental baselines.

Finally, we stress that in our bandit approach the parameters to be estimated throughout a campaign must capture how good an influencer still is (its *remaining potential*). Hence a key difference with other multi-armed bandit studies for IM [Wu et al. (2019); Wen et al. (2017); Chen et al. (2016); Vaswani et al. (2017b)] is that they look for a constant optimal seed set, while in our setting a round's best action (choice of seeds) depends on the number of previous rounds and the context.

Table 3.1: Summary of notations.

T	total number of rounds in a campaign
K	total number of available influencers
Y_t	the d-dimensional context in round t
I_t	the set of L influencers selected in round t
A_k	set of basic nodes reachable by influencer k
$S(I_t, Y_t)$	the spread given by the environment in round t
$p_{k,j}(t)$	the probability of influencer k to activate basic node j in round t
$\theta_{k,j}$	feature vector that explains the probability of influencer k to activate basic node j in round's context Y_t
$n_k(t)$	the history of number of selections of influencer k in round t
$p(j)$	the basic node's j intrinsic probability of activating itself
α	the external factor function which adjusts the basic node's activation probability; e.g. defined as in Equation 3.12.
F_t	the set of IDs of the activated basic nodes at the end of round t
r_t	the reward at the end of round t
$r'_k(t)$	the reward for the external factor's linear regression problem
$C_j(t)$	the cumulative Poisson count of activations for node j in round t
θ_k	the influencer k's feature vector
$\hat{\theta}_k(t)$	the estimator of the influencer k's feature vector in round t
λ_j	the Poisson intensity of activations for basic node j
λ_k	the Poisson intensity of activations due to influencer k
$R_k(t)$	the influencer k's remaining potential (i.e. the feasible reward) in round t
$G_k(t)$	Good-Turing estimator of the remaining potential for influencer k
$V_k(t)$	design matrix updated by the context vectors in rounds when influencer k is played
$s_k(t)$	the rewards history factor for linear regression
γ	the regularization factor for linear regression
$b_k(t)$	the UCB computed for influencer k in round t

3.2 Problem Statement

We formalize the IM problem, set in a discrete-time campaign consisting of T rounds, with K influencers among which the algorithm chooses seeds at each round.

We model each influencer k as having access to A_k basic nodes, each one being influenced by k with a probability $p_{k,j}(t), \forall j \in [1, \ldots, A_k]$. We assume that $p_{k,j}(t)$ depends on each basic node's inner probability $p(j)$ of activating itself, on some d-dimensional profile $\theta_{k,j}$, and on the round's context. In each round, a d-dimensional context $Y_t \in [0, 1]^d$ is provided by the environment, similar to the contextual multi-armed bandit setting [Slivkins (2019)]. Considering that in

our setting the reward is the number of *newly* activated nodes, we assume also the impact of the number of selections of the influencer up to round t, $n_k(t)$, on the probability $p_{k,j}(t)$. Therefore, the probability of a basic node j to be influenced by influencer k is well-approximated by a function $\alpha(\langle\theta_{k,j}, Y_t\rangle, n_k(t))$ applied as a modifier to the basic node's inner activation probability $p(j)$. The modifier α is a function of the relation between the influencer, the basic node, and the round's context. Formally, the problem we study in this chapter is defined as follows:

Problem 3 (Contextual Influence Maximization with Persistence)**.** *Given a set of influencers $\mathcal{K} = \{1, \ldots, K\}, |\mathcal{K}| = K$, a budget of T rounds (or trials), and a number $1 \leq L \leq K$ of influencers to be activated at each round, the objective is to solve the following optimization problem:*

$$\underset{I_t \subseteq \mathcal{K}, |I_t| = L, \forall 1 \leq t \leq T}{\operatorname{argmax}} \mathbb{E}|\bigcup_{1 \leq t \leq T} S(I_t, Y_t)|, \tag{3.1}$$

where $S(I_t, Y_t)$ is the spread of the chosen set of influencers for round t, and the probability that influencer k activates basic node j depends on the round's context Y_t and the number of k's selections $n_k(t)$:

$$p_{k,j}(t) = \alpha(\langle\theta_{k,j}, Y_t\rangle, n_k(t))p(j). \tag{3.2}$$

A similar variant of this problem, which does not use contexts, was proven to be NP-hard in Lagrée et al. (2018), and this hardness result immediately transfers to our problem (e.g., with a constant context for all rounds).

We now formulate the problem in a contextual bandit setting. We assume a semi-bandit feedback at the end of each round, denoted F_t, consisting of the set of IDs of the activated basic nodes. The *reward* for the round is the number of new activations:

$$r_t = \sum_{j=1}^{\bigcup_{k \in I_t} A_k} \mathbb{I}\{C_j(t) > 0\} - r_{t-1}; r_0 = 0, \tag{3.3}$$

where $C_j(t) = \sum_{s=1}^{t} \mathbb{I}\{j \in F_s\}$ denotes for each basic node the number of times it has been activated. When $L > 1$ influencers are selected per round, F_t can be separated per influencer if such information is available in the feedback. E.g., it is common in influencer marketing for

basic nodes to use a code specific to the influencer that reached them when they activate themselves; this still qualifies as semi-bandit feedback as it does not require full information about the influence propagation. In the absence of such additional information, the feedback F_t can be randomly divided equally among the selected influencers; the agent only uses the quantitative aspect of the reward in the learning process.

Given that the reward in each round is the number of *newly* activated basic nodes, Problem 3 exhibits a *diminishing returns property*: for each influencer, the expected number of new basic nodes it can activate decreases with each of its selections.

For each basic node j, its cumulative count of activations $C_j(t)$ up to round t is a random quantity depending on the node's probability $p_{k,j}(t)$ of being activated by the played influencer; these activation probabilities are assumed to be unknown. As estimating all user profiles $\theta_{k,j}$ is computationally expensive, our goal will be instead – given that the objective is to select the best influencer(s) at each round – to directly estimate the influencers' potential based on the context at each round, as a proxy for the probabilities of individual nodes.

To achieve this, we propose two algorithms that both assume a generalization θ_k of the unknown parameters $\theta_{k,j}$, and two different assumptions on the distribution of new activations for each influencer. More precisely, we assume that activations follow either (i) a Poisson distribution, given that they are counts of nodes, or (ii) a Log-Normal distribution, assuming that the scales of the rewards are normally distributed (in line with observations on the distribution of real-world social phenomena [Sala et al. (2010)]). In Section 3.3 we present the UCB-based solution that uses the Poisson distribution assumption, and in Section 3.4 we present the LinUCB-based solution that assumes a log-normal distribution.

3.3 GLM-GT-UCB Algorithm

The main idea behind the `GLM-GT-UCB` algorithm is to estimate the potential of each influencer, at each round, by some proxy measure. Here, by an influencer's *potential* we understand the number of nodes that it can *still* activate (i.e., the reward); more formally, each influencer's remaining

potential of activating new basic nodes in round t is:

$$R_k(t) = A_k - \sum_{j=1}^{A_k} \mathbb{I}\{C_j(t-1) > 0\} \tag{3.4}$$

The stochasticity of $C_j(t-1)$ means that the remaining potential is a random variable too. While this has been analyzed in the non-contextual case [Lagrée et al. (2018)], the challenge here is to account for the contextual dimension. The proxy we choose is the Good-Turing estimator [Good (1953)], estimating the proportion of unseen items in a random process as the fraction of items seen only once (*hapaxes*).

There are two main technical challenges to modeling the remaining potential using Good-Turing estimators: (i) we are counting only *new* activations, so a *fatigue* factor needs to be added to the estimator, and (2) the contextual case forces us to make an assumption on the model – in our case, we opted for a generalized linear model using a Poisson distribution.

3.3.1 Good-Turing with Poisson and External Factor

An influencer's remaining potential is an unknown random variable. The Good-Turing estimator [Good (1953)], adjusted with a fatigue function, was shown to successfully model an influencer's fatigue [Lagrée et al. (2018)]. The fatigue function, non-increasing in the number of influencer's selections, does not explicitly model an influencer's potential w.r.t. the diffused content. We thus propose a Good-Turing estimator adjusted by a function of the diffused content.

For each basic node j, its activation probability $p_{k,j}(t)$ is a function of (a) the linear combination of the node's feature vector $\theta_{k,j}$ and the round's context, and (b) the number of influencer's selections $n_k(t)$. The assumption we make is that the underlying distribution of each node's cumulative count of activations $C_j(t)$ is Poisson with intensities $\lambda_j \sum_{s=1}^{t} \sum_{k \in I_s} \alpha(\langle \theta_{k,j}, Y_s \rangle,\, n_k(s))$. Our approach is then to assume that the underlying distribution for the entire remaining potential of an influencer is Poisson with intensities $\alpha(\langle \theta_k, Y_t \rangle, n_k(t))\lambda_k, k \in \mathcal{K}$, where the individual user response probabilities are small: $\lambda_k \geq \sum_{j=1}^{A_k} \lambda_j \ll A_k$. Recall the true feature vector θ_k is initially unknown, so its estimation becomes a sub-problem of our problem. The classical solution is to

use the regularized least-squares estimator:

$$\hat{\theta}_k(t) = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{s=1}^{t-1} (r_k^{'}(s) - \langle \theta, Y_t \rangle)^2 + \gamma \|\theta\|_2^2), \tag{3.5}$$

where $r_k^{'}(s)$ is the round's reward (adapted for the sub-problem) and γ is the penalty factor that ensures the solution's uniqueness; more details are given in Sec. 3.3.2.

After t rounds, we observe the cumulative Poisson counts $C_j(t)$ of activations of each basic node $j \in \{1, \ldots, A_k\}$ by the corresponding influencer. The cumulative counts are distributed at the rate

$$\lambda_j \sum_{s=1}^{t} \sum_{k \in I_s} \alpha(\langle \theta_{k,j}, Y_s \rangle, n_k(s)), \tag{3.6}$$

and in estimation with rate

$$\lambda_j \sum_{s=1}^{t} \sum_{k \in I_s} \alpha(\langle \theta_k, Y_s \rangle, n_k(s)). \tag{3.7}$$

Thus, the remaining potential can be expressed as the conditional expectation of cumulative counts of new basic nodes that would be influenced in round t:

$$R_k(t) = \sum_{j=1}^{A_k} \lambda_j \alpha(\langle \theta_k, Y_t \rangle, n_k(t)) \mathbb{I}\{C_j(t-1) = 0\}. \tag{3.8}$$

The discrete random variable $C_j(t-1)$ has a Poisson distribution with parameter $\lambda_j \sum_{s=1}^{t-1} \sum_{k' \in I_s} \alpha(\langle \theta_{k'}, Y_s \rangle, n_{k'}(s))$ and the probability that node j is not activated:

$$P(C_j(t-1) = 0) = e^{-\lambda_j \sum_{s=1}^{t-1} \sum_{k' \in I_s} \alpha(\langle \theta_{k'}, Y_s \rangle, n_{k'}(s))}. \tag{3.9}$$

The expectation of k's remaining potential in round t is:

$$\mathbb{E}[R_k(t)] = \alpha(\langle \theta_k, Y_t \rangle, n_k(t)) \sum_{j=1}^{A_k} \lambda_j e^{-\lambda_j \sum_{s=1}^{t-1} \sum_{k' \in I_s} \alpha(\langle \theta_{k'}, Y_s \rangle, n_{k'}(s))}. \tag{3.10}$$

`GLM-GT-UCB` estimates k's remaining potential by:

$$G_k(t) = \frac{\alpha(\langle\hat{\theta}_k(t), Y_t\rangle, n_k(t))}{n_k(t)} \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{\mathbb{I}\{X_{s,j,k} = 1, \{X_{s,j,k'} = 0\}_{k' \in I_s \setminus \{k\}}, \{X_{l,j,k'} = 0\}_{l \neq s, k' \in I_l}\}}{\alpha(\langle\hat{\theta}_k(s), Y_s\rangle, n_k(s))}, \tag{3.11}$$

where $n_k(t)$ is the number of selections of influencer k up to round t, $X_{s,j,k}$ is a binary random variable equal to 1 when j is activated in round s by influencer k, $l \in \{1, 2, \ldots, t-1\}, k^{'} \in I_l \subseteq \mathcal{K}$. We discuss next the external factor estimated through regular linear regression, used to regulate the proportion of hapaxes in the cascades generated by influencer k.

3.3.2 The External Factor

The external factor, as stated before, is a sub-problem of Problem 3. The remaining potential of an influencer is modeled by the combination of the external factor and the average count of hapaxes from the Good-Turing estimator. Under the assumption of a Poisson distribution for the rewards, and their property of diminishing returns, the external factor can be chosen as an adaptation of the inverse link function (mean function) for the Poisson distribution:

$$\alpha(\langle\theta_k, Y_t\rangle, n_k(t)) = e^{f(n_k(t))\langle\theta_k, Y_t\rangle} \tag{3.12}$$

The $f(n_k(t))$ function is assumed to be non-increasing, models the influencer's fatigue, and depends on the influencer's selections. By combining the two estimators, the predicted values for this sub-problem are:

$$r'_k(t) = \frac{\ln\left(\frac{r_t n_k(t)}{\sum_{j=1}^{A_k}\sum_{s=1}^{t-1}\frac{\text{hapax}_{s,j,k}}{\alpha(\langle\hat{\theta}_k(s), Y_s\rangle, n_k(s))}}\right)}{f(n_k(t))}, \text{where} \tag{3.13}$$

$$\text{hapax}_{s,j,k} = \mathbb{I}\{X_{s,j,k} = 1, \{X_{s,j,k'} = 0\}_{k' \in I_s \setminus k}, \{X_{l,j,k'} = 0\}_{l \neq s, k' \in I_l}\} \tag{3.14}$$

The argument of the external factor function is a random variable $r'_k(t) = \langle\theta_k, Y_t\rangle + \eta_t$. The

noise η_t is assumed conditionally 1-subgaussian. The regularized least-squares estimator for the feature vector is:

$$\hat{\theta}_k(t) = V_k^{-1}(t) \sum_{s=1}^{t-1} Y_s r_k^{'}(s) \mathbb{I}\{k \in I_s\}, \tag{3.15}$$

where $V_k(t) = \gamma I_d + \sum_{s=1}^{t-1} Y_s Y_s^T \mathbb{I}\{k \in I_s\}$; $\gamma \geq 0$ is the penalty factor that ensures an unique solution. The design matrix $V_k(t)$ is computed from the contexts of the rounds in which the corresponding influencer was played, adjusted by its number of selections.

3.3.3 Upper-Confidence Bound

UCB algorithms provide a disciplined balance between the exploitation of the options that are known as best up to the decision round and the exploration of the ones for which the learning agent has not acquired enough information yet. The `GLM-GT-UCB` algorithm follows the main lines of an UCB-based algorithm, and its flow is presented in Algorithm 2. It starts with an initialization phase, where each influencer is played once in a random context. The observed rewards are used to initialize the influencer's statistics, necessary for further decisions. For the Good-Turing estimator, we maintain the number of selections $n_k(t)$ and the history of the discounted rewards for computing this estimator, as well as the sample-mean new activations for computing the UCB index. For the linear regression of the external factor, we maintain a history of the rewards and the design matrix for each influencer:

$$V_k(t) = \gamma I_d + \sum_{s=1}^{t-1} Y_s Y_s^T \mathbb{I}\{k \in I_s\} \tag{3.16}$$

$$s_k(t) = \sum_{s=1}^{t-1} Y_s r_k'(s) \mathbb{I}\{k \in I_s\}. \tag{3.17}$$

In each subsequent round, the agent gets the context from the environment. It estimates for each influencer its feature vector, by the regularized least-square estimator in the stochastic linear bandit $\hat{\theta}_k(t)$, which is then used to compute the estimator of the remaining potential. The UCB

$b_k(t)$ is obtained by adding the confidence factor $\beta_k(t)$. The agent plays the influencers with the highest UCBs, observes and divides the reward equally among them, and updates their statistics.

The UCB index computed on the adapted Good-Turing estimator captures both the confidence in the unmodified Good-Turing estimator and the one in the estimator of the influencer's true unknown vector θ_k:

$$b_k(t) = G_k(t) + \beta_k(t) + \hat{\lambda}_k(t) \left(1 - e^{\frac{-2+C_k(t)}{n_k(t)}} \frac{1}{n_k(t)} \sum_{s=1}^{t-1} e^{\frac{-2-C_k(s)}{n_k(s)}}\right), \text{ where} \tag{3.18}$$

$$\beta_k(t) = \sqrt{\frac{2\hat{\lambda}_k(t) e^{\frac{3+2C_k(t)}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{2-2C_k(s)}{n_k(s)}}}{n_k^2(t)} \ln\frac{1}{\delta}} + \sqrt{\frac{e^{2/n_k(t)} \hat{\lambda}_k(t) \ln(1/\delta)}{\sum_{s=1}^{t-1} \sum_{k' \in I_s} e^{-1/n_{k'}(s)}}} + \frac{e^{\frac{1+C_k(t)}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{1+C_k(s)}{n_k(s)}}}{3n_k(t)} \ln\frac{1}{\delta}, \tag{3.19}$$

$C_k(t) = \gamma \|Y_t\|_{V_k^{-1}(t)}$ is the contextual UCB for the external factor and the sample-mean number of influencer k's new activations is:

$$\hat{\lambda}_k(t) = \frac{\alpha(\langle \theta_k(t), Y_t \rangle n_k(t))}{n_k(t)} \sum_{s=1}^{t-1} \frac{|S_s| \mathbb{I}\{k \in I_s\}}{\alpha(\langle \theta_k(s), Y_s \rangle n_k(s)) L}. \tag{3.20}$$

3.3.4 Theoretical Analysis

The UCB index is chosen as the maximum difference that can occur between the GT estimator and the true remaining potential with some chosen confidence. Theorem 3.3.1 provides the confidence interval for the estimated remaining potential. Its proof has three steps: the concentration of the true remaining potential, the concentration of the Good-Turing estimator, and the bias of the estimator.

Theorem 3.3.1. *With probability at least* $1 - \delta$*, having the expected activations* $\lambda_k = \sum_{j=1}^{A_k} p_{k,j}(t)$*, and* $\beta_k(t)$ *set as in Equation 3.19 , we have*

[1] $r'_k(1) = \frac{1}{f(1)} \ln\left(\frac{r_t}{\sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{\mathsf{hapax}_{s,j,k}}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, 1)}}\right)$.

Algorithm 2 GLM-GT-UCB

Input: influencers $\mathcal{K}$, rounds budget T, external factor function α, regularization factor γ, fatigue function f, number of selections L

Initialization: play each influencer $k \in \mathcal{K}$ once in given random contexts Y_t, observe the reward $r_t, t \in \mathcal{K}$, and update the statistics $n_k(1) = 1$, and $V_k(1) = \gamma I_d + Y_t Y_t^T$, $s_k(1) = Y_t r'_k(1)$ [1] for the external factor.

for $t = K+1, \ldots, T$ **do**

  Get the context Y_t

  for $k \in \mathcal{K}$ **do**

    Estimate the unknown vector:

$$\hat{\theta}_k(t) = V_k^{-1}(t) s_k(t) \tag{3.21}$$

    Compute UCB for remaining potential estimator

$$b_k(t) = G_k(t) + \beta_k(t), \tag{3.22}$$

$$G_k(t) = \alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t)) \frac{1}{n_k(t)} \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{\text{hapax}_{s,j,k}}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))} \tag{3.23}$$

    and $\beta_k(t)$ is given by the confidence interval, and $\text{hapax}_{s,j,k}$ by Equation 3.14.

  end for

  Choose set I_t of L influencers with largest UCB.

  Play the chosen influencers, observe spread, divide it equally among influencers, and update their statistics:

  for $k' \in I_t$ **do**

    Update $r_{k'}(t)$ by Eq. (3.13).; $n_{k'}(t+1) = n_{k'}(t) + 1$; $V_{k'}(t+1) = V_{k'}(t) + Y_t Y_t^T$; $s_{k'}(t+1) = s_{k'}(t) + Y_t r'_{k'}(t)$

  end for

end for

$$-\beta_k(t) + \Omega\left(\frac{T\lambda_k(T)}{n_k(T)} e^{\frac{C_k(T)}{n_k(T)}}\right) \leq R_k(t) - G_k(t) \leq \beta_k(t) + \mathcal{O}\left(T\frac{\lambda_k(T)}{n_k(T)} e^{\frac{C_k(T)}{n_k(T)}}\right) \tag{3.24}$$

Proof. See the complete proof in Appendix A. □

3.4 Log-normal Distribution

We now consider the second alternative, that the underlying distribution is a log-normal one. A log-normal distribution of rewards is equivalent to a normal distribution of the reward scale.

Now, the influence maximization problem can be solved by an adapted LinUCB [Chu et al. (2011)]. LinUCB computes the expected reward of each arm by finding a linear combination of the previous rewards of the arm. It estimates the unknown parameter θ_t of the current round as a linear combination of the previously seen feature vectors and rewards, and it estimates the expected reward on the current round by linearly combining it with the current feature vector. The adaptation of LinUCB to our problem consists in maintaining a design matrix per influencer, which is updated by the context of the round in which the influencer has been played. This change implies that a separate parameter is estimated for each influencer, and its linear combination with the current round's context will estimate the reward at a logarithmic scale. Note that the linear combination estimates the scale of the reward since we assume that the rewards are log-normally distributed. The main flow is presented in Algorithm 3. It is similar to the one of LinUCB [Chu et al. (2011)], in that at each step it chooses the best reward in terms of the linear combination of the context and the learned profile plus confidence bound. In general linear models – of which LogNorm-LinUCB is part of – this bound is based on a design matrix V_k and the given context Y_t.

3.4.1 Regret analysis

The regret analysis is performed at a logarithmic scale; this restriction stems from having the logarithm of the new activations being normally distributed. In Chu et al. (2011), theoretical guarantees for LinUCB were challenging, due to the lack of independence of the random variables for the rounds' rewards. The solution was to use a supporting algorithm, SupLinUCB, estimating the unknown parameter only from the feature vectors and rewards from the rounds in which the agent performs random exploration. Each round is split into levels, and each level maintains an index set used for learning, comprising the indices of the rounds with independent rewards. When exploring, the round is added to the index set of the corresponding level.

We designed similarly IM-SupLinUCB and its sub-routine IM-BaseLinUCB, preserving the steps of SupLinUCB [Chu et al. (2011)] and SupLinRel [Auer (2002)]. Each influencer's UCB is computed for the scale of the reward – new activations. We skip the analysis of IM-BaseLinUCB and IM-SupLinUCB's, as it is similar to Chu et al. (2011); Auer (2002). Regret for stochastic linear bandits

Algorithm 3 `LogNorm-LinUCB`

Input: influencers K, selections L, $\gamma \in \mathbb{R}_+$, $d \in \mathbb{N}$
$V_k(1) = I_d, \forall k \in \mathcal{K}$ and $s_k(1) = \mathbf{0}_d, \forall k \in \mathcal{K}$
for t=1,...,T **do**
  Get context Y_t.
  for $k \in \mathcal{K}$ **do**
    $\hat{\theta}_k(t) = V_k^{-1}(t) s_k(t)$
    $b_k(t) = \langle \hat{\theta}_k(t), Y_t \rangle + \gamma \sqrt{Y_t^T V_k^{-1}(t) Y_t}$
  end for
  Choose set I_t of L influencers with largest UCB $b_k(t)$.
  Observe spread, compute reward r by discounting previously activated basic nodes and dividing by L.
  for $k^{'} \in I_t$ **do**
    $V_{k'(t)}(t+1) = V_{k'(t)}(t) + Y_t Y_t^T$
    $s_{k'(t)}(t+1) = s_{k'(t)}(t) + \ln(r) Y_t$
  end for
end for

is generally defined as:

$$\hat{\mathcal{R}}_t = \sum_{s=1}^{t} \max_{k \in [K]} \langle \theta_k, Y_t \rangle - \sum_{s=1}^{t} r_s \tag{3.25}$$

$$\mathcal{R}_t = \mathbb{E}[\hat{\mathcal{R}}_t] = \mathbb{E}\left[\sum_{s=1}^{t} \max_{k \in [K]} \langle \theta_k, Y_t \rangle - \sum_{s=1}^{t} r_s \right] \tag{3.26}$$

We have the following $\tilde{\mathcal{O}}(\sqrt{T})$ regret bound for the supporting algorithm on logarithms of rewards:

Theorem 3.4.1. *If `IM-SupLinUCB` uses parameter $\gamma = \sqrt{\frac{1}{2}\ln(\frac{2TK}{\delta})}$, with probability $1-\delta$ the regret of `LogNorm-LinUCB` at logarithmic scale is*

$$\hat{\mathcal{R}}_t \leq 2\sqrt{T} + 44K \left(1 + \ln\left(2TK\ln(T)/\delta\right)/2\right)^{\frac{3}{2}} \sqrt{Td} \tag{3.27}$$

Proof. Proof similar to that of (Auer, 2002, Theorem 6.). □

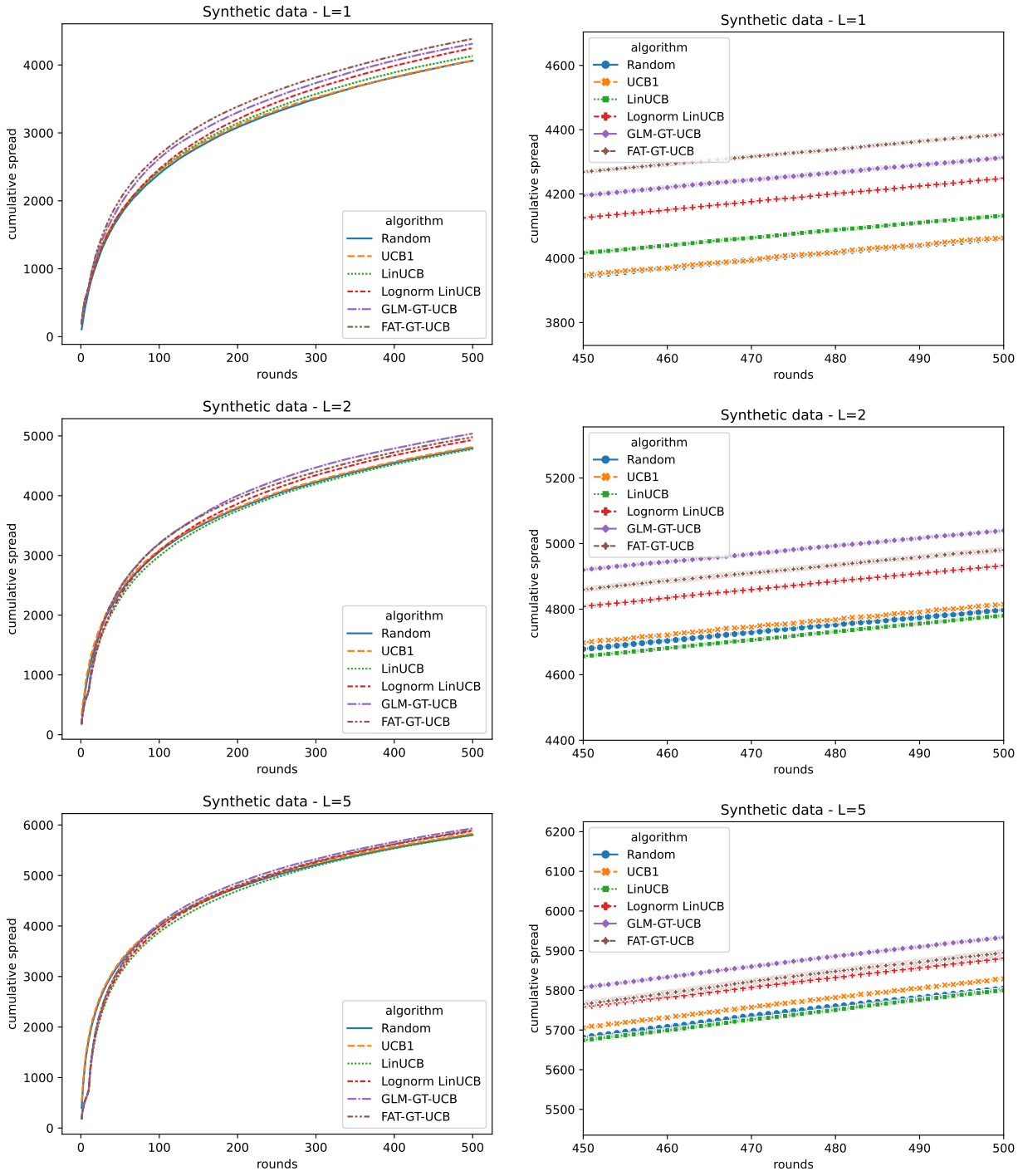

Figure 3.1: Synthetic - Cumulative rewards, full plot (left column), plot zoomed to last 100 rounds (right column).

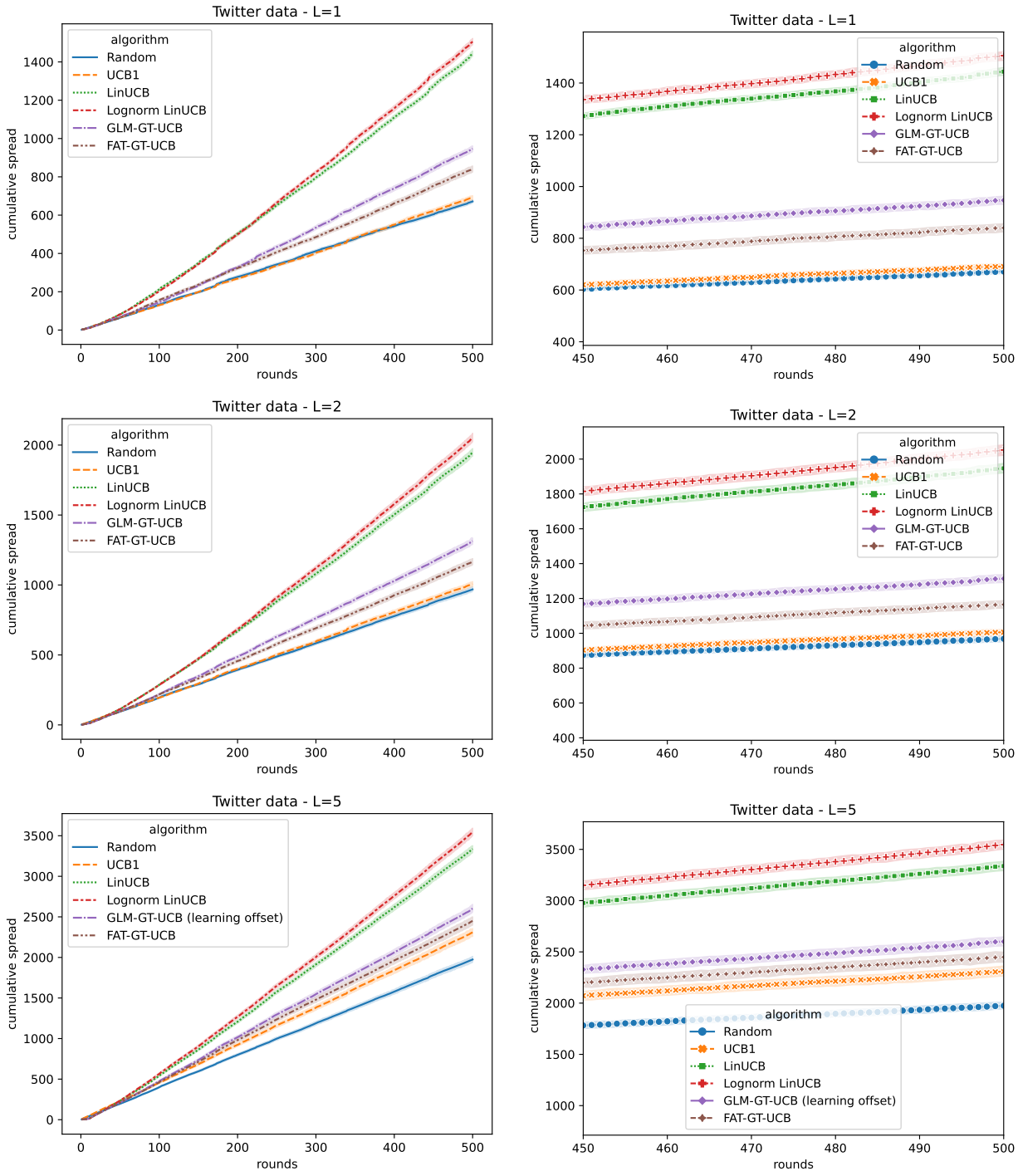

Figure 3.2: Twitter - Cumulative rewards, full plot (left column), plot zoomed to last 50 rounds (right column).

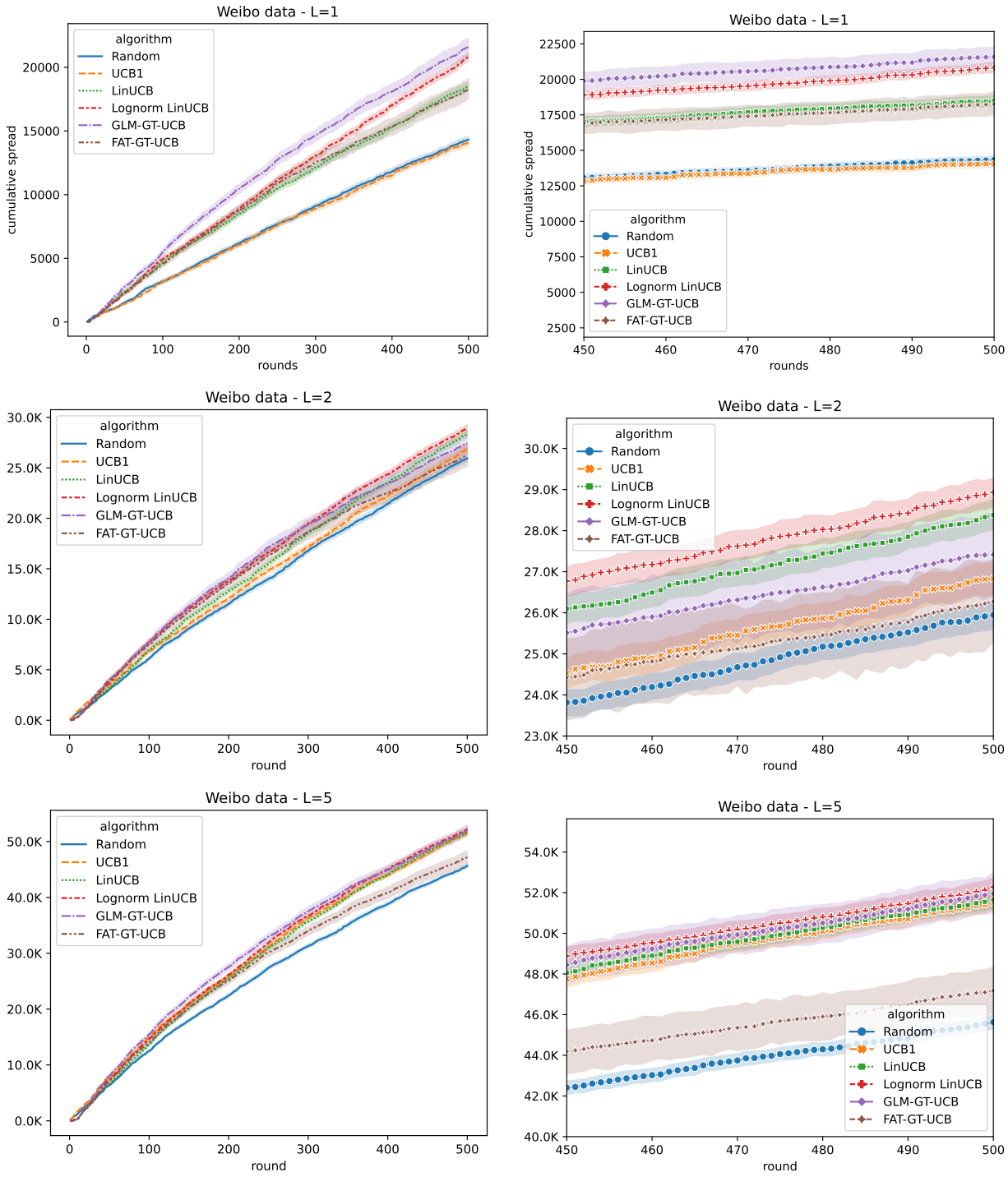

Figure 3.3: Sina Weibo - Cumulative rewards, full plot (left column), plot zoomed to last 50 rounds (right column).

3.5 Experiments

We tested `GLM-GT-UCB` and `LogNorm-LinUCB` on synthetically generated data, on data we collected from Twitter, and on a publicly available dataset from Sina Weibo [Zhang et al. (2013)]. All the results are averaged over 100 runs.

Synthetic data experiments. The synthetic data is generated starting from the premise that each basic node's activation probability is known. Therefore, all the edges and nodes are assumed to be known as well. The synthetic graph is randomly generated following the Barabási-Albert preferential attachment model [Barabási et al. (2016)]. The model's parameters are chosen as follows: $30{,}000$ nodes and, at each step, one new edge to be attached from new nodes to existing ones. Then, the 10 nodes having the maximum degrees are chosen to be the influencers.

Activation probabilities are computed as a sigmoid function of the inner product of the context, sampled from a normal distribution $\mathcal{N}(1, 1)$, and the basic node's feature vector, sampled from $\mathcal{N}(1, 3)$, plus some random small noise. This is preferable in order to project the results into probability thresholds, i.e., the value over which the node is considered activated - 0.999 in our experiments. The inner product captures the linear relationship between context and hidden profile. For each node, its feature vector is randomly generated from a normal distribution. Then, the context of a campaign's round is generated from another normal distribution. A given round is chosen to be viral with a 50% probability, i.e., the distribution from which the context is drawn is chosen such that its inner product with most of the basic user feature vectors results in higher values for the activation function. For these rounds, only $L + 1$ influencers are chosen to use the viral context. The diffusion model is assumed to be Independent Cascade [Kempe et al. (2003)], a campaign consisting of 500 rounds, and the results are averaged over 100 runs. γ is set to $\sqrt{1/2 \ln(\sqrt{2TK/\delta})}$ everywhere, the external factor α is defined in Equation 3.12, and λ_k is estimated by the sample-mean defined in Equation 3.20.

Baselines methods. We compare against Random, `UCB1` [Auer et al. (2002a)], `LinUCB` [Chu et al. (2011)], and `FAT-GT-UCB` [Lagrée et al. (2018)]. The random policy chooses a random influencer in each round. `UCB1` is a well-known algorithm in the bandit literature, one which does not model

contexts. The FAT-GT-UCB algorithm models the influencer's fatigue in a context-free setting. The results (Fig. 3.1) show that GLM-GT-UCB and LogNorm-LinUCB are both capable of learning the remaining potential of influencers from their performance in different contexts. Making decisions based on the available information about the round's context has a clear added value, compared to only considering the time-based fatigue of approaches such as FAT-GT-UCB.

Twitter dataset. We extracted from Twitter logs a collection of retweets. These can be viewed as belonging to basic nodes, representing successful activations of the original tweets from influencers. To test the capability of the algorithms to choose the right influencers for a given context, we extracted from the tweet the round's context. As in Shin et al. (2015), a tweet is encoded into a multi-dimensional vector. The encoding represents the distribution of the tweets' words over a predefined number of centroids (24 in our experiments). The centroids are obtained via clustering (K-means) on the public vocabulary *glove-twitter-200*[2] from the word embedding open-source library Gensim[3]. Each word is assigned to its closest centroid, thus obtaining the distribution. The largest cluster is split into 5 smaller clusters.

In Twitter and Weibo, we improve the learning rate of GLM-GT-UCB by adding $10/L$ activations only when learning the external factor via linear regression. The plotted results are with the true values of activations.

The campaigns are created by randomly choosing the context for each round to be one of the available centroid distributions in the dataset. We chose the set of influencers to be the users with the highest degrees. In each round, each algorithm chooses which influencers it wants to play. Due to the sparsity in the data, we implemented the bandit to sample with replacement from the set of all tweets with the round's context matching their centroid distribution and the algorithm's chosen influencer as the original user id. If there is no log for this tuple, we consider that no basic node has been activated. The reward is computed by discounting previously activated basic nodes.

The results are in Fig. 3.2, for either the entire campaign of 500 rounds or zoomed on rounds 450 to 500; the shaded areas represent the uncertainty.

[2] https://nlp.stanford.edu/projects/glove/
[3] https://github.com/RaRe-Technologies/gensim-data

Weibo dataset. Using a public dataset from the popular Chinese microblogging platform, we designed the experiment as in the Twitter scenario. The topic distributions created by Zhang et al. (2013) are used as contexts. There are 100 topics, and for each post, the distribution of topics is computed by using Latent Dirichlet Allocation [Heinrich (2005)]. Once again, in Fig. 3.3 we can see that our methods manage to perform better by using the round's context information when selecting influencers. The relative performance can depend on time: `GLM-GT-UCB` seems to initially learn faster.

From both experiments on real-world datasets, we can conclude that our approaches – especially `LogNorm-LinUCB` – are capable of learning viral cascades in different datasets and cascade settings, which increases their potential in spread maximization (visible in the "steps" of the plots); this is not the case with other approaches, which seem to work best when the cascades have fewer outliers in terms of size; hence, they do not learn quickly enough to adapt.

3.6 Conclusion

We presented in this chapter the problem of designing advertising campaigns from the point of view of contextual influence maximization when the exact diffusion model is not fully exploitable. By adapting approaches from the contextual bandit literature, we designed algorithms `GLM-GT-UCB` and `LogNorm-LinUCB`, using different assumptions on the underlying distributions of the number of influenced nodes: Poisson and log-normal respectively. We showed both theoretically and experimentally that our approaches have the potential to learn the influencers' potential, leading to improved IM campaigns compared to other state-of-the-art methods.

Chapter 4

Episodic Contextual Influence Maximization with Persistence

In this chapter, we extend the CIMP problem to the *Episodic Contextual Influence Maximization with Persistence(ECIMP)* problem to also consider the state of the environment when making decisions and with the benefit of also learning from previous campaigns. As in Chapter 3, we begin by presenting the state of the art related to our problem in Section 4.1, which is then formally defined in Section 4.2. After extending the CIMP problem to the multi-campaign scenario from ECIMP, we formalize the latter in terms of MDPs for episodic RL. For this problem, we propose in Section 4.3 `LSVI-GT-UCB` which learns from previous campaigns and for the context of the round with a Least-Squares Value Iteration estimator, which we modify with the Good-Turing estimator with fatigue for a more efficient representation of the *diminishing returns* property of the reward. In Section 4.4 we compare the performance of `LSVI-GT-UCB` with the two algorithms it is based on, `LSVI-UCB` and `Fat-GT-UCB`, on synthetic and real datasets, and demonstrate that the sum of cumulative rewards over all campaigns is significantly improved by the optimistic combination of the two estimators. Finally, in Section 4.5 we propose a research direction for solving ECIMP with deep RL. We detail the adaptation of an existing framework for learning combinatorial optimization algorithms over graphs [Khalil et al. (2017)] to our problem.

4.1 Main Related Work

Li et al. (2020) study the OIM problem in social networks under the assumptions of the LT model and node-level feedback. The proposed algorithmic solution is called `LT-LinUCB`. It exploits the linearity of node activations in the LT model, obtaining an $O(\text{poly}(m)\sqrt{T}\log T)$ regret, where m is the number of edges and T is the number of rounds. They propose also the model-free `OIM-ETC` algorithm, with an $O(\text{poly}(m)T^{\frac{2}{3}})$ regret bound. Zhang et al. (2022) propose an $\tilde{O}(\sqrt{T})$ algorithm, called `IC-UCB`, for the OIM problem, assuming the IC model and node-level feedback. `IC-UCB` uses a standard offline IM oracle to find the best seed set, and estimates the IC model's edge parameters p by transforming them into another parameter; this leads to an instance of the generalized linear model problem [Filippi et al. (2010b)], which is solved with MLE.

Khalil et al. (2017) designed a generic framework for learning graph heuristics and finding approximate solutions for various NP-hard combinatorial optimization problems. The framework is a greedy meta-algorithm, `Q-learning for the Greedy Algorithm`, learned over multiple episodes of RL over different problem instances sampled from a given graph distribution. For each problem type, one must provide specific helper and cost functions, as well as the termination criteria which enable the meta-learning algorithm to solve the problem. E.g., for the Set-Cover problem, the helper function would be the identity one, as there is no need for a combinatorial structure on the partial solution, the cost function would be the size of the partial solution, and the termination criteria would be either when all nodes are covered or the budget is spent. At each step within the episode's horizon, `Q-learning for the Greedy Algorithm` chooses a seed node either randomly with probability ϵ, or the one which maximizes the estimated Q-function $\hat{Q}(h(S_t), v; \Theta), \forall v \in V, G = (V, E)$. For each node $v \in V$ in the graph, the algorithm uses *structure2vec* (S2V) [Dai et al. (2016)] to encode its neighborhood, given the current partial solution S_t. This framework is quite versatile and applicable to a wide range of combinatorial optimization problems on graphs. However, it requires knowledge about the graph's topology, as the embedding of the potential new seed node and the pooled embedding over the entire graph are combined to provide the estimated Q-function. The approximator's parameters are updated in

batches, which allows delayed rewards. The algorithm is experimentally proven to be successful for various types of NP-hard problems, graph types, and graph sizes. Li et al. (2019) have later adapted this approach, in the DISCO framework for influence maximization.

Jin et al. (2020) address the challenge of RL with a very large number of states, by incorporating function approximation in the learning process. The proposed algorithm, called *LSVI-UCB*, models the problem as an episodic Markov Decision Process (MDP), with the assumption that the transition dynamics and the reward function are linear. It is proven that the action-value function is consequently linear as well, and the algorithm is designed to approximate well this quantity. Inspired by the linear bandits literature, the algorithm implements the *"optimism in the face of uncertainty"* principle – it encourages exploration by adding a UCB bonus. It achieves $\tilde{O}(\sqrt{d^3H^3T})$ regret, where d is the ambient dimension of the feature space, H denotes the horizon (i.e., length of each episode), and T is the total number of steps. LSVI-UCB runs in polynomial time ($O(d^2AKT)$), where A is the size of the action space and K is the number of episodes. The algorithm benefits from the sample complexity guarantees, in the sense that with a constant probability, it can learn an $\epsilon-$optimal policy π which satisfies $V^*(x_1) - V^\pi(x_1) \leq \epsilon$, using $\tilde{O}(d^3H^4/\epsilon^2)$ samples, when the initial state x_1 is fixed for all episodes. The algorithm is shown to be robust to small model variations, under the condition of using a different hyper-parameter β from the UCB in different episodes. The main drawback of Jin et al. (2020) is that their solution remains limited to "almost" linear MDPs.

Under weaker assumptions than Jin et al. (2020), the work of Wang et al. (2019) proposes an efficient least-squares dynamic programming algorithm for episodic RL, also called *LSVI-UCB*, which approximates the Q-function with a Generalized Linear Model. The approximator overestimates the optimal Q-function, implementing the "optimism in face of uncertainty" principle. The statistical efficiency is theoretically proven with a regret bound of $\tilde{O}(\sqrt{d^3T})$, where d is the feature dimension, and T is the number of episodes. For these results, an optimistic closure concerning the Bellman operator assumption is made. The assumption is proven to be strictly weaker than the one of linearity, by providing an MDP which meets the former but not the latter. Optimistic closure also implies realizability, which is typical for the contextual multi-armed bandits setting,

where the horizon is $H = 1$.

4.2 Problem Formulation

The IM problem addressed in this Ph.D. work is aimed at information diffusion scenarios – e.g., in a social network, but generally in any medium that may exhibit stochastic / epidemic diffusion – where multiple attempts of spreading the information are made, and the new activations make up the reward. The diffusion network can be naturally represented as a graph $G = (V, E)$ where V are the nodes (users, profiles) and E are the edges (relationships). In our setting, this topology is assumed to be unknown.

Instead, a set of K potential influencers (among the nodes in V) is assumed to be known, and the influence process can only start from them, with the effect of activating certain nodes among those from an unknown overall set of *basic* (influenced) nodes. While we make no assumptions on the diffusion model that leads to activations, we assume to get *semi-bandit feedback* after each *round* that spreads a given "message", as the set influenced nodes (a set of node *Ids*).

Over a *campaign*, consisting of a number of H rounds, the *reward* is defined as the number of *new activations*.

The message that is to be spread at each round $h \in [H]$ is encoded as a vector $Y_h \in \mathbb{R}^d$ and the probability that each target (or basic) node j adopts it – or gets influenced/activated by it – depends linearly on j's hidden profile relative to the influencer k seeded at that round, denoted $\theta_{k,j}$, and the message (plus some noise). So the response of a target node j is given by $\langle \theta_{k,j}, Y_h \rangle + \epsilon$.

This response of node j, along with the number of times the influencer k was seeded to send a message in the campaign, denoted n_k, are used in a generalized linear function α, called the *external factor*. The role of the external factor α is to modulate the default (inherent) propensity of node j to activations, denoted as $p(j)$.

The single-campaign problem, CIMP, formulated in Chapter 3, can be used as an intermediary step to introduce the more general ECIMP setting we study in this chapter. The solution to CIMP, Problem 3, relies on upper-confidence bound approaches (UCB), which need to estimate

at each step the *remaining potential* of influencers. Applying the Good-Turing estimator, where the remaining potential can be estimated via the *hapaxes*, coupled with a theoretically derived upper-confidence bound, has been shown to work well in practice when the distribution on the number of newly activated nodes follows a Poisson distribution (Algorithm `GLM-GT-UCB`).

We describe next how the Problem 3 can be extended to the *episodic* case of multiple campaigns, each having multiple rounds, to learn between campaigns.

4.2.1 Reinforcement Learning Setting

In the episodic, i.e., multi-campaign setting, the problem becomes:

Problem 4 (Episodic Contextual Influence Maximization with Persistence (ECIMP))**.** *Given a set of influencers $\mathcal{K} = \{1, \ldots, K\}$, a budget of T campaigns, each consisting of H rounds, and a number $1 \leq L \leq K$ of influencers to be activated at each round, the objective is to solve the following optimization problem:*

$$\underset{I_{h,t} \subseteq \mathcal{K}, |I_{h,t}| = L, \forall 1 \leq h \leq H, \forall 1 \leq t \leq T}{\operatorname{argmax}} \mathbb{E} \Big| \bigcup_{1 \leq h \leq H, 1 \leq t \leq T} S(I_{h,t}, Y_{h,t}) \Big|, \tag{4.1}$$

where $S(I_{h,t}, Y_{h,t})$ is the spread of the chosen set of influencers for round h in campaign t, and the probability that an influencer k activates some basic node j depends on the round's context $Y_{h,t}$ and the number of k's selections $n_k(h,t)$ in campaign t:

$$p_{k,j}(h,t) = \alpha(\langle \theta_{k,j}, Y_{h,t} \rangle, n_k(h,t)) p(j). \tag{4.2}$$

(In what follows, we will use the terms campaign and episode interchangeably. The former is closer to the terminology of the application scenario, the latter is common from episodic RL.)

The problem can be naturally modeled as K episodic Markov decision processes, one for each influencer $k \in \mathcal{K}$, namely MDP$(\mathcal{S}_k, \mathcal{A}_k, H_k, \mathbb{P}_k, r_k)$, where $\mathcal{S}_k$ is the state space, $\mathcal{A}_k$ is the set of possible actions, H_k is the horizon within each episode, $\mathbb{P}_k = \{\mathbb{P}^H_{k,h=1}\}$ is the set of state transition probability measures, and $r_k = \{r^H_{k,h=1}\}$ is the set of reward functions. The state space of such an MDP can be very large, possibly infinite. The action spaces, given that an MDP is maintained for

each influencer, is the binary set for each influencer being selected or not, i.e., $\mathcal{A}_k = \{0, 1\}$ and $\mathcal{A} = \{0, 1\}^K$. The reward is assumed to be uniquely defined by $r_{k,h}(s_k, a_k, s_k^{'}), \forall s_k, s_k^{'} \in \mathcal{S}_k, a_k \in \mathcal{A}_k$, which can further be bounded and simplified to $r_{k,h}(s_k, a_k)$. Recall the reward is the count of new activations, which is naturally bounded by the total number of users.

As in the RL literature in general, at the beginning of each episode, the initial states $s_{k,1}, \forall k \in \mathcal{K}$ are given and thereon the learning agent interacts with the episodic MDPs. It observes the states $s_{k,h} \in \mathcal{S}_k$ at each step $h \in [H]$ and proceeds to take an overall action $a_h \in \mathcal{A}$. The MDPs of the selected influencers are transitioning according to their transition dynamics $\mathbb{P}_k$ to the new states $s_{k,h+1}$. When an influencer is not selected, i.e., $a_{k,h} = 0$, the state remains the same, $\mathbb{P}_{k,h}(s|s, 0) = 1$. The state transition dynamics are stochastic and unknown upon selecting an influencer. The final state of each MDP is $s_{k,H+1}$, where no action can be taken anymore and the reward is consequently zero. Obviously, in this setup, the MDPs may not reach their final state before the total campaign budget H is spent.

The goal is to maximize the number of distinct activations at the end of a campaign, as defined in Problem 4, leveraging information from previous rounds *and previous campaigns*. This optimization problem expressed in terms of episodic MDPs has as the solution the optimal policy $\pi^* \in \operatorname{argmax}_{\pi \in \Pi} V^{\pi}(s), \forall s \in \mathcal{S}$, where Π is the policy set, and $\mathcal{S} = \bigcup_{k \in \mathcal{K}} \mathcal{S}_k$. The agent aims to learn the optimal policy $\pi^* : \mathcal{S} \times [H] \to \mathcal{A}$.

The policies are evaluated by their corresponding value functions or their action-value functions. The Bellman equations for these values are the following :

$$V_k^{\pi}(h, s_k) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r_{k,\tau}(s_{k,\tau}, \pi(s_{k,\tau}, \tau)) | s_{k,h} = s_k\right], \\ \forall s_k \in \mathcal{S}_k, \forall h \in [H].$$

$$Q_k^{\pi}(h, s_k, a_k) = r_{k,h}(s_k, a_k) + \\ \mathbb{E}\left[\sum_{\tau=h}^{H-1} r_{k,\tau}(s_{k,\tau}, \pi(s_{k,\tau}, \tau)) | s_{k,h} = s_k, a_{k,h} = a_k\right], \\ \forall (s_k, a_k) \in \mathcal{S}_k \times \mathcal{A}_k, \forall h \in [H].$$

The optimal value function is

$$V^*(h,s) = \max_{a_k \in \mathcal{A}_k, k \in \mathcal{K}} Q_k^*(h,s,a_k),\ \forall s \in \mathcal{S}.$$

So the learning agent chooses its actions according to the greedy policy with respect to the estimated action-value functions:

$$\pi(h,s) = \operatorname*{argmax}_{a_k \in \mathcal{A}_k, k \in \mathcal{K}} Q_k^{\pi}(h,s,a_k).$$

The flow of this learning process hence depends on how the Q function is estimated, as detailed next.

4.3 RL with average GT estimators and LSVI learned modifiers

The Multi-Armed Bandit problem, and many of its well-known variants such as stochastic bandits, contextual bandits [Lattimore and Szepesvári (2020)], contextual bandits with linear rewards [Chu et al. (2011)], with generalized linear rewards [Filippi et al. (2010a)], or with Good-Turing reward estimators [Lagrée et al. (2018)], restrict the reward random variables to be independent and identically distributed, i.e. independent of the previous action choices and rewards. However, the choice of actions may alter the state of the environment. In more general Reinforcement Learning problems, theoretical guarantees for the estimators can be obtained without ignoring the state of the environment.

4.3.1 The Good-Turing estimator

[Lagrée et al. (2018)] have used the GT estimator for the potential new activations to be seen in online influence maximization campaigns. In this work, the influencer's diminishing reward is modelled as a function of the number of selections $\gamma(n_{k,t})$, and applied it as a modifier to the GT

estimator (denoted Fat-GT, for Fatigue-aware Transformation of the Good-Turing estimator):

$$\hat{R}_{k,h} = \frac{1}{n_{k,h}} \sum_{j \in A_k} U^{\gamma}_{n_{k,h}}(j) \tag{4.3}$$

where

$$\begin{aligned} U^{\gamma}_{n_{k,h}}(j) \quad = \quad & \sum_{1 \le s \le n_{k,h}} \mathbb{I}\{X_{k,1}(j) = \cdots = X_{k,s-1}(j) = X_{k,s+1}(j) = \\ & \cdots = X_{k,n_{k,h}}(j) = 0, X_{k,s}(j) = 1\} \times \frac{\gamma(n_{k,h}+1)}{\gamma(s)}, \end{aligned}$$

while $n_{k,h}$ is the number of selections of influencer k at round h, A_k is the set of basic nodes reachable by influencer k, $\gamma(n_{k,h})$ is the fatigue function (e.g., $\frac{1}{n_{k,h}}$), $X_{k,s}(j)$ is the i.i.d. random variable equal to 1 if influencer k activates basic node j at round s.

In short, the influencer k's estimated remaining potential $\hat{R}_{k,h}$ at round h is the average number of discounted hapaxes.

In Chapter 3, we adapted the Good-Turing estimator to the new activations in the scenario where context information would be available at the beginning of each round in the campaign. To this end, the modifier to the GT estimator is replaced by a function of both the influencer's number of selections and the round's context: $\gamma(\langle \hat{\theta}_{k,t}, Y_t \rangle, n_{k,t})$. The influencer's potential within a given context is assumed to be well-represented by the scalar product of the round's context Y_t and an estimated unknown quantity $\hat{\theta}_{k,t}$ for that influencer.

4.3.2 `LSVI-GT-UCB`

Motivated by the potential gain from using available historical information when choosing actions, we propose the novel algorithm `LSVI-GT-UCB`.

The state of each influencer's MDP is composed by concatenating the context given by the environment at the beginning of each step in the horizon $Y_{t,h} \in \mathbb{R}^d$, and the reward received by the respective influencer upon its previous selection within the current episode $r_{k,t,n_{k,t,h}}$.

At a high level, `LSVI-GT-UCB` combines the LSVI algorithm of [Jin et al. (2020)], based on linear

regression estimation, to which we add the Good-Turing estimator approach. For each influencer we have an MDP $(\mathcal{S}_k, \mathcal{A}_k, H, \mathbb{P}_k, r_k)$, assumed to be linear via a feature map $\phi_k : \mathcal{S}_k \times \mathcal{A}_k \to \mathbb{R}^d$ [Bradtke and Barto (1996); Melo and Ribeiro (2007); Jin et al. (2020)]. Since each influencer has their own MDP, their action set is binary, i.e. $\mathcal{A}_k = \{0, 1\}$.

The linear regression data is created for each influencer, based on its historical selections and rewards:

$$y_{k,\tau,h} = r_{k,\tau,n_{k,\tau,h}} + \hat{V}^{LSVI}_{k,t,h+1}(s_{k,\tau,h+1}), \tau \in [1, t], \tag{4.4}$$

where $\hat{V}^{LSVI}_{k,t,h+1}(\cdot) = \max_{a \in \mathcal{A}_k} \hat{Q}^{LSVI}_{k,t,h+1}(\cdot, a)$. The data is used in the linear regression estimator to which an UCB bound is added:

$$\begin{aligned} \hat{Q}^{LSVI}_{k,t,h}(\cdot,\cdot) &= \langle \phi_k(\cdot,\cdot), \hat{\theta}_{k,t,h} \rangle + \zeta \sqrt{\phi_k(\cdot,\cdot)^T \Sigma^{-1}_{k,t,h} \phi_k(\cdot,\cdot)}, \text{where} \\ \hat{\theta}_{k,t,h} &= \Sigma^{-1}_{k,t,h} \sum_{\tau=1}^{t-1} \phi_k(s_{k,\tau,h}, a_{k,\tau,h}) y_{k,\tau,h}, \\ \Sigma_{k,t,h} &= \eta \cdot I_d + \sum_{\tau=1}^{t-1} \phi_k(s_{k,\tau,h}, a_{k,\tau,h}) \phi_k(s_{k,\tau,h}, a_{k,\tau,h})^T, \end{aligned} \tag{4.5}$$

where $\zeta = cdH\sqrt{\log(\frac{2dTH}{\delta})}$ as in [Jin et al. (2020)][Theorem 3.1], with an absolute constant $c > 0$, ensures with probability $1 - \delta$ a total regret of $O\left(\sqrt{d^3H^4T\log^2(\frac{2dTH}{\delta})}\right)$, and the penalty factor η ensures a unique minimizer for the regularized least-squares estimator.

In parallel, the Good-Turing estimator for the remaining potential is computed for each step in each influencer's MDP as well. The Fat-GT estimator for the remaining potential, previously introduced in Equation (4.3), is adapted to Problem **??** by maintaining an independent estimator for each episode, as follows:

$$\begin{aligned} \hat{R}_{k,t,h} &= \frac{1}{n_{k,t,h}} \sum_{j \in A_k} U^{\gamma}_{n_{k,t,h}}(j), \text{where} \\ U^{\gamma}_{n_{k,t,h}}(j) &= \sum_{1 \le i \le n_{k,t,h}} \mathbb{I}\{X_{k,t,1}(j) = \dots \\ \cdots &= X_{k,t,h}(j) = 0, X_{k,t,i}(j) = 1\} \frac{\gamma(n_{k,t,h}+1)}{\gamma(i)}, \end{aligned} \tag{4.6}$$

with its respective confidence bound index provided by [Lagrée et al. (2018)][Theorem C.2]:

$$\begin{aligned}\beta_{k,t,h} = (1+\sqrt{2})\sqrt{\frac{\hat{\lambda}_{k,t,h}\log 4h}{n_{k,t,h}}} + \frac{\log 4h}{3n_{k,t,h}}, \text{ where} \\ \hat{\lambda}_{k,t,h} = \frac{\gamma(n_{k,t,h}+1)}{n_{k,t,h}}\sum_{s=1}^{n_{k,t,h}}\frac{r_{k,t,s}}{\gamma(s)}.\end{aligned} \tag{4.7}$$

Furthermore, in order to learn from historical data, an average of the Fat-GT estimators for the given step over the previous and current episodes is computed, as follows:

$$\hat{Q}^{GT}_{k,t,h} = \frac{1}{t}\sum_{\tau=1}^{t}\hat{R}_{k,\tau,h}, \tag{4.8}$$

implementing our interpretation that the state-action value function (Q-function) in an MDP is the influencer's remaining potential:

$$\begin{aligned}Q^{GT}_{k,t,h} &= R_{k,t,h} \\ &= \sum_{j\in A_k}\mathbb{I}\left\{j \notin \bigcup_{i=1}^{h} S(I_{t,i}, Y_{t,i})\right\}\gamma(n_{k,t,h}+1)p_{k,j}(t,h).\end{aligned} \tag{4.9}$$

We derive the optimism bonus for the Q-function estimator in the following theorem:

Theorem 4.3.1. *With probability at least* $1-\delta$*, for* $\lambda_{k,t,h} = \gamma(n_{k,t,h})\sum_{j\in A_k}p(j)$ *and* $\beta_{k,t,h} = (1+\sqrt{2})\sqrt{\frac{\lambda_{k,t,h+1}\log 4/\delta}{n_{k,t,h}}} + \frac{1}{3n_{k,t,h}}\log\frac{4}{\delta}$*, the following holds:*

$$\begin{aligned}\frac{-1}{t}\sum_{\tau=1}^{t}\left(\beta_{k,\tau,h} + \frac{(n_{k,\tau,h}+1)\lambda_{k,\tau,h}}{n_{k,\tau,h}}\right) &\leq Q^{GT}_{k,t,h} - \hat{Q}^{GT}_{k,t,h} \\ &\leq \lambda_{k,t,h} + \frac{1}{t}\sum_{\tau=1}^{t}\beta_{k,\tau,h}.\end{aligned}$$

Proof. Estimating the Q-function with the averaged Fat-GT estimators as in Equation (4.8), the estimator's confidence interval is:

$$Q^{GT}_{k,t,h} - \hat{Q}^{GT}_{k,t,h} = R_{k,t,h} - \frac{1}{t}\sum_{\tau=1}^{t}\hat{R}_{k,\tau,h}.$$

We know from [Lagrée et al. (2018)][Theorem C.2] that:

$$-\beta_{k,\tau,h} - \frac{\lambda_{k,\tau,h}}{n_{k,\tau,h}} \leq R_{k,\tau,h} - \hat{R}_{k,\tau,h} \leq \beta_{k,\tau,h}, \forall \tau \in [1,t].$$

Aggregating the confidence bounds from all episodes, we obtain:

$$\begin{aligned}\frac{-1}{t}\sum_{\tau=1}^{t}\left(\beta_{k,\tau,h} + \frac{\lambda_{k,\tau,h}}{n_{k,\tau,h}}\right) \leq \frac{1}{t}\sum_{\tau=1}^{t}\left(R_{k,\tau,h} - \hat{R}_{k,\tau,h}\right) \leq \frac{1}{t}\sum_{\tau=1}^{t}\beta_{k,\tau,h} \\ \Leftrightarrow \frac{-1}{t}\sum_{\tau=1}^{t}\left(\beta_{k,\tau,h} + \frac{\lambda_{k,\tau,h}}{n_{k,\tau,h}}\right) + R_{k,t,h} - \frac{1}{t}\sum_{\tau=1}^{t}R_{k,\tau,h} \\ \leq R_{k,t,h} - \frac{1}{t}\sum_{\tau=1}^{t}\hat{R}_{k,\tau,h} \leq R_{k,t,h} - \frac{1}{t}\sum_{\tau=1}^{t}R_{k,\tau,h} + \frac{1}{t}\sum_{\tau=1}^{t}\beta_{k,\tau,h}.\end{aligned}$$

Having the remaining potential of influencer k at round h of episode t defined as in Equation (4.9), we can obtain that:

$$\begin{aligned}0 \leq R_{k,t,h} \leq \lambda_{k,t,h} \\ \Leftrightarrow \frac{-1}{t}\sum_{\tau=1}^{t}\lambda_{k,\tau,h} \leq R_{k,t,h} - \frac{1}{t}\sum_{\tau=1}^{t}R_{k,\tau,h} \leq \lambda_{k,t,h}.\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{-1}{t}\sum_{\tau=1}^{t}\left(\beta_{k,\tau,h} + \frac{(n_{k,\tau,h}+1)\lambda_{k,\tau,h}}{n_{k,\tau,h}}\right) \leq Q^{GT}_{k,t,h} - \hat{Q}^{GT}_{k,t,h} \\ \leq \lambda_{k,t,h} + \frac{1}{t}\sum_{\tau=1}^{t}\beta_{k,\tau,h}.\end{aligned}$$

which concludes our proof. □

The Q-function is finally estimated by optimistically choosing from the linear regression-based Q estimator and the GT estimators for each potential influencer:

$$\hat{Q}_{k,t,h}(\cdot,\cdot) = \max\left\{\hat{Q}^{LSVI}_{k,t,h}(\cdot,\cdot), \hat{Q}^{GT}_{k,t,h} + \lambda_{k,t,h} + \frac{1}{t}\sum_{\tau=1}^{t}\beta_{k,\tau,h}\right\}. \tag{4.10}$$

The L influencers having the highest value of $\hat{Q}_{k,t,h}(\cdot,\cdot)$ are then chosen. This process is outlined

in Algorithm 4, whose flow is briefly described next.

We assume that the environment provides a context at the beginning of each step, and each MDP's state is computed by concatenating (i) the influencer's number of selections, and (ii) the reward resulting from playing that influencer at its last selection. `LSVI-GT-UCB` starts the first episode by playing each influencer once in order to gather initial information. Then, for the following steps, it proceeds by computing for each influencer k the two Q-function estimators: $\hat{Q}^{LSVI}_{k,t,h}$ and $\hat{Q}^{GT}_{k,t,h}$. The former is computed using regularized least squares, as in [Jin et al. (2020)]. The latter is computed with the formula from Equation (4.8), with its optimism bonus given by Theorem 4.3.1. Finally, the learning agent chooses to play the L influencers with the highest estimated Q-functions, observes the reward, and updates the statistics.

By this flow, `LSVI-GT-UCB` learns *in parallel* the linear and the GT estimators from feedback collected by either of them, and chooses its action with optimism not only from the estimator's UCB, but also from the highest estimated remaining potential with either method.

4.4 Experiments

The algorithm `LSVI-GT-UCB` is tested on three datasets: one with synthetically generated data, and two based on real-world data. Its performance is compared to `LSVI-UCB` [Jin et al. (2020)], and `Fat-GT-UCB` [Lagrée et al. (2018)]. `LSVI-UCB` is originally designed to use upper confidence bounds for linear function approximation MDPs, and can be adapted to Problem 4 by equating an episode with a campaign. `Fat-GT-UCB`, on the othe hand, is run independently between episodes. As a comparison metric, we use the total sum of cumulative rewards over all campaigns. In addition to these two state-of-the-art solutions, we can also design two other baselines: `RL-Fat-GT-UCB` is created by adapting `Fat-GT-UCB` to learn from the previous episodes by averaging the GT estimators over the episode, for each step in the horizon; and `LSVI-UCB -separate thetas` is created by adapting `LSVI-UCB` to learn an estimator per influencer. This modification enables the combination of the estimator of the Q-function with the GT estimator

Algorithm 4 LSVI-GT-UCB

Input: Number of influencers L per round, penalty factor η, the ambient dimension d of the feature space, feature maps ϕ_k.
Initialize $\hat{Q}_{k,1,h}(s_{k,t,h}, a_{k,t,h}) = 0, \forall (s_{k,t,h}, a_{k,t,h}) \in \mathcal{S}_k \times \mathcal{A}_k$.
for episodes $t = 1, \ldots, T$ **do**
  if first episode $t = 1$ **then**
    for step $k = 1, \ldots, K$ **do**
      Receive the arbitrary context $Y_{t,h}$,and create the state
      $s_{k,t,h} = [Y_{t,h}|0|0]$.
      Play influencer k.
      Observe rewards $r_{t,h}$, and next state $s_{k,t,h+1}$.
      Update $n_{k,t,h} = 1, \Sigma_{k,t,h} = \eta \cdot I_d + \phi_k(s_{k,t,h}, 1)\phi_k(s_{k,t,h}, 1)^T$.
    end for
  else
    for step $h = H, \ldots, 1$ (or until K, for the first episode) **do**
      Receive the arbitrary context $Y_{t,h}$.
      for influencer $k = 1, \ldots, K$ **do**
        Create the state $s_{k,t,h} = [Y_{t,h}|n_{k,t,h}|r_{k,t,n_{k,t,h}}]$, and set
        $\hat{\theta}_{k,t,H+1} = \mathbf{0}$,
        $\hat{Q}^{LSVI}_{k,t,H+1}(s_{k,t,h}, a_{k,t,h}) = \langle \phi_k(s_{k,t,h}, a_{k,t,h}), \hat{\theta}_{k,t,H+1} \rangle = 0$.
        Compute the linear regression data $y_{k,\tau,h}, \forall \tau \in [1, t]$
        (as in Equation (4.4)).
        Calculate the regularized least-squares estimator (as in Equation (4.5)).
      end for
    end for
    for step $h = 1, \ldots, H$ **do**
      for influencer $k = 1, \ldots, K$ **do**
        Compute Fat-GT estimator $\hat{Q}^{GT}_{k,t,h}$ as in Equation (4.8).
        Compute the influencer k's Q-function estimator
        (as in Equation (4.10)).
      end for
      Play action $a_{t,h}$ made of the L influencers with the highest
      estimated Q-functions $\hat{Q}_{k,t,h}$.
      Update $n_{k,t,h} = n_{k,t,h-1} + 1, \forall a_{t,h}[k] = 1$.
      Observe reward $r_{t,h}$, and next states $s_{k,t,h+1}$.
    end for
  end if
end for

in order to estimate an influencer's remaining potential.

4.4.1 Real-world Datasets

Experiments are run on two real-world datasets, Sina Weibo [Zhang et al. (2013)] and Twitter (Section 3.5). The dataset statistics are presented in Table A.2.

The *Sina Weibo* dataset containts a log of posts (equivalent to tweets) with each posts's text being encoded as a distribution over 100 topics computed using Latent Dirichlet Allocation. The dataset contains the topic distribution for each post, the reposting logs containing the list of unique users which had reposted the post, and information about the original author. We processed and merged this information in order to obtain in the end a file containing the original post, its author/the influencer's ID, the set of basic user IDs which reposted that message, and the topic distribution for the message. The set of influencers is found by taking the ones having the highest number of reposts, and all the tweets of the other influencers are filtered out. During the experiments, random contexts, i.e. topic distributions, are chosen for each round from all the available contexts in the dataset. At the beginning of each round the context is provided by the environment, an algorithm chooses the influencer(s), and a post for the pair (influencer ID, context) is sampled from the log. The *new activations* are given by discounting the previously seen basic user IDs from the sampled post's set of user IDs.

The *Twitter* dataset is created from a crawled dataset from Twitter with tweets from August 2012. We have extracted, using K-means, 24 centroids from the *glove-twitter-200*[1] vocabulary, and each tweet's text was processed by encoding each word and replacing it with its nearest centroid. The original tweet text is then replaced by a distribution over 24 centroids. Each tweet contains a set of node IDs representing the users which retweeted it. The set of influencers, as in the case of Sina Weibo, are chosen as the ones containing the highest number of retweeting users. The experimental simulation process is the same as the one described for Sina Weibo.

[1] `https://nlp.stanford.edu/projects/glove/`

4.4.2 Synthetic Data

We also generated a synthetic dataset in the following way. First, a graph is generated using the Albert-Barabàsi model [Barabási et al. (2016)] for $30{,}000$ nodes, each influecer is chosen using their degree, i.e., the K highest degrees in the graph. Then, the activation probability of each node attached to these influencers is computed using a sigmoid function of the scalar product of the randomly generated context and a randomly chosen feature vector, specific to the node. are chosen with MaxDegree, each nodes activation probability is computed with a sigmoid function of the scalar product of the round's randomly generated context and the node's randomly chosen feature vector. For each dimension, the feature vectors are sampled from a normal distribution $\mathcal{N}(1, 3)$, and the contexts are sampled from another normal distribution $\mathcal{N}(1, 0.1)$. The variance in the distribution of user profiles is greater than that in the distribution of contexts to mark the greater difference that can appear between users compared to differences between messages from a campaign. The diffusion model is assumed to be the Independent Cascade [Kempe et al. (2003)].

4.4.3 Results

The results of the experiments are averaged over 50 runs, and the algorithms are run for 50 episodes each with a horizon of 30 rounds, i.e. 1500 rounds in the end. For all the LSVI based algorithms the exploration factor is $\beta = c \cdot dH\sqrt{\log(\frac{2dT}{p})}$, where $c > 0$ is an absolute constant, $T = 50$ is the number of episodes, $H = 30$ is the number of steps, d is the dimension of the feature space. For Sina Weibo and the synthetic dataset, an absolute constant $c = 1$ performs well. However, for the Twitter dataset we had to choose a much smaller absolute constant, $c = 0.0005$, to have an exploration factor suitable for the scale of the reward. The dimension of the feature space d depends on the how the state is constructed for each algorithm. This follows the theoretical results of Theorem 3.1 in Jin et al. (2020), with probability set as $1 - p = 0.99$.

The results of Sina Weibo – Figure 4.1 – show that, in terms of cumulative reward, `LSVI-GT-UCB` outperforms the other methods for $L = 2$ and $L = 5$ especially. For $L = 1$ it is competitive with

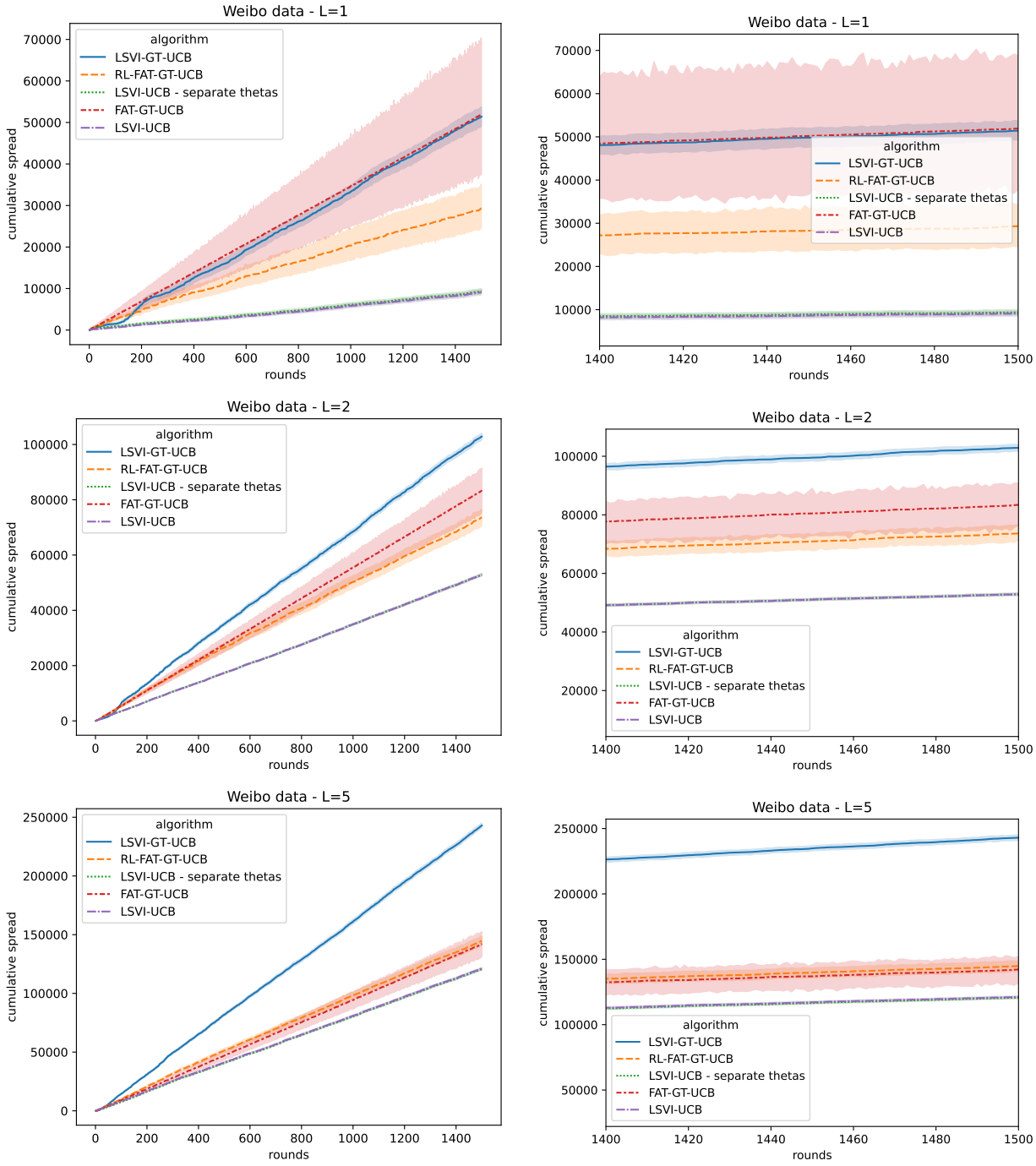

Figure 4.1: Sina Weibo - Cumulative rewards, full plot (left column), plot zoomed to last 100 rounds (right column).

RL-Fat-GT-UCB, but it exhibits much lower variance in the rewards, making it a more reasonable choice in practice.

On the other hand , in the Twitter dataset – Figure 4.2 –, our algorithm clearly outperforms other algorithms for $L = 1$. For $L = 2$ and $L = 5$ it seems that the results of several algorithms – including ours – are essentially identical, with low variance. This may indicate that the dataset characteristics lead to a saturation of the rewards when more influencers are chosen. The algorithms Fat-GT-UCB and RL-Fat-GT-UCB perform similarly.

On the synthetic dataset – Figure 4.3 –, we see results similar to the ones on the Twitter dataset. Our algorithm outperforms the other algorithms for $L = 1$, and $L = 2$. For $L = 5$ the LSVI-based algorithm with an estimator for each influencer is stronger than the Good-Turing based estimator. We witness again a saturation of the rewards when more influencers are chosen, which explains the periodical flattening in the graphs of the reward functions. The large final cumulative rewards are possible due to the reset of counting the new activations at the start of each episode.

4.5 Deep Reinforcement Learning for ECIMP

In this section, we present a third direction of research for solving the *(Episodic) Contextual Influence Maximization with Persistence* problem. This work is inspired by the solution provided by Khalil et al. (2017); Li et al. (2019). Li et al. (2019) later adapted it to maximize influence, proposing the DISCO framework (albeit with rather incomplete and unclear presentation and evaluation).

4.5.1 Generic framework

Khalil et al. (2017) designed a framework for learning approximate solutions to various NP-hard combinatorial optimization problems. The framework is an ϵ-greedy meta-algorithm, Q-learning for the Greedy Algorithm, which estimates a Q-function by learning a predefined set of parameters Θ from different instances of the same problem, i.e. one of the NP-hard problems.

Once the problem to solve is chosen, the algorithm is run for T episodes each with the horizon

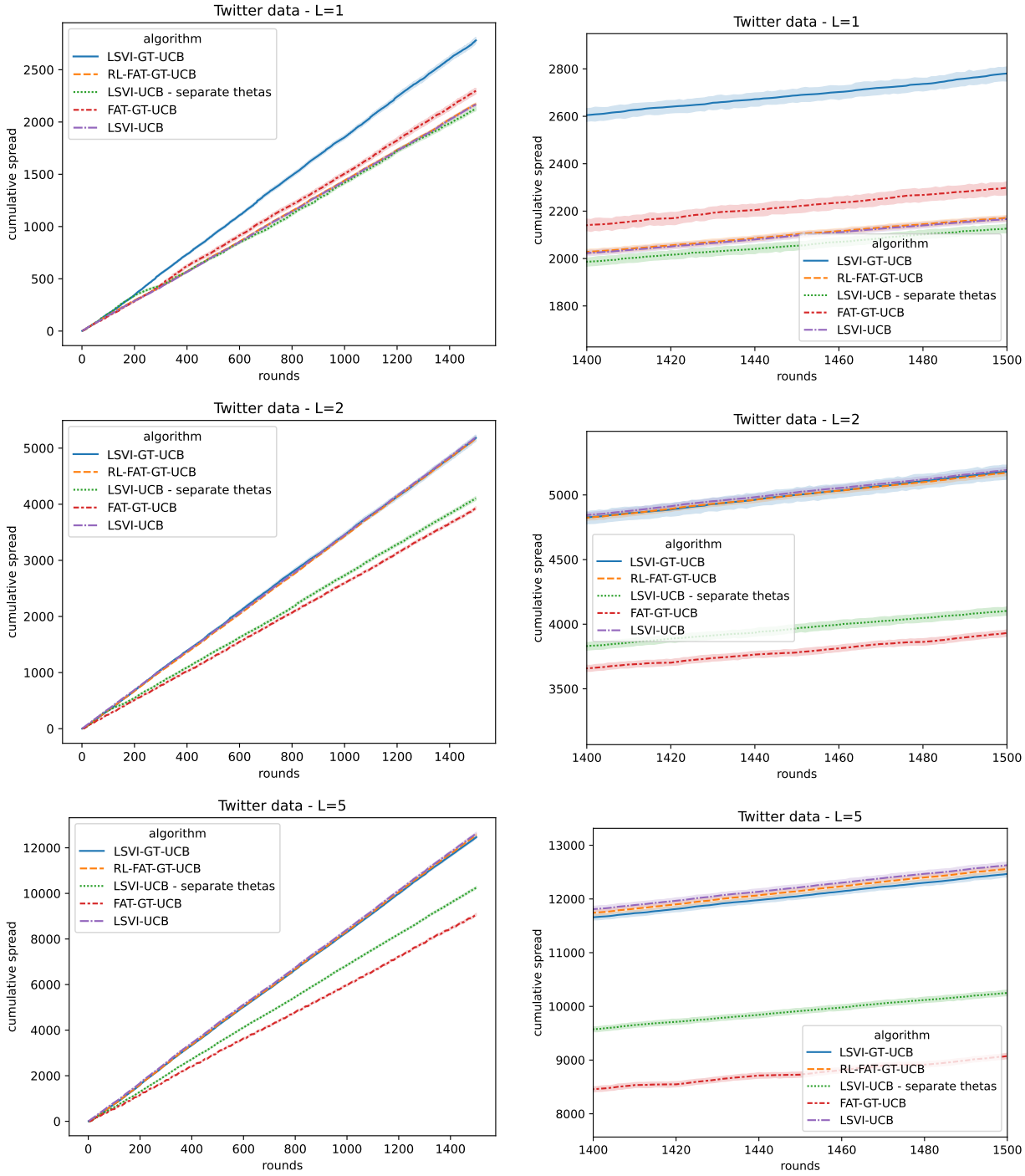

Figure 4.2: Twitter - Cumulative rewards, full plot (left column), plot zoomed to last 100 rounds (right column).

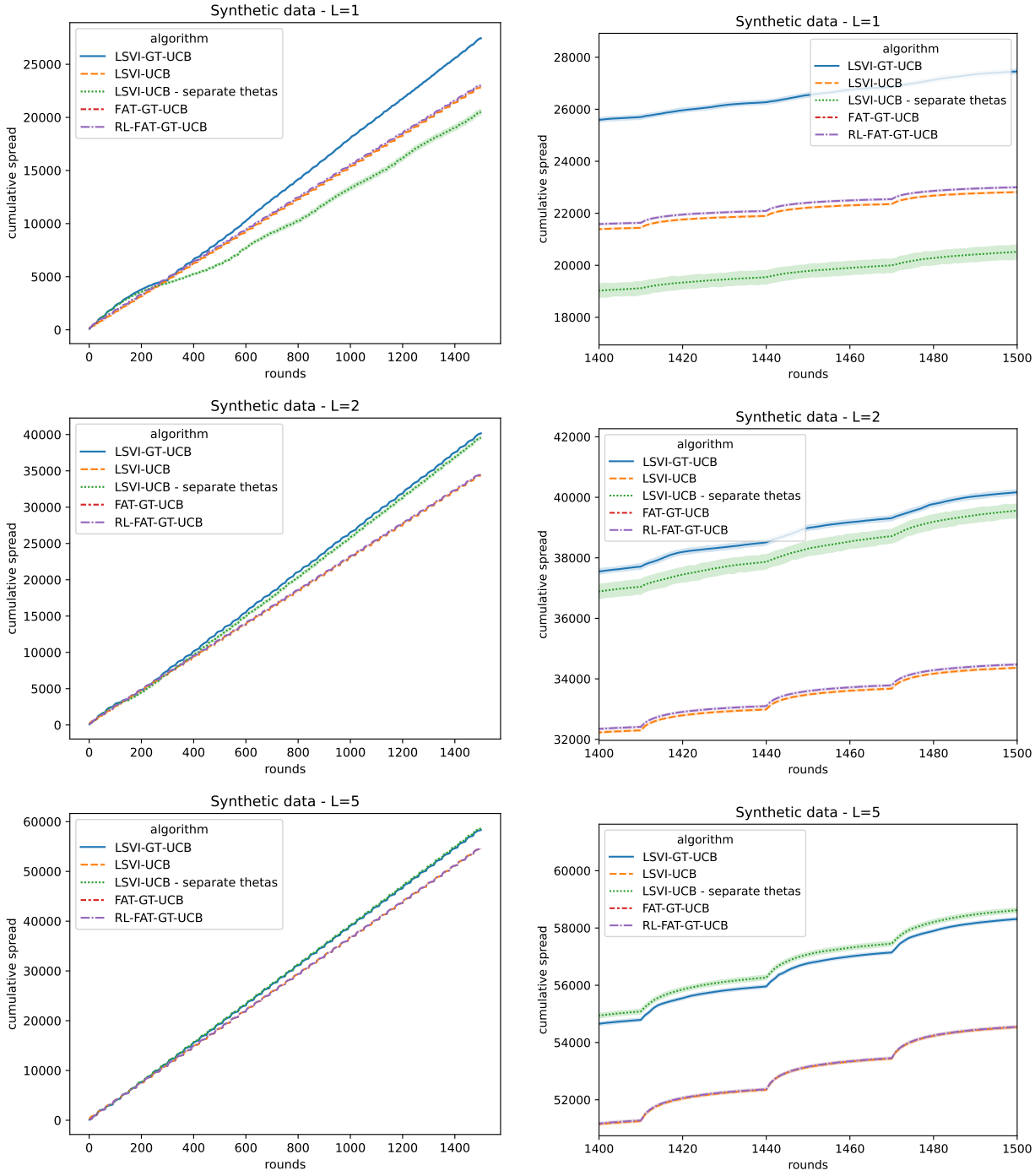

Figure 4.3: Synthetic - Cumulative rewards, full plot (left column), plot zoomed to last 100 rounds (right column).

H. At the beginning of each episode, a graph $G(V, E)$ is sampled from a graph distribution $\mathbb{D}$, the learning agent makes decisions and learns from its experience until the horizon or termination criteria is met. At each step h within the episode's horizon H, and the learning agent chooses a seed node k_h either randomly with probability ϵ, or the one which maximizes the estimated Q-function:

$$v_h = \begin{cases} \text{random node } v \in \bar{S}_h & \text{w.p. } \epsilon, \\ \text{argmax}_{k \in \bar{S}_h} \hat{Q}(\iota(S_h), v; \Theta) & \text{otherwise}, \end{cases} \tag{4.11}$$

where S_h is the partial solution at step h and $\bar{S}_h = V \setminus S_h$ is its complement. The function $\iota(S)$ is a helper function needed only for some NP-hard problems when a combinatorial structure of the partial solution is required, e.g. the insertion operation for the Traveling Salesman Problem. The Q-function estimator is a function of the parameters $\Theta = (\theta_1, \dots, \theta_7)$ to be learned. It estimates the value of selecting the seed node v for state $\iota(S_h)$ as follows:

$$\hat{Q}(\iota(S_h), v; \Theta) = \langle \theta_5, \text{relu}([\theta_6 \sum_{u \in V} \mu_u^{(H)}, \theta_7 \mu_v^{(H)}]) \rangle, \tag{4.12}$$

where $\theta_5 \in \mathbb{R}^{2d}, \theta_6, \theta_7 \in \mathbb{R}^{d \times d}$, and $\mu_v^{(H)}, \forall v \in V$ is node v's embedding at step H, d is the dimension of the feature space of nodes, $x_v = 1$ if $v \in S$ is a binary variable indicating whether v is already in the partial solution or not, and $\text{relu}(x) = \max(0, x)$. The node embeddings are updated at each step according to the `structure2vec` graph embedding model [Dai et al. (2016)]:

$$\mu_v^{h+1} = \text{relu}(\theta_1 x_v + \theta_2 \sum_{u \in \mathcal{N}(v)} \mu_u^{(t)} + \theta_3 \sum_{u \in \mathcal{N}(v)} \text{relu}(\theta_4 w(v, u))), \tag{4.13}$$

$\theta_1 \in \mathbb{R}^d, \theta_2, \theta_3 \in \mathbb{R}^{d \times d}, \theta_4 \in \mathbb{R}^d$ are the model used by this These embeddings are updated in each step and capture each node's properties in the context of its graph neighborhood. The parameters Θ are updated either at each step or periodically, based o a batch sampled from history, with stochastic gradient decent over:

$$(y - \hat{Q}(\iota(S), v_h; \Theta))^2,$$

where $y = r(S_h, v_h) + \max_{v'} \hat{Q}(\iota(S_{h+1}, v'; \Theta)$.

The reward function $r(S, v) = c(\iota(S \cup \{v\}), G) - c(\iota(S), G)$ is defined as the change in the cost function. The choice of the cost function, in turn, depends on the problem being solved. Depending on the problem, there may also be a termination criterion, e.g. for TSP the tour/solution includes all nodes.

4.5.2 Contribution

The ECIMP problem can be reduced to the Set Covering Problem (SCP), and the RL components are defined as in Table 4.1.

Problem	Set Covering Problem (SCP)	ECIMP
State	subset of nodes selected so far	list of *influencers* selected so far
Action	add node to subset	(re)add node to list
Helper function	none	none
Reward	-1	new activations
Termination	all nodes in G have been covered	budget of $T \times H$ rounds

Table 4.1: Definition of RL components.

ECIMP assumes that a context $Y_h \in \mathbb{R}^d$ is provided by the environment at each step. We incorporate this side-information in the Equation 4.13 as follows:

$$\mu_v^{h+1} = \text{relu}(Y_h + \theta_1 x_v + \theta_2 \sum_{u \in \mathcal{N}(v)} \mu_u^{(t)} + \theta_3 \sum_{u \in \mathcal{N}(v)} \text{relu}(\theta_4 w(v, u))), \tag{4.14}$$

$Y_h, \theta_1 \in \mathbb{R}^d, \theta_2, \theta_3 \in \mathbb{R}^{d \times d}, \theta_4 \in \mathbb{R}^d$. One of the limitations we face is that `structure2vec` requires the topology of the graph, whereas we assume that we do not have such information. One way to overcome this obstacle is to construct an initial bipartite graph, if logs are available, or to start from scratch and, in either case, update the graph structure from feedback. We retain the assumption for ECIMP that a set of seed nodes, i.e. potential influencers, is available. Considering this assumption and this graphic structure, node embeddings for influencers can still be calculated with Equation 4.14, but the formula is different for basic user embeddings, being reduced

to a single weighted binary decision variable $\mu_u^{(h+1)} = \text{relu}(Y_h + \theta_1 x_u), \forall u \in V \setminus \mathcal{K}$. In this case, it indicates whether the basic user is enabled, compared to its meaning in the work of Khalil et al. (2017) where it indicates whether the node has been selected by the greedy policy. In this design, information is quite limited. To overcome this issue, we construct a synthetic neighborhood for basic nodes by adding a new edge between all nodes that share an influencer, e.g. Figures 4.4. This allows Equation 4.14 to be used for basic nodes as well.

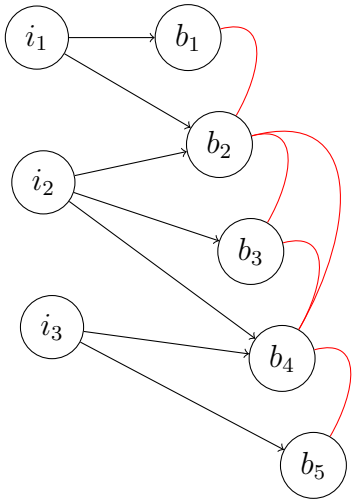

Figure 4.4: Bipartite graph extension for `structure2vec`.

We adapt `Q-learning for the Greedy Algorithm` from Khalil et al. (2017) to maximize influence by building our solution for ECIMP for a model framework that closely resembles the one of `DISCO`. Unlike the SCP problem, ECIMP allows the re-selection of the same influencer in multiple rounds. Thus the learning agent makes the decision according to the following:

$$v_h = \begin{cases} \text{random node } v \in \bar{S}_h & \text{w.p. } \epsilon, \\ \text{argmax}_{k \in \mathcal{K}} \hat{Q}(S_h, v; \Theta) & \text{otherwise,} \end{cases} \tag{4.15}$$

We implemented the Algorithm 5 and performed initial experiments on the Twitter dataset. We separated the Twitter logs into two sets, one to create a bipartite graph as in Fig. 4.4, and the other to extract the side information for the diffusion campaigns. The campaigns are divided into 1000 initial ones for training the parameters, and the next 1000 ones for evaluating the solution. From the initial results, presented in Figure 4.5, it does not seem that the average reward

Algorithm 5 Deep RL for ECIMP

Input: Batch of training network, a positive number H, experience replay memory $\mathcal{M}$ with capacity N.
Output: parameters $\Theta = \{\theta_1, \ldots, \alpha_7\}$
for episodes $t = 1, \ldots, T$ **do**
    Given a network G (which includes the potential influencers) and set seed set $S = \emptyset$;
    for step $h = 1, \ldots, H$ **do**
        Get the context Y_h.
        Compute with Equation 4.14 the embedding for each node $v \in V$;
        Calculate with Equation 4.12 the Q value of each potential influencer $\mathcal{K}$;
        Choose with Equation 4.15 the influencer to play;
        Add k_h to S. Update $S, \bar{S}$;
        if $h \geq \delta$ **then**
            Add tuple $\langle S_h, k_h, \sum_h^{h+\delta} R_h, S_{h+\delta} \rangle$ to $\mathcal{M}$;
            Sample random batch from $B \sim \mathcal{M}$;
            Update Θ by SGD with B;
        end if
    end for
end for

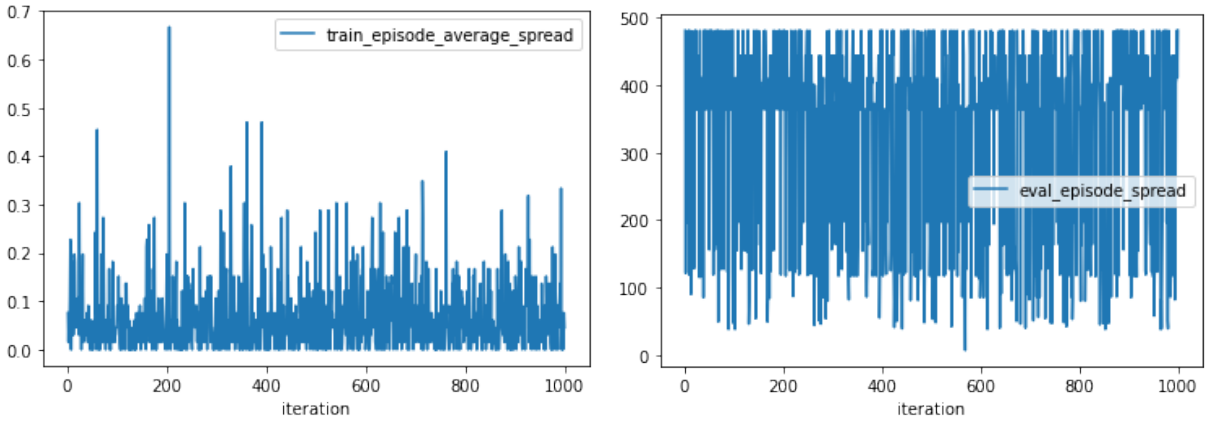

Figure 4.5: Average cumulative new activations.

improves from one campaign to another. This may be due to the small size of the created graph, from which the algorithm samples an even smaller graph, which can quickly lead to reward saturation. A possible improvement in the results can be obtained by allocating more data from the logs to create the distribution of graphs and by using an entire graph for each episode. For future work, Algorithm 5 would be better contrasted to competitors, e.g., a potential one being `LSVI-GT-UCB`.

4.6 Conclusions

We presented in this chapter a new algorithm, `LSVI-GT-UCB`– a combination of the LSVI approach for linear function approximation and the Good-Turing estimator used in multi-armed bandits for estimating missing mass – with an application to contextual influence maximization over multiple influence campaigns. `LSVI-GT-UCB` runs the two estimators in parallel, and makes the optimistic choice between them when deciding which influencers to select at each step of the campaign – thus implementing the *optimism in the face of uncertainty* principle. The experimental study, performed on two real-world datasets – Sina Weibo and Twitter – and a synthetically generated one, shows that `LSVI-GT-UCB` is competitive to state-of-the-art baselines, is less susceptible to noise, while allowing learning over multiple campaigns.

Chapter 5

Conclusions and perspectives

5.1 Conclusions

Influence Maximization is a notoriously difficult problem to solve. Under different assumptions about the underlying diffusion process model, it can be reduced to either the Set Covering problem or the Minimum Vertex Cover problem. Both are NP-hard problems, which leads us to look for heuristic or approximation algorithms to solve Influence Maximization problems. Inspired by recent years' trends in social media, we consider a particular scenario for maximizing influence, which mirrors influencer marketing, and we derive from there the characteristics of the problems addressed in this thesis.

We aim to learn *online*, i.e., on the fly, how to select the best influencers to spread the desired information within a diffusion network. An influence campaign is divided into several rounds, and in this thesis, we set the reward at the end of each round to be the number of newly activated basic nodes. Assuming no knowledge of the diffusion topology nor of the diffusion probabilities, but knowing the set of potential influencers to be engaged in an influence campaign and the message to be disseminated, the objective is to find the most suitable influencers for each round, so that the cumulative final reward is maximized. We first formally define this problem in terms of Multi-Armed Bandits (sequential learning without state information), i.e., *Contextual Influence Maximization with Persistence*, and then in terms of Reinforcement Learning with Markov

Decision Processes (sequential learning with state information), i.e., *Episodic Contextual Influence Maximization with Persistence*. We solve both of these problems with Upper Confidence Bound approximation algorithms, as we want to strike a good balance between exploiting influencers known to perform well and exploring other influencers to uncover more of their potential.

Since the reward consists only of new activations per round and due to the lack of other topology and diffusion probability information, we focus on directly estimating every influencer's remaining potential for activating new nodes. This quantity is easier to calculate and is ultimately what we are interested in for selecting the influencers, alleviating the scalability limitations of all state-of-the-art methods that try to directly estimate each influencer's probability of activating a basic node (hence facing a huge parametric space).

We provide a first solution in this direction, for the Multi-Armed Bandit formulation, namely the `LogNorm-LinUCB` algorithm. Under the assumption that the reward scale is normally distributed, this algorithm approximates with linear regression the expected reward scale for each potential influencer; the scale of the reward is the inner product between the disseminated message and the unknown parameter to be estimated for the influencer.

We then present a second, alternative approach for solving the *Contextual Influence Maximization with Persistence* problem, called `GLM-GT-UCB`. We have found the Good-Turing estimator to be versatile in estimating the probability mass of entities from a population not yet seen in the sample, and have adapted it for both online learning problems. To this end, we assigned a Good-Turing estimator to each potential influencer and updated these statistics with each feedback we received from the environment. These estimators are powerful in themselves but are unable to capture side information available for the round, e.g., the message to be disseminated. For this, we apply a side information modifier function to the Good-Turing estimator. `GLM-GT-UCB` assumes that the rewards follow a Poisson distribution and defines this modifier function accordingly.

For the *Episodic Contextual Influence Maximization with Persistence* problem, we assign to each influencer a Good-Turing estimator and a Markov Decision Process that incorporates any information available in the states of that round. Our algorithm, `LSVI-GT-UCB`, selects the influencers

of each round optimistically, between the estimated Q-function and the Good-Turing estimated remaining potential.

All three algorithms, `GLM-GT-UCB`, `LogNorm-LinUCB`, and `LSVI-GT-UCB` are based on theoretically guaranteed Upper Confidence Bounds. Their performance is tested on a synthetically generated dataset and two real-world datasets, one constructed from Twitter logs and one based on publicly available data from Sina Weibo. For `GLM-GT-UCB` and `LogNorm-LinUCB`, we observe that the Good-Turing estimators perform better on the Sina Weibo dataset, while the linearly approximated estimators tend to perform better on the Twitter dataset. There are two differences between the two datasets that could explain the results, (i) the latter has a smaller reward scale compared to the former, and (ii) the latter uses *word2vec* to create the topics, while the former uses *Latent Dirichlet Allocation*. `LSVI-GT-UCB` proves to successfully alternate between the two types of estimators when making decisions.

5.2 Perspectives

Given the empirical results, one direction of further development for solving the *Contextual Influence Maximization with Persistence* problem is to design a framework that allows us to learn which estimator to use from the observed data.

For the *Episodic Contextual Influence Maximization with Persistence* problem, `LSVI-GT-UCB` might have better results with generalized linear function approximation. In this direction, the first step would be analyzing the distribution of the data used at each step in each episode to estimate the Q-function. Another promising research direction for ECIMP is Deep Reinforcement Learning. An immediate improvement to our solution along these lines, Algorithm 5, is to either provide more graphs for the learning process or to remove the graph sampling part entirely since the bipartite graph is already small. Given that the initial graph may not fully reflect the complete social network, i.e. new basic nodes could be discovered with each diffusion campaign, the framework can be extended to update the graph with new knowledge.

Finally, we stress that both the CIMP and ECIMP problems and their respective solutions are

initially motivated by influencer marketing, but are by no means limited to this application scenario. The algorithms proposed in this Ph.D. thesis generically solve Influence Maximization, abstracting away from any online marketing or advertising-related aspects. Therefore, other diffusion problems in an unknown environment, such as anomaly propagation in an electrical grid, may also benefit from the ideas and algorithms proposed here.

Appendix A

GLM-GT-UCB

A.1 Other empirical results

As additional empirical support for assuming in GLM-GT-UCB a Poisson distribution for each influencer's rewards, we inspected the reward distributions per influencer and context, in the Twitter dataset. Given that the reward is a dynamic quantity – as it depends on previously activated nodes – we generate random campaigns and use an Oracle to select the influencer with the highest reward for each round. Then we plot for each influencer and context the empirical distribution of its rewards and the Poisson fit. The results support the assumption for the underlying distributions. Fig. A.1 (supplementary material) illustrates the distributions with the largest number of samples. Note that we cannot perform a similar test in Weibo, due to the large dimensionality of contexts (100 topics), and thus higher contextual diversity, leading to too few samples to create an empirical distribution of rewards for an influencer in a given context.

A.2 GLM-GT-UCB - Theoretical analysis

Proof. The theoretical analysis is performed for one influencer k; it follows the same steps for any other influencer.

Preliminaries:

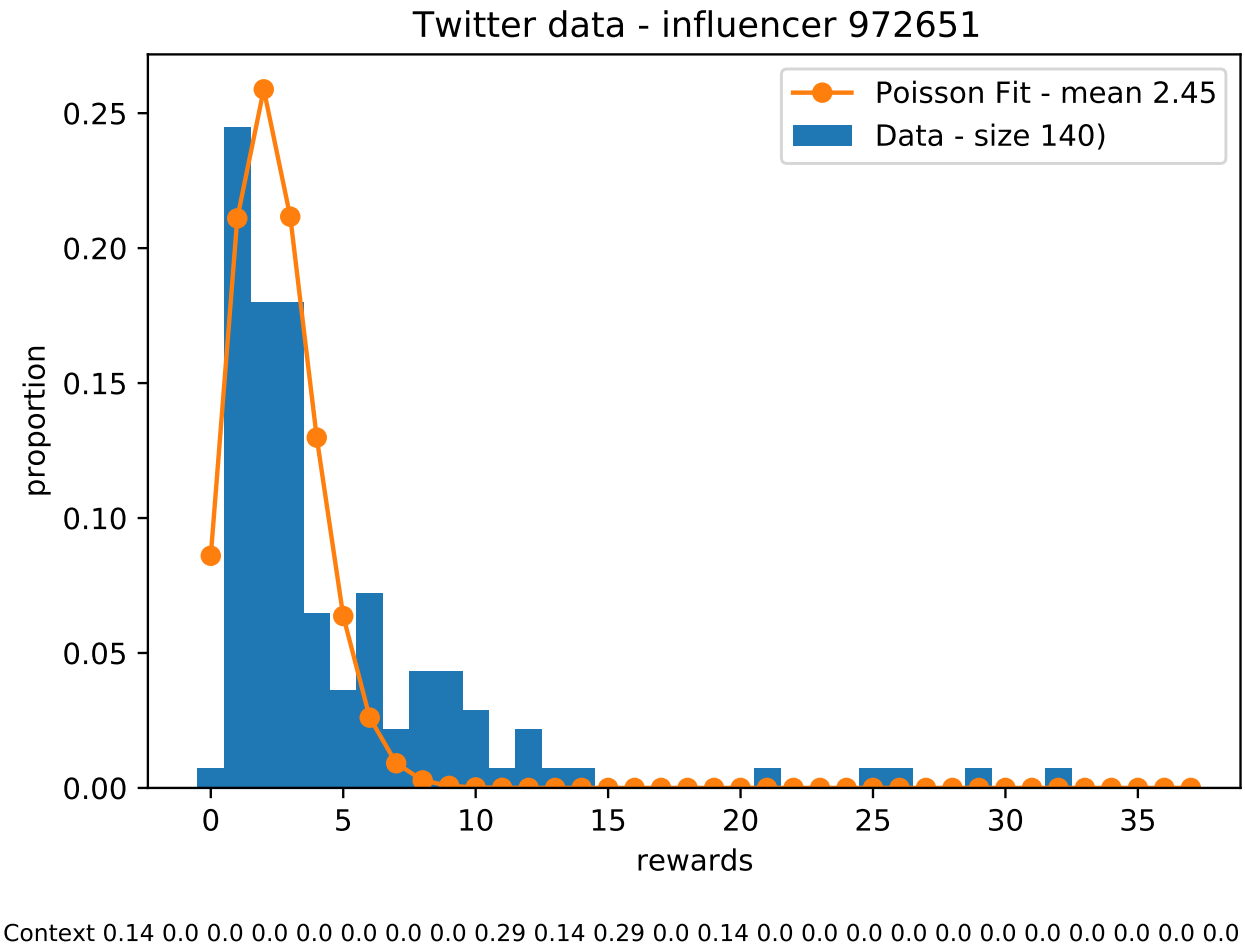

Figure A.1: Reward distributions in Twitter.

Table A.1: Assumptions

$\theta_{k,j} \approx \theta_k$ (the influencer's unkown vector θ_k captures the basic node's trend in activating in a context along its inner probability $p(j)$)
$\alpha(\langle \theta_k, Y_t \rangle, n_k(t)) = e^{f(n_k(t))\langle \theta_k, Y_t \rangle}$
$f(n_k(t)) = \frac{1}{n_k(t)}$
$\|\theta_k\| \leq 1$
$\|Y_t\| \leq 1$

$$p_{k,j}(t) = \alpha(\langle \theta_k, Y_t \rangle, n_k(t))p(j) \tag{A.1}$$

$$P(C_j(t-1)=0) = e^{-\lambda_j \sum_{s=1}^{t-1} \sum_{k' \in I_s} \alpha(\langle \theta_{k'}, Y_s \rangle, n_{k'}(s))} \tag{A.2}$$

$$P(C_{k,j}(t-1)=1) = \sum_{s=1}^{t-1} \lambda_j \alpha(\langle \theta_k, Y_s \rangle, n_k(s)) e^{-\lambda_j \sum_{l=1}^{t-1} \sum_{k' \in I_l} \alpha(\langle \theta_{k'}, Y_l \rangle, n_{k'}(l))} \tag{A.3}$$

$$R_k(t) = \alpha(\langle \theta_k, Y_t \rangle, n_k(t)) \sum_{j=1}^{A_k} \lambda_j \mathbb{I}\{C_j(t-1)=0\} \tag{A.4}$$

$$\mathbb{E}[R_k(t)] = \alpha(\langle \theta_k, Y_t \rangle, n_k(t)) \sum_{j=1}^{A_k} \lambda_j e^{-\lambda_j \sum_{s=1}^{t-1} \sum_{k' \in I_s} \alpha(\langle \theta_{k'}, Y_s \rangle, n_{k'}(s))} \tag{A.5}$$

$$G_k(t) = \alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t)) \frac{1}{n_k(t)} \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{\mathbb{I}\{X_{s,j,k}=1, \{X_{l,j,k'}=0\}_{k' \in I_s \setminus \{k\}}, \{X_{l,j,k'}=0\}_{l \neq s, k' \in I_l}\}}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))} \tag{A.6}$$

$$\begin{aligned}
\mathbb{E}[G_k(t)] &= \alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t)) \frac{1}{n_k(t)} \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{\alpha(\langle \theta_k, Y_s \rangle, n_k(s)) \lambda_j e^{-\lambda_j \sum_{l=1}^{t-1} \sum_{k' \in I_l} \alpha(\langle \theta_{k'}, Y_l \rangle, n_{k'}(l))}}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))} \\
&= \frac{\alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t))}{\alpha(\langle \theta_k, Y_t \rangle, n_k(t)) n_k(t)} \sum_{j=1}^{A_k} \alpha(\langle \theta_k, Y_t \rangle, n_k(t)) \lambda_j e^{-\lambda_j \sum_{l=1}^{t-1} \sum_{k' \in I_l} \alpha(\langle \theta_{k'}, Y_l \rangle, n_{k'}(l))} \\
&\cdot \sum_{s=1}^{t-1} \frac{\alpha(\langle \theta_k, Y_s \rangle, n_k(s))}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))} \\
&= \frac{\alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t))}{\alpha(\langle \theta_k, Y_t \rangle, n_k(t)) n_k(t)} \sum_{s=1}^{t-1} \frac{\alpha(\langle \theta_k, Y_s \rangle, n_k(s))}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))} \alpha(\langle \theta_k, Y_t \rangle, n_k(t)) \\
&\cdot \sum_{j=1}^{A_k} \lambda_j e^{-\lambda_j \sum_{l=1}^{t-1} \sum_{k' \in I_l} \alpha(\langle \theta_{k'}, Y_l \rangle, n_{k'}(l))} \\
&= \frac{\alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t))}{\alpha(\langle \theta_k, Y_t \rangle, n_k(t))} \frac{\mathbb{E}[R_k(t)]}{n_k(t)} \sum_{s=1}^{t-1} \frac{\alpha(\langle \theta_k, Y_s \rangle, n_k(s))}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))}
\end{aligned} \tag{A.7}$$

The bias of the remaining potential estimator:

$$\mathbb{E}[R_k(t)] - \mathbb{E}[G_k(t)] = \mathbb{E}[R_k(t)] - \frac{\mathbb{E}[R_k(t)]}{n_k(t)} \frac{\alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t))}{\alpha(\langle \theta_k, Y_t \rangle, n_k(t))} \sum_{s=1}^{t-1} \frac{\alpha(\langle \theta_k, Y_s \rangle, n_k(s))}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))} \tag{A.8}$$

$$\begin{aligned}
\mathbb{E}[R_k(t)] - \mathbb{E}[G_k(t)] &= \mathbb{E}[R_k(t)] \left(1 - \frac{1}{n_k(t)} e^{\frac{\langle \hat{\theta}_k(t) - \theta_k, Y_t \rangle + \gamma \|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{\langle \theta_k - \hat{\theta}_k(s), Y_s \rangle - \gamma \|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}} \right) \\
&= \mathbb{E}[R_k(t)] \left(1 - \frac{1}{n_k(t)} e^{\frac{\langle \hat{\theta}_k(t) - \theta_k, Y_t \rangle + \gamma \|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{\langle \theta_k - \hat{\theta}_k(s), Y_s \rangle - \gamma \|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}} \right)
\end{aligned} \tag{A.9}$$

$$\|Y_t\| \leq 1$$

$$\|\theta_k\| \leq 1, \|\hat{\theta}_k(t)\| \leq 1$$

Cauchy-Schwarz:

$$|\langle Y_t, \theta_k \rangle| \leq \|Y_t\| \|\theta_k\| \leq 1$$

$$|\langle Y_t, \hat{\theta}_k(t) \rangle| \leq \|Y_t\| \|\hat{\theta}_k(t)\| \leq 1$$

$$e^{\frac{-2}{n_k(t)}} \leq e^{\frac{\langle Y_t, \theta_k - \hat{\theta}_k(t) \rangle}{n_k(t)}} \leq e^{\frac{2}{n_k(t)}}$$

$$\begin{aligned}
1 - \frac{e^{\frac{2+\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}}}{n_k(t)} \sum_{s=1}^{t-1} e^{\frac{2-\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}} &\leq 1 - \frac{1}{n_k(t)} e^{\frac{\langle Y_t, \theta_k - \hat{\theta}_k(t) \rangle + \gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{\langle Y_s, \theta_k - \hat{\theta}_k(s) \rangle - \gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}} \\
&\leq 1 - \frac{1}{n_k(t)} e^{\frac{-2+\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{-2-\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}
\end{aligned} \tag{A.10}$$

Denote: $\mathbb{E}[R_k(t)] = \lambda_k$.

$$\lambda_k \left(1 - \frac{1}{n_k(t)} e^{\frac{2+\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{2-\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}\right) \leq \mathbb{E}[R_k(t)] - \mathbb{E}[G_k(t)]$$
$$\leq \lambda_k \left(1 - \frac{1}{n_k(t)} e^{\frac{-2+\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{-2-\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}\right) \quad \text{(A.11)}$$

The concentration of the remaining potential:

Theorem A.2.1. *Concentration of the remaining potential with external factors:*

$$\mathbb{E}\left[e^{s(R_k(t)-\mathbb{E}[R_k(t)]}\right] \leq \exp\left(\frac{s^2\lambda_k\alpha^2(\langle\theta_k, Y_t\rangle, n_k(t))}{4\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}\right) \quad \text{(A.12)}$$

Proof.

$$\mathbb{E}\left[e^{s(R_k(t)-\mathbb{E}[R_k(t)]}\right] =$$
$$\Pi_{j=1}^{A_k}\left(1 - e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}\right) e^{-s\lambda_j\alpha(\langle\theta_k, Y_t\rangle, n_k(t)) e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}} \quad \text{(A.13)}$$
$$+ e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))} e^{s\lambda_j\alpha(\langle\theta_k, Y_t\rangle, n_k(t))\left(1-e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}\right)}$$

By denoting, $p_j = e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}$ and $t_j = s\lambda_j\alpha(\langle\theta_k, Y_t\rangle, n_k(t))$, we have,

$$\begin{aligned}
\mathbb{E}\left[e^{s(R_k(t)-\mathbb{E}[R_k(t)]}\right] &= \Pi_{j=1}^{A_k}(1-p_j)e^{-p_jt_j} + p_je^{t_j(1-p_j)} \\
&\leq \Pi_{j=1}^{A_k} exp\left(\frac{1-2p_j}{4\ln((1-p_j)/p_j)}t_j^2\right) [\text{Theorem 3.2Berend et al. (2013)}] \\
&\leq \Pi_{j=1}^{A_k} exp\left(\frac{1}{4\ln(1/p_j)}t_j^2\right) [\textbf{Lemma 3.5}Berendet\ al.\ (2013)] \\
&\leq \Pi_{j=1}^{A_k} exp\left(\frac{s^2\lambda_j\alpha^2(\langle\theta_k, Y_t\rangle, n_k(t))}{4\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}\right) \\
&\leq \exp\left(\frac{\alpha^2(\langle\theta_k, Y_t\rangle, n_k(t))}{4\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}s^2\lambda_k\right)
\end{aligned} \quad \text{(A.14)}$$

Using Chernoff we finally have,

$$P(R_k(t) > \mathbb{E}[R_k(t)] + \sqrt{\frac{\alpha^2(\langle\theta_k, Y_t\rangle, n_k(t))\lambda_k \ln(1/\delta)}{\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}}) \leq \delta \tag{A.15}$$

□

The concentration of the estimator of the remaining potential:

$$G_k(t) = \frac{1}{n_k(t)}\sum_{j=1}^{A_k} U_t^\alpha(j,k) \tag{A.16}$$

For overlapping influencer supports, is the independence of the reward random variables preserved if L is constant per campaign and the reward is obtained by randomly dividing the feedback to each chosen influencer?

$$\begin{aligned} U_t^\alpha(j) &= \sum_{s=1}^{t-1}\frac{\alpha(\langle\hat{\theta}_k(t), Y_t\rangle, n_k(t))}{\alpha(\langle\hat{\theta}_k(s), Y_s\rangle, n_k(s))}\mathbb{I}\{X_{s,j,k}=1, \{X_{l,j,k'}=0\}_{k'\in I_s\setminus\{k\}}, \{X_{l,j,k'}=0\}_{l\neq s, k'\in I_l}\} \\ &= \sum_{s=1}^{t-1} e^{\frac{\langle\hat{\theta}_k(t), Y_t\rangle + \gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)} - \frac{\langle\hat{\theta}_k(s), Y_s\rangle + \gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}} \\ &\cdot \mathbb{I}\{X_{s,j,k}=1, \{X_{l,j,k'}=0\}_{k'\in I_s\setminus\{k\}}, \{X_{l,j,k'}=0\}_{l\neq s, k'\in I_l}\} \end{aligned} \tag{A.17}$$

$$\begin{aligned} G_k(t) &= \sum_{s=1}^{t-1}\frac{\mathbb{I}\left\{X_{s,j,k}=1, \{X_{l,j,k'}=0\}_{l\neq s,k'\in I_l}, \{X_{s,j,k'}=0\}_{k'\in I_s\setminus\{k\}}\right\}}{n_k(t)} \\ &\cdot e^{\frac{\langle\hat{\theta}_k(t), Y_t\rangle + \gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)} - \frac{\langle\hat{\theta}_k(s), Y_s\rangle + \gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}} \end{aligned} \tag{A.18}$$

For Bennett's inequality:

Denote $X_t(j) = \frac{U_t^\alpha(j)}{n_k(t)}$.

Denote $X_t(j,k) = \frac{U_t^\alpha(j,k)}{n_k(t)}$.

$$X_t(j) = \sum_{s=1}^{t-1} e^{\frac{\langle \hat{\theta}_k(t), Y_t \rangle + \gamma \|Y_t\|_{V_k^{-1}(t)}}{n_k(t)} - \frac{\langle \hat{\theta}_k(s), Y_s \rangle + \gamma \|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}} \cdot \frac{\mathbb{I}\left\{ X_{s,j,k} = 1, \{X_{l,j,k'} = 0\}_{l \neq s, k' \in I_l}, \{X_{s,j,k'} = 0\}_{k' \in I_s \setminus \{k\}} \right\}}{n_k(t)} \quad \text{(A.19)}$$
$$\leq \frac{1}{n_k(t)} e^{\frac{1+\gamma \|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{1+\gamma \|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}$$

$$X_t(j,k) \leq \frac{1}{n_k(t)} e^{\frac{1+\gamma \|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{1+\gamma \|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}} \quad \text{(A.20)}$$

$$S = \sum_{j=1}^{A_k} (X_t(j) - \mathbb{E}[X_t(j)]) \quad \text{(A.21)}$$

$$v = \sum_{j=1}^{A_k} \mathbb{E}[X_t^2(j)] = \sum_{j=1}^{A_k} \mathbb{E}\left[\frac{U_t^\alpha(j)^2}{n_k(t)^2} \right] = \frac{\sum_{j=1}^{A_k} \mathbb{E}[U_t^\alpha(j)^2]}{n_k^2(t)}$$
$$= \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{1^2 P(C_j(t-1)=1)}{n_k^2(t)} e^{\frac{2\langle \hat{\theta}_k(t), Y_t \rangle + 2\langle \gamma \|Y_t\|_{V_k^{-1}(t)}, Y_t \rangle}{n_k(t)} - \frac{2\langle \hat{\theta}_k(s), Y_s \rangle + 2\langle \gamma \|Y_s\|_{V_k^{-1}(s)}, Y_s \rangle}{n_k(s)}} \quad \text{(A.22)}$$
$$= \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{\lambda_j}{n_k^2(t)} e^{-\lambda_j \sum_{l=1}^{t-1} \sum_{k' \in I_l} e^{\frac{\langle \theta_{k'}, Y_l \rangle}{n_{k'}(l)}}} e^{\frac{\langle \theta_k, Y_t \rangle}{n_k(t)} + \frac{2\langle \hat{\theta}_k(t), Y_t \rangle + 2\gamma \|Y_t\|_{V_k^{-1}(t)}}{n_k(t)} + \frac{-2\langle \hat{\theta}_k(s), Y_s \rangle - 2\gamma \|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}$$

$$v \leq \frac{e^{\frac{3+2\gamma \|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{2-2\gamma \|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}}{n_k^2(t)} \sum_{j=1}^{A_k} \lambda_j \quad \text{(A.23)}$$

$$\sum_{j=1}^{A_k} \lambda_j \leq \lambda_k << A_k \quad \text{(A.24)}$$

$$v \leq \frac{e^{\frac{3+2\gamma \|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{2-2\gamma \|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}}{n_k^2(t)} \lambda_k \quad \text{(A.25)}$$

$$P(G_k(t) - \mathbb{E}[G_k(t)] > \sqrt{\frac{2\lambda_k e^{\frac{3+2\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{2-2\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}}{n_k^2(t)} \ln\frac{1}{\delta}} + \frac{e^{\frac{1+\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{1+\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}}{3n_k(t)} \ln\frac{1}{\delta}) \leq \delta \quad \text{(A.26)}$$

The confidence interval:

$$\begin{aligned} R_k(t) - G_k(t) \leq & \sqrt{\frac{\alpha^2(\langle\theta_k, Y_t\rangle, n_k(t))\lambda_k \ln(1/\delta)}{\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}} \\ & + \sqrt{\frac{2\lambda_k e^{\frac{3+2\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{2-2\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}}{n_k^2(t)} \ln\frac{1}{\delta}} \\ & + \frac{e^{\frac{1+\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{1+\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}}{3n_k(t)} \ln\frac{1}{\delta} \\ & + \lambda_k \left(1 - e^{\frac{-2+\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \frac{1}{n_k(t)} \sum_{s=1}^{t-1} e^{\frac{-2-\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}\right). \end{aligned} \quad \text{(A.27)}$$

$$-\beta_k(t) + \Omega\left(\frac{T\lambda_k(T)}{n_k(T)} e^{\frac{\gamma\|Y_T\|_{V_k^{-1}(T)}}{n_k(T)}}\right) \leq R_k(t) - G_k(t) \leq \beta_k(t) + \mathcal{O}\left(T\frac{\lambda_k(T)}{n_k(T)} e^{\frac{\gamma\|Y_T\|_{V_k^{-1}(T)}}{n_k(T)}}\right), \quad \text{(A.28)}$$

where:

$$\begin{aligned} \beta_k(t) = & \sqrt{\frac{2\lambda_k(t) e^{\frac{3+2\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{2-2\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}}{n_k^2(t)} \ln\frac{1}{\delta}} \\ & + \sqrt{\frac{e^{2/n_k(t)}\lambda_k(t)\ln(1/\delta)}{\sum_{s=1}^{t-1}\sum_{k'\in I_s} e^{-1/n_{k'}(s)}}} + \frac{e^{\frac{1+\gamma\|Y_t\|_{V_k^{-1}(t)}}{n_k(t)}} \sum_{s=1}^{t-1} e^{\frac{1+\gamma\|Y_s\|_{V_k^{-1}(s)}}{n_k(s)}}}{3n_k(t)} \ln\frac{1}{\delta} \end{aligned} \quad \text{(A.29)}$$

□

Table A.2: Data statistics.

Net.	# Users	#Edges	Net.	#Orig.-microblogs	#Retweets
Weibo	1,776,950	308M	Weibo	300,000	23,755,810
Twitter	11.6M	309M	Twitter	242M	341,811,085

A.3 Twitter dataset statistics

	Random tweets	Random campaign
count	705586	705586
mean	0.6	3.5
std	6.68	16.37
min	0	0
25%	0	1
50%	0	2
75%	1	4
max	4581	10676

Table A.3: The rewards distribution in the datasets.

The statistics for both types of simulation are presented in Table A.3.

A.4 `GLM-GT-UCB with MLE` - Theoretical analysis

In this Section, we present a variant of `GLM-GT-UCB` , `GLM-GT-UCB with MLE` presented in Algorithm 6, which assumes the *generalized linear model* for the external factor with the exponential inverse link function, and uses Maximum Likelihood Estimation to learn it relative to the historical rewards within their contexts. The difference between the two algorithms is that in the latter, the unknown influencer parameter θ_k is estimated with MLE instead of RLSE.

Theorem A.4.1. *With probability at least* $1-\delta$*, having the expected activations* $\lambda_k = \sum_{j=1}^{A_k} p_{k,j}(t)$*,* $\beta_k(t) = \sqrt{\frac{\alpha^2(\langle\theta_k, Y_t\rangle, n_k(t))\lambda_k \ln(1/\delta)}{\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}} + \sqrt{\frac{2e^5 f(n_k(t))\lambda_k \sum_{s=1}^{t-1}\frac{1}{f(n_k(s))}}{n_k^2(t)}\ln\frac{1}{\delta}} + \frac{e^2 f(n_k(t))\sum_{s=1}^{t-1}\frac{1}{f(n_k(s))}}{3n_k(t)}\ln\frac{1}{\delta}$*, and* $\rho(t) = \sqrt{\frac{2k_\mu}{c_\mu}\log(t)}$*, with the Lipschitz constant* k_μ *and* $c_\mu = \inf_{\theta\in\Theta}\dot{\mu}(\langle\theta, Y\rangle) > 0$ *for the link func-*

Algorithm 6 GLM-GT-UCB with MLE

Input: influencers $\mathcal{K}$, rounds budget T, external factor function α, regularization factor γ, fatigue function f, number of selections L per round.
Initialization: play each influencer $k \in \mathcal{K}$ once in given random contexts Y_t, observe the reward $r_t, t \in \mathcal{K}$, and update the statistics $n_k(1) = 1$ for the Good-Turing estimator, and $V_k(1) = \gamma I_d + Y_t Y_t^T, s_k(1) = Y_t r_1$
for $t = K+1, \ldots, T$ **do**
    Get the context Y_t.
    for $k \in \mathcal{K}$ **do**
        Estimate the unknown parameter in the GLM by solving:

$$\sum_{s=1}^{n_k(t)-1} (r_s - e^{\langle \hat{\theta}_k(t), Y_s \rangle}) Y_s = 0 \tag{A.30}$$

        Compute UCB for remaining potential estimator

$$b_k(t) = G_k(t) + \beta_k(t), \tag{A.31}$$

$$G_k(t) = \alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t)) \frac{1}{n_k(t)} \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{\text{hapax}_{s,j,k}}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))} \tag{A.32}$$

        and $\beta_k(t)$ is given by the confidence interval, and $\text{hapax}_{s,j,k}$ by Equation 3.14.
    end for
    Choose set I_t of L influencers with largest UCB.
    Play the chosen influencers, observe spread, divide it equally among influencers, and update their statistics:
    for $k' \in I_t$ **do**
        Update $r_{k'}(t)$ by Eq. (3.13).; $n_{k'}(t+1) = n_{k'}(t) + 1; V_{k'}(t+1) = V_{k'}(t) + Y_t Y_t^T$.
    end for
end for

tion $\mu : \mathbb{R} \to \mathbb{R}$ for the Poisson regression model $\mu(x) = e^x$, we have

$$-\beta_k(t) + \lambda_k \left(1 - \frac{\rho_k(t)\|Y_t\|_{V_k^{-1}(t)} + 1}{n_k(t)} \sum_{s=1}^{t-1} (\rho_k(s)\|Y_s\|_{V_k^{-1}(s)} + 1)\right) \leq$$
$$R_k(t) - G_k(t) \leq \beta_k(t) + \lambda_k \left(1 - \frac{1}{n_k(t)(\rho_k(t)\|Y_t\|_{V_k^{-1}(t)} + 1)} \sum_{s=1}^{t-1} \frac{1}{\rho_k(s)\|Y_s\|_{V_k^{-1}(s)} + 1}\right), \tag{A.33}$$

Proof. The theoretical analysis is performed for one influencer k; it follows the same steps for any other influencer.

Preliminaries:

Table A.4: Assumptions

$\theta_{k,j} \approx \theta_k$ (the influencer's unknown vector θ_k captures the basic node's trend in activating in a context along its inner probability $p(j)$)
$\alpha(\langle \theta_k, Y_t \rangle, n_k(t)) = e^{\langle \theta_k, Y_t \rangle} f(n_k(t))$
e.g. $f(n_k(t)) = \frac{1}{\sqrt{n_k(t)}}$
$\|\theta_k\| \leq 1$
$\|Y_t\| \leq 1$

$$p_{k,j}(t) = \alpha(\langle \theta_k, Y_t \rangle, n_k(t)) p(j) \tag{A.34}$$

$$P(C_j(t-1) = 0) = e^{-\lambda_j \sum_{s=1}^{t-1} \sum_{k' \in I_s} \alpha(\langle \theta_{k'}, Y_s \rangle, n_{k'}(s))} \tag{A.35}$$

$$P(C_{k,j}(t-1) = 1) = \sum_{s=1}^{t-1} \lambda_j \alpha(\langle \theta_k, Y_s \rangle, n_k(s)) e^{-\lambda_j \sum_{l=1}^{t-1} \sum_{k' \in I_l} \alpha(\langle \theta_{k'}, Y_l \rangle, n_{k'}(l))} \tag{A.36}$$

$$R_k(t) = \alpha(\langle \theta_k, Y_t \rangle, n_k(t)) \sum_{j=1}^{A_k} \lambda_j \mathbb{I}\{C_j(t-1) = 0\} \tag{A.37}$$

$$\mathbb{E}[R_k(t)] = \alpha(\langle \theta_k, Y_t \rangle, n_k(t)) \sum_{j=1}^{A_k} \lambda_j e^{-\lambda_j \sum_{s=1}^{t-1} \sum_{k' \in I_s} \alpha(\langle \theta_{k'}, Y_s \rangle, n_{k'}(s))} \tag{A.38}$$

$$G_k(t) = \alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t)) \frac{1}{n_k(t)} \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{\mathbb{I}\{X_{s,j,k} = 1, \{X_{l,j,k'} = 0\}_{k' \in I_s \setminus \{k\}}, \{X_{l,j,k'} = 0\}_{l \neq s, k' \in I_l}\}}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))} \tag{A.39}$$

$$\mathbb{E}[G_k(t)] = \alpha(\langle\hat{\theta}_k(t), Y_t\rangle, n_k(t))\frac{1}{n_k(t)}\sum_{j=1}^{A_k}\sum_{s=1}^{t-1}\frac{\alpha(\langle\theta_k, Y_s\rangle, n_k(s))\lambda_j e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}}{\alpha(\langle\hat{\theta}_k(s), Y_s\rangle, n_k(s))}$$
$$= \frac{\alpha(\langle\hat{\theta}_k(t), Y_t\rangle, n_k(t))}{\alpha(\langle\theta_k, Y_t\rangle, n_k(t))n_k(t)}\sum_{j=1}^{A_k}\alpha(\langle\theta_k, Y_t\rangle, n_k(t))\lambda_j e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}\sum_{s=1}^{t-1}\frac{\alpha(\langle\theta_k, Y_s\rangle, n_k(s))}{\alpha(\langle\hat{\theta}_k(s), Y_s\rangle, n_k(s))}$$
$$= \frac{\alpha(\langle\hat{\theta}_k(t), Y_t\rangle, n_k(t))}{\alpha(\langle\theta_k, Y_t\rangle, n_k(t))n_k(t)}\sum_{s=1}^{t-1}\frac{\alpha(\langle\theta_k, Y_s\rangle, n_k(s))}{\alpha(\langle\hat{\theta}_k(s), Y_s\rangle, n_k(s))}\alpha(\langle\theta_k, Y_t\rangle, n_k(t))\sum_{j=1}^{A_k}\lambda_j e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}$$
$$= \frac{\alpha(\langle\hat{\theta}_k(t), Y_t\rangle, n_k(t))}{\alpha(\langle\theta_k, Y_t\rangle, n_k(t))}\frac{\mathbb{E}[R_k(t)]}{n_k(t)}\sum_{s=1}^{t-1}\frac{\alpha(\langle\theta_k, Y_s\rangle, n_k(s))}{\alpha(\langle\hat{\theta}_k(s), Y_s\rangle, n_k(s))} \tag{A.40}$$

The bias of the remaining potential estimator:

$$\mathbb{E}[R_k(t)] - \mathbb{E}[G_k(t)] = \mathbb{E}[R_k(t)] - \frac{\mathbb{E}[R_k(t)]}{n_k(t)}\frac{\alpha(\langle\hat{\theta}_k(t), Y_t\rangle, n_k(t))}{\alpha(\langle\theta_k, Y_t\rangle, n_k(t))}\sum_{s=1}^{t-1}\frac{\alpha(\langle\theta_k, Y_s\rangle, n_k(s))}{\alpha(\langle\hat{\theta}_k(s), Y_s\rangle, n_k(s))} \tag{A.41}$$

$$\mathbb{E}[R_k(t)] - \mathbb{E}[G_k(t)] = \mathbb{E}[R_k(t)]\left(1 - \frac{e^{\langle\hat{\theta}_k(t)-\theta_k, Y_t\rangle}}{n_k(t)}\sum_{s=1}^{t-1}e^{\langle\theta_k-\hat{\theta}_k(s), Y_s\rangle}\right) \tag{A.42}$$

$$\|Y_t\| \leq 1 \tag{A.43}$$

$$\|\theta_k\| \leq 1, \|\hat{\theta}_k(t)\| \leq 1 \tag{A.44}$$

Cauchy-Schwarz:

$$|\langle Y_t, \theta_k\rangle| \leq \|Y_t\|\|\theta_k\| \leq 1 \tag{A.45}$$

$$|\langle Y_t, \hat{\theta}_k(t)\rangle| \leq \|Y_t\|\|\hat{\theta}_k(t)\| \leq 1 \tag{A.46}$$

From Filippi et al. (2010b):

$$|e^{\langle\theta_k,Y_t\rangle} - e^{\langle\hat{\theta}_k(t),Y_t\rangle}| \leq \rho_k(t)\|Y_t\|_{V_k^{-1}(t)}, \tag{A.47}$$

where $\rho(t) = \sqrt{\frac{2k_\mu}{c_\mu}\log(t)}$, with the Lipschitz constant k_μ and $c_\mu = \inf_{\theta\in\Theta}\dot{\mu}(\langle\theta, Y\rangle) > 0$ for the link function $\mu : \mathbb{R} \to \mathbb{R}$ for the Poisson regression model $\mu(x) = e^x$.

$$e^{|\langle\theta_k-\hat{\theta}_k(t),Y_t\rangle|} - 1 \leq |e^{\langle\theta_k,Y_t\rangle} - e^{\langle\hat{\theta}_k(t),Y_t\rangle}| \leq \rho_k(t)\|Y_t\|_{V_k^{-1}(t)} \tag{A.48}$$

$$e^{|\langle\theta_k-\hat{\theta}_k(t),Y_t\rangle|} \leq \rho_k(t)\|Y_t\|_{V_k^{-1}(t)} + 1 \tag{A.49}$$

Denote: $\mathbb{E}[R_k(t)] = \lambda_k$.

$$\begin{aligned}\lambda_k\left(1 - \frac{\rho_k(t)\|Y_t\|_{V_k^{-1}(t)} + 1}{n_k(t)}\sum_{s=1}^{t-1}(\rho_k(s)\|Y_s\|_{V_k^{-1}(s)} + 1)\right) &\leq \mathbb{E}[R_k(t)] - \mathbb{E}[G_k(t)] \\ &\leq \lambda_k\left(1 - \frac{1}{n_k(t)(\rho_k(t)\|Y_t\|_{V_k^{-1}(t)} + 1)}\sum_{s=1}^{t-1}\frac{1}{\rho_k(s)\|Y_s\|_{V_k^{-1}(s)} + 1}\right)\end{aligned} \tag{A.50}$$

The concentration of the remaining potential:

Theorem A.4.2. *The concentration of the remaining potential with external factors:*

$$\mathbb{E}\left[e^{s(R_k(t)-\mathbb{E}[R_k(t)]}\right] \leq \exp\left(\frac{s^2\lambda_k\alpha^2(\langle\theta_k, Y_t\rangle, n_k(t))}{4\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}\right) \tag{A.51}$$

Proof.

$$\begin{aligned}&\mathbb{E}\left[e^{s(R_k(t)-\mathbb{E}[R_k(t)]}\right] = \\ &\Pi_{j=1}^{A_k}\left(1 - e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'},Y_l\rangle,n_{k'}(l))}\right)e^{-s\lambda_j\alpha(\langle\theta_k,Y_t\rangle,n_k(t))e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'},Y_l\rangle,n_{k'}(l))}} \\ &+ e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'},Y_l\rangle,n_{k'}(l))}e^{s\lambda_j\alpha(\langle\theta_k,Y_t\rangle,n_k(t))\left(1-e^{-\lambda_j\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'},Y_l\rangle,n_{k'}(l))}\right)}\end{aligned} \tag{A.52}$$

By denoting, $p_j = e^{-\lambda_j \sum_{l=1}^{t-1} \sum_{k' \in I_l} \alpha(\langle \theta_{k'}, Y_l \rangle, n_{k'}(l))}$ and $t_j = s\lambda_j \alpha(\langle \theta_k, Y_t \rangle, n_k(t))$, we have,

$$\begin{aligned}
\mathbb{E}\left[e^{s(R_k(t)-\mathbb{E}[R_k(t)]}\right] &= \Pi_{j=1}^{A_k}(1-p_j)e^{-p_j t_j} + p_j e^{t_j(1-p_j)} \\
&\leq \Pi_{j=1}^{A_k} exp\left(\frac{1-2p_j}{4\ln((1-p_j)/p_j)} t_j^2\right) [\textbf{Theorem 3.2} Berendet\ al.\ (2013)] \\
&\leq \Pi_{j=1}^{A_k} exp\left(\frac{1}{4\ln(1/p_j)} t_j^2\right) [\textbf{Lemma 3.5} Berendet\ al.\ (2013)] \\
&\leq \Pi_{j=1}^{A_k} exp\left(\frac{s^2\lambda_j \alpha^2(\langle \theta_k, Y_t \rangle, n_k(t))}{4\sum_{l=1}^{t-1}\sum_{k' \in I_l} \alpha(\langle \theta_{k'}, Y_l \rangle, n_{k'}(l))}\right) \\
&\leq \exp\left(\frac{\alpha^2(\langle \theta_k, Y_t \rangle, n_k(t))}{4\sum_{l=1}^{t-1}\sum_{k' \in I_l} \alpha(\langle \theta_{k'}, Y_l \rangle, n_{k'}(l))} s^2 \lambda_k\right)
\end{aligned} \tag{A.53}$$

Using Chernoff we finally have,

$$P(R_k(t) > \mathbb{E}[R_k(t)] + \sqrt{\frac{\alpha^2(\langle \theta_k, Y_t \rangle, n_k(t))\lambda_k \ln(1/\delta)}{\sum_{l=1}^{t-1}\sum_{k' \in I_l} \alpha(\langle \theta_{k'}, Y_l \rangle, n_{k'}(l))}}) \leq \delta \tag{A.54}$$

□

The concentration of the estimator of the remaining potential:

$$G_k(t) = \frac{1}{n_k(t)} \sum_{j=1}^{A_k} U_t^\alpha(j,k) \tag{A.55}$$

$$\begin{aligned}
U_t^\alpha(j) &= \sum_{s=1}^{t-1} \frac{\alpha(\langle \hat{\theta}_k(t), Y_t \rangle, n_k(t))}{\alpha(\langle \hat{\theta}_k(s), Y_s \rangle, n_k(s))} \mathbb{I}\{X_{s,j,k} = 1, \{X_{l,j,k'} = 0\}_{k' \in I_s \setminus \{k\}}, \{X_{l,j,k'} = 0\}_{l \neq s, k' \in I_l}\} \\
&= \sum_{s=1}^{t-1} \frac{f(n_k(s))}{f(n_k(t))} e^{\langle \hat{\theta}_k(t), Y_t \rangle - \langle \hat{\theta}_k(s), Y_s \rangle} \mathbb{I}\{X_{s,j,k} = 1, \{X_{l,j,k'} = 0\}_{k' \in I_s \setminus \{k\}}, \{X_{l,j,k'} = 0\}_{l \neq s, k' \in I_l}\}
\end{aligned} \tag{A.56}$$

$$G_k(t) = \frac{f(n_k(t))}{n_k(t)} \sum_{s=1}^{t-1} \mathbb{I}\left\{X_{s,j,k} = 1, \{X_{l,j,k'} = 0\}_{l \neq s, k' \in I_l}, \{X_{s,j,k'} = 0\}_{k' \in I_s \setminus \{k\}}\right\} \frac{e^{\langle \hat{\theta}_k(t), Y_t \rangle - \langle \hat{\theta}_k(s), Y_s \rangle}}{f(n_k(s))} \tag{A.57}$$

For Bennett's inequality:

Denote i.i.d r.v. $X_t(j) = \frac{U_t^\alpha(j)}{n_k(t)}, \forall j \in A_k$.

$$\begin{aligned} X_t(j) &= \sum_{s=1}^{t-1} \frac{f(n_k(t)}{n_k(s)} e^{\langle \hat{\theta}_k(t), Y_t \rangle - \langle \hat{\theta}_k(s), Y_s \rangle} \frac{\mathbb{I}\left\{X_{s,j,k} = 1, \{X_{l,j,k'} = 0\}_{l \neq s, k' \in I_l}, \{X_{s,j,k'} = 0\}_{k' \in I_s \setminus \{k\}}\right\}}{n_k(t)} \\ &\leq \frac{e^2 f(n_k(t))}{n_k(t)} \sum_{s=1}^{t-1} \frac{1}{f(n_k(s))} \end{aligned} \tag{A.58}$$

$$\begin{aligned} v &= \sum_{j=1}^{A_k} \mathbb{E}[X_t^2(j)] \\ &= \sum_{j=1}^{A_k} \mathbb{E}\left[\frac{U_t^\alpha(j)^2}{n_k(t)^2}\right] \\ &= \frac{\sum_{j=1}^{A_k} \mathbb{E}[U_t^\alpha(j)^2]}{n_k^2(t)} \\ &= \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{1^2 P(C_j(t-1) = 1)}{n_k^2(t)} \frac{f^2(n_k(t)}{f^2(n_k(s))} e^{2\langle \hat{\theta}_k(t), Y_t \rangle - 2\langle \hat{\theta}_k(s), Y_s \rangle} \\ &= \sum_{j=1}^{A_k} \sum_{s=1}^{t-1} \frac{\lambda_j f(n_k(t))}{n_k^2(t) f(n_k(s))} e^{-\lambda_j \sum_{l=1}^{t-1} \sum_{k' \in I_l} e^{\langle \theta_{k'}, Y_l \rangle} f(n_{k'}(l))} e^{\langle \theta_k, Y_s \rangle + 2\langle \hat{\theta}_k(t), Y_t \rangle - 2\langle \hat{\theta}_k(s), Y_s \rangle} \end{aligned} \tag{A.59}$$

$$v \leq \frac{e^5 f(n_k(t)) \sum_{j=1}^{A_k} \lambda_j \sum_{s=1}^{t-1} \frac{1}{f(n_k(s))}}{n_k^2(t)} \tag{A.60}$$

$$\sum_{j=1}^{A_k} \lambda_j \leq \lambda_k << A_k \tag{A.61}$$

$$v \leq \frac{e^5 f(n_k(t))\lambda_k \sum_{s=1}^{t-1} \frac{1}{f(n_k(s))}}{n_k^2(t)} \tag{A.62}$$

$$P\left(G_k(t) - \mathbb{E}[G_k(t)] > \sqrt{\frac{2e^5 f(n_k(t))\lambda_k \sum_{s=1}^{t-1} \frac{1}{f(n_k(s))}}{n_k^2(t)} \ln\frac{1}{\delta}} + \frac{e^2 f(n_k(t)) \sum_{s=1}^{t-1} \frac{1}{f(n_k(s))}}{3n_k(t)} \ln\frac{1}{\delta}\right) \leq \delta \tag{A.63}$$

The confidence interval:

$$R_k(t) - G_k(t) \leq \sqrt{\frac{\alpha^2(\langle\theta_k, Y_t\rangle, n_k(t))\lambda_k \ln(1/\delta)}{\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}} + \sqrt{\frac{2e^5 f(n_k(t))\lambda_k \sum_{s=1}^{t-1} \frac{1}{f(n_k(s))}}{n_k^2(t)} \ln\frac{1}{\delta}}$$
$$+ \frac{e^2 f(n_k(t)) \sum_{s=1}^{t-1} \frac{1}{f(n_k(s))}}{3n_k(t)} \ln\frac{1}{\delta} + \lambda_k\left(1 - \frac{1}{n_k(t)(\rho_k(t)\|Y_t\|_{V_k^{-1}(t)} + 1)} \sum_{s=1}^{t-1} \frac{1}{\rho_k(s)\|Y_s\|_{V_k^{-1}(s)} + 1}\right). \tag{A.64}$$

$$-\beta_k(t) + \lambda_k\left(1 - \frac{\rho_k(t)\|Y_t\|_{V_k^{-1}(t)} + 1}{n_k(t)} \sum_{s=1}^{t-1}(\rho_k(s)\|Y_s\|_{V_k^{-1}(s)} + 1)\right) \leq$$
$$R_k(t) - G_k(t) \leq \beta_k(t) + \lambda_k\left(1 - \frac{1}{n_k(t)(\rho_k(t)\|Y_t\|_{V_k^{-1}(t)} + 1)} \sum_{s=1}^{t-1} \frac{1}{\rho_k(s)\|Y_s\|_{V_k^{-1}(s)} + 1}\right), \tag{A.65}$$

where:

$$\beta_k(t) = \sqrt{\frac{\alpha^2(\langle\theta_k, Y_t\rangle, n_k(t))\lambda_k \ln(1/\delta)}{\sum_{l=1}^{t-1}\sum_{k'\in I_l}\alpha(\langle\theta_{k'}, Y_l\rangle, n_{k'}(l))}}$$
$$+ \sqrt{\frac{2e^5 f(n_k(t))\lambda_k \sum_{s=1}^{t-1} \frac{1}{f(n_k(s))}}{n_k^2(t)} \ln\frac{1}{\delta}} \tag{A.66}$$
$$+ \frac{e^2 f(n_k(t)) \sum_{s=1}^{t-1} \frac{1}{f(n_k(s))}}{3n_k(t)} \ln\frac{1}{\delta}$$

□

A.4.1 Experiments

For the sake of comparison, we run the experiments for `GLM-GT-UCB with MLE` with the same configuration as in section 3.5. `GLM-GT-UCB with MLE` does not seem to perform better than `GLM-GT-UCB` on the synthetic dataset (Figure A.2) or on the Twitter dataset (Figure A.3). However, it surpasses all the other solutions on the Sina Weibo dataset when multiple influencers are selected per round, see Figure A.4. This result indicates a high performance for this solution especially when the reward scale is greater than 20K and can be explained by the new confidence bound for the remaining potential estimators.

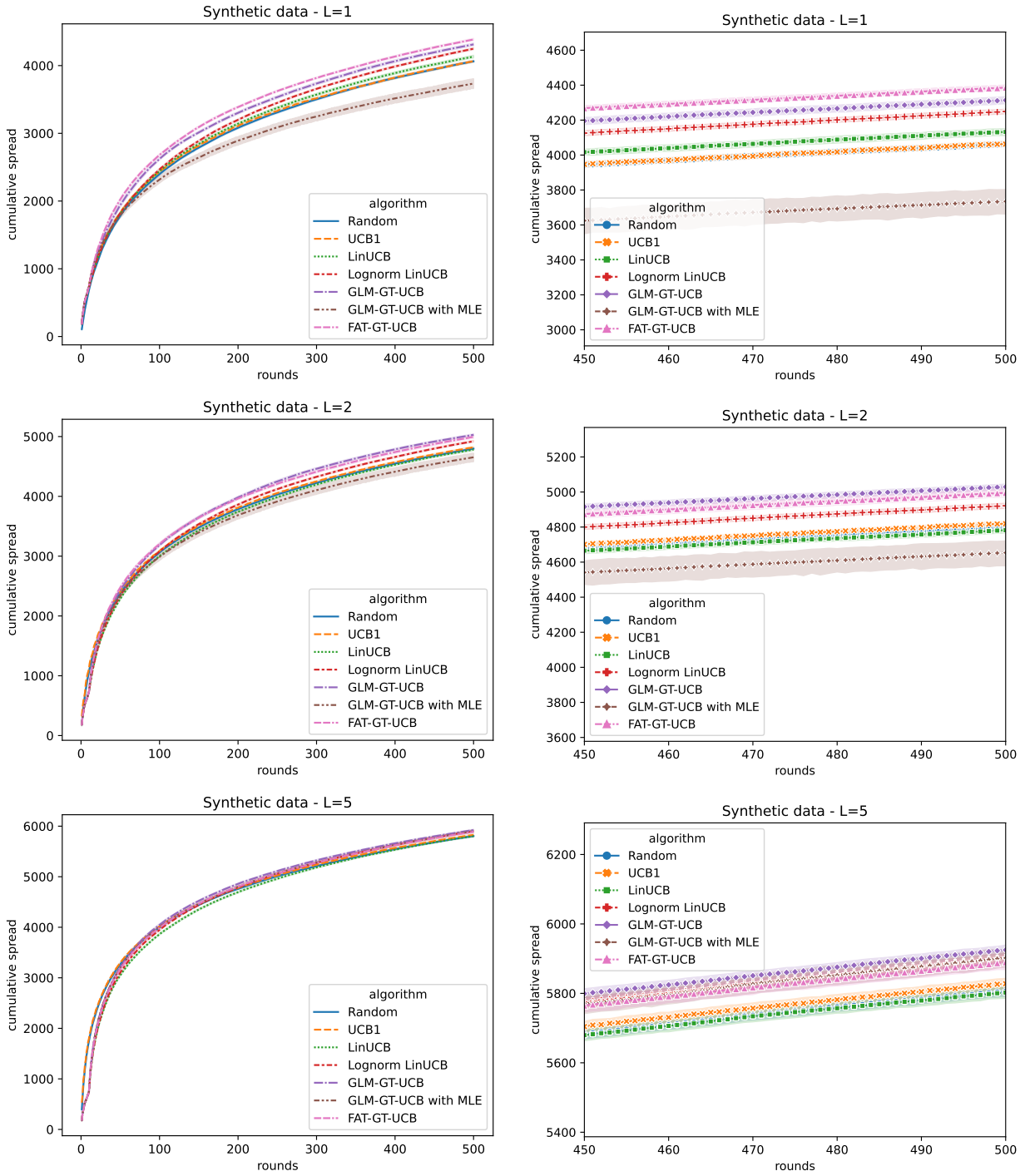

Figure A.2: Synthetic - Cumulative rewards, full plot (left column), plot zoomed to last 100 rounds (right column).

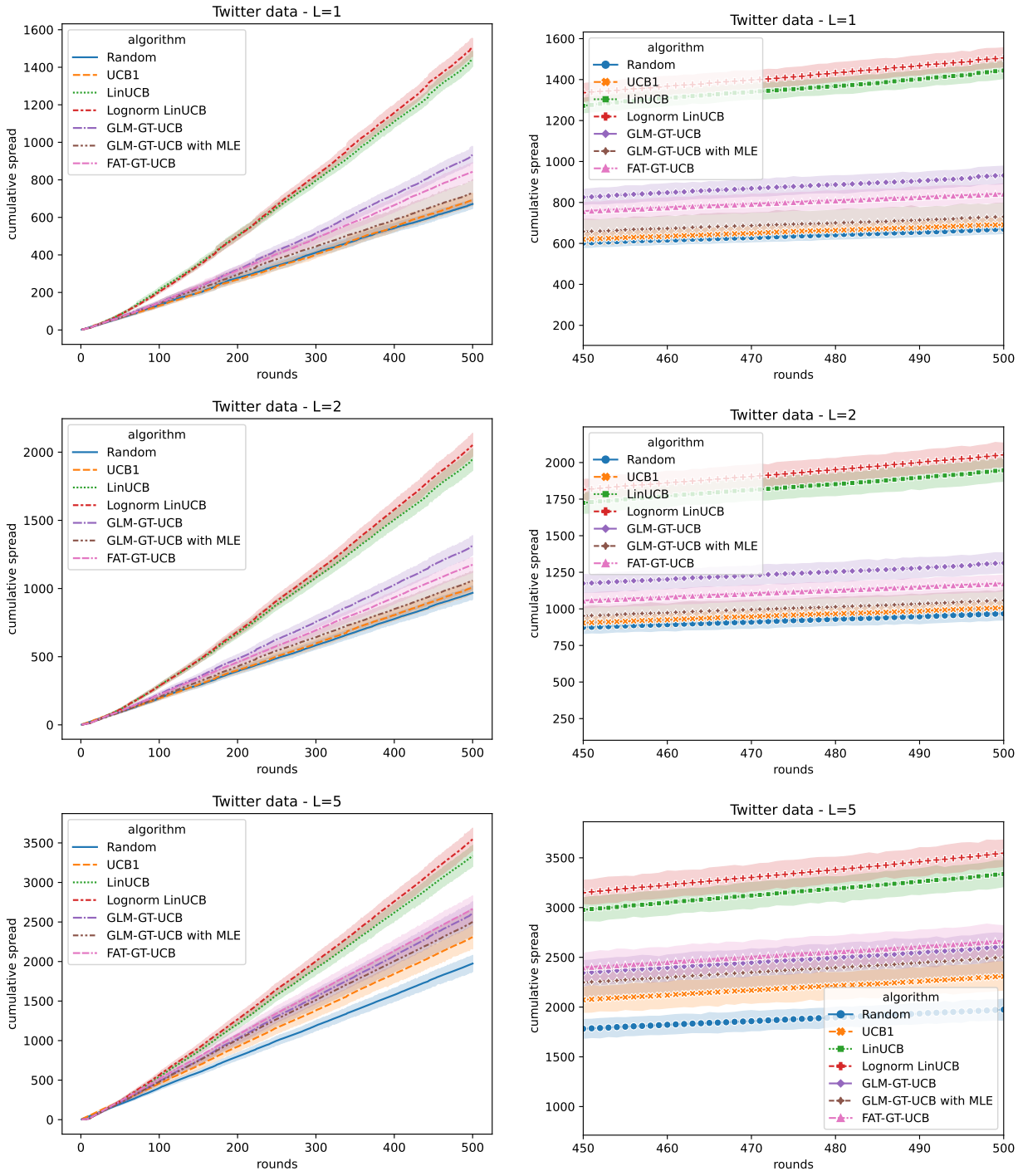

Figure A.3: Twitter - Cumulative rewards, full plot (left column), plot zoomed to last 100 rounds (right column).

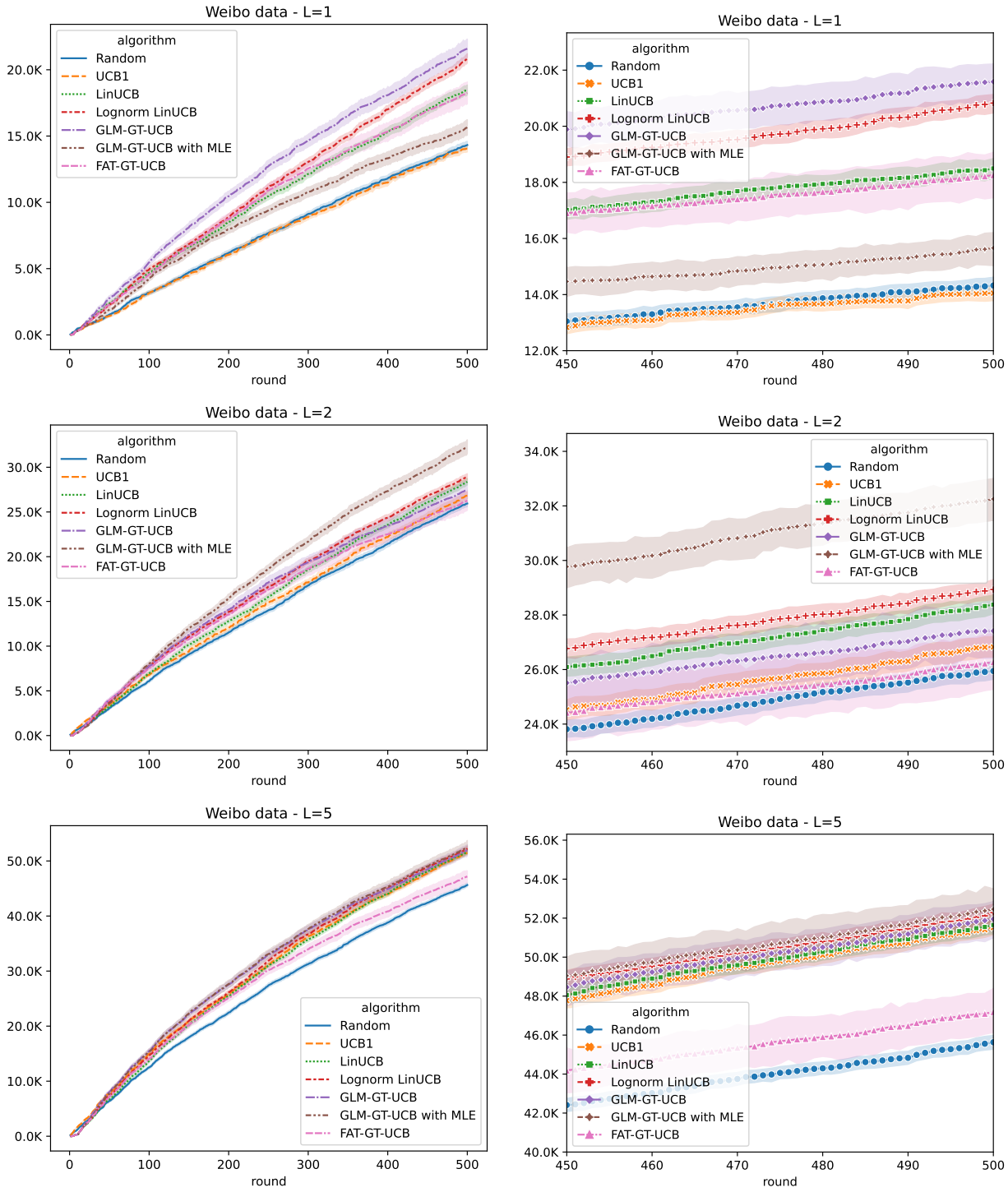

Figure A.4: Sina Weibo - Cumulative rewards, full plot (left column), plot zoomed to last 100 rounds (right column).

Bibliography

A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1638–1646, 2014.

A. Agarwal, H. Luo, B. Neyshabur, and R. E. Schapire. Corralling a band of bandit algorithms. In *Conference on Learning Theory*, pages 12–38. PMLR, 2017.

N. Alon, I. Gamzu, and M. Tennenholtz. Optimizing budget allocation among channels and influencers. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, pages 381–388, 2012. doi: 10.1145/2187836.2187888. URL https://doi.org/10.1145/2187836.2187888.

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

A. Arora, S. Galhotra, and S. Ranu. Debunking the myths of influence maximization: An in-depth benchmarking study. In *SIGMOD*, 2017.

A. Arora, S. Galhotra, and S. Ranu. Influence maximization revisited: The state of the art and the gaps that remain. In *EDBT*, 2019.

C. Aslay, N. Barbieri, F. Bonchi, and R. Baeza-Yates. Online topic-aware influence maximization queries. In *EDBT*, pages 295–306, 2014.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 2002.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 2002a.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002b.

A.-L. Barabási et al. *Network science*. Cambridge university press, 2016.

H. Bastani and M. Bayati. Online decision making with high-dimensional covariates. *Operations Research*, 68(1):276–294, 2020.

D. Berend, A. Kontorovich, et al. On the concentration of the missing mass. *ECP*, 2013.

D. A. Berry. Modified two-armed bandit strategies for certain clinical trials. *Journal of the American Statistical Association*, 73(362):339–345, 1978.

S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1):33–57, 1996.

D. Brown and S. Fiorella. *Influence Marketing: How to Create, Manage, and Measure Brand Influencers in Social Media Marketing*. Que Pub, 2013.

S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Found. Trends Mach. Learn.*, 2012.

S. Bubeck, D. Ernst, and A. Garivier. Optimal discovery with probabilistic expert advice: finite time analysis and macroscopic optimality. *JMLR*, 2013a.

S. Bubeck, D. Ernst, and A. Garivier. Optimal discovery with probabilistic expert advice: finite time analysis and macroscopic optimality. *Journal of Machine Learning Research*, 14, 2013b.

J. Buchin. McKinsey Quarterly – Getting a sharper picture of social media's influence, 2015. URL `https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/getting-a-sharper-picture-of-social-medias-influence#`.

N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu. Boltzmann exploration done right. *Advances in neural information processing systems*, 30, 2017.

S. Chen, J. Fan, G. Li, J. Feng, K.-l. Tan, and J. Tang. Online topic-aware influence maximization.

Proceedings of the VLDB Endowment, 8(6):666–677, 2015.

W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038, 2010.

W. Chen, Y. Wang, Y. Yuan, and Q. Wang. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *JMLR*, 2016.

W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. In *AISTATS'11*, 2011.

H. Dai, B. Dai, and L. Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711. PMLR, 2016.

P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66, 2001.

J. Dong, K. Li, S. Li, and B. Wang. Combinatorial bandits under strategic manipulations. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 219–229, 2022.

N. Du, L. Song, H. Woo, and H. Zha. Uncover topic-sensitive information diffusion networks. In *AISTATS'13*, 2013.

A. Durand, C. Achilleos, D. Iacovides, K. Strati, G. D. Mitsis, and J. Pineau. Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis. In *Machine learning for healthcare conference*, pages 67–82. PMLR, 2018.

R. Dye. The buzz on buzz. *Harvard business review*, 78(6):139–146, 2000.

M. Ewing. 71% More Likely to Purchase Based on Social Media Referrals, 2019. URL `https://blog.hubspot.com/blog/tabid/6307/bid/30239/71-More-Likely-to-Purchase-Based-on-Social-Media-Referrals-Infographic.aspx`.

S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári. Parametric bandits: The generalized linear

case. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*. 2010a.

S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári. Parametric bandits: The generalized linear case. In *NIPS*, 2010b.

M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *ACM TKDD'12*, 2012.

I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 1953.

J. Guo, P. Zhang, C. Zhou, Y. Cao, and L. Guo. Personalized influence maximization on social networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 199–208, 2013.

G. Heinrich. Parameter estimation for text analysis. Technical report, 2005.

M. Herbster, S. Pasteris, F. Vitale, and M. Pontil. A gang of adversarial bandits. *Advances in Neural Information Processing Systems*, 34:2265–2279, 2021.

S. Hu, B. Cautis, Z. Chen, L. Chan, Y. Geng, and X. He. Model-free inference of diffusion networks using RKHS embeddings. *Data Min. Knowl. Discov.*, 2019.

A. Iacob, B. Cautis, and S. Maniu. Contextual bandits for advertising campaigns: A diffusion-model independent approach. In *Proceedings of the 2022 SIAM International Conference on Data Mining, SDM 2022, Alexandria, VA, USA, April 28-30, 2022*, pages 513–521, 2022.

C. Jin, Z. Yang, Z. Wang, and M. I. Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory (CoLT)*, pages 2137–2143. PMLR, 2020.

R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social

network. In *KDD'03*, 2003.

E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

P. Lagrée, C. Vernade, and O. Cappe. Multiple-play bandits in the position-based model. *Advances in Neural Information Processing Systems*, 29, 2016.

P. Lagrée, O. Cappé, B. Cautis, and S. Maniu. Algorithms for online influencer marketing. *ACM TKDD'18*, 2018.

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

S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart. Online influence maximization. In *SIGKDD'15*, 2015a.

S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart. Online influence maximization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 645–654, 2015b.

N. Levine, K. Crammer, and S. Mannor. Rotting bandits. *arXiv preprint arXiv:1702.07274*, 2017.

H. Li, M. Xu, S. S. Bhowmick, C. Sun, Z. Jiang, and J. Cui. Disco: Influence maximization meets network embedding and deep learning. *arXiv preprint arXiv:1906.07378*, 2019.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW, pages = 661–670, numpages = 10,*, 2010a.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010b.

L. Li, Y. Lu, and D. Zhou. Provably optimal algorithms for generalized linear contextual bandits. In *ICML*, 2017.

S. Li, B. Wang, S. Zhang, and W. Chen. Contextual combinatorial cascading bandits. In *ICML*, pages 1245–1253. PMLR, 2016.

S. Li, F. Kong, K. Tang, Q. Li, and W. Chen. Online influence maximization under linear threshold model. *Advances in Neural Information Processing Systems*, 33:1192–1204, 2020.

Y. Li, J. Fan, Y. Wang, and K. Tan. Influence maximization on social graphs: A survey. *TKDE'18*, 2018.

J. Mary, R. Gaudel, and P. Preux. Bandits and recommender systems. In *International Workshop on Machine Learning, Optimization and Big Data*, pages 325–336. Springer, 2015.

D. A. McAllester and R. E. Schapire. On the convergence rate of good-turing estimators. In *COLT*, pages 1–6, 2000.

F. S. Melo and M. I. Ribeiro. Q-learning with linear function approximation. In *International Conference on Computational Learning Theory*, pages 308–322. Springer, 2007.

A. Sala, H. Zheng, B. Y. Zhao, S. Gaito, and G. P. Rossi. Brief announcement: revisiting the power-law degree distribution for social graph analysis. In *PODC*, 2010.

D. Shin, S. Cetintas, K.-C. Lee, and I. S. Dhillon. TUMBLR blog recommendation with boosted inductive matrix completion. In *CIKM'15*, 2015.

A. Slivkins. Introduction to multi-armed bandits. *Found. Trends Mach. Learn.*, 12(1-2):1–286, 2019.

X. Song, B. L. Tseng, C.-Y. Lin, and M.-T. Sun. Personalized recommendation driven by information flow. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 509–516, 2006.

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

R. S. Sutton, A. G. Barto, et al. Introduction to reinforcement learning. 1998.

L. Tang, Y. Jiang, L. Li, and T. Li. Ensemble contextual bandits for personalized recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 73–80, 2014.

S. Tang and J. Yuan. Optimizing ad allocation in social advertising. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1383–1392, 2016.

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

R. J. Trudeau. *Introduction to graph theory*. Courier Corporation, 2013.

S. Vaswani, B. Kveton, Z. Wen, M. Ghavamzadeh, L. V. Lakshmanan, and M. Schmidt. Model-independent online learning for influence maximization. In *ICML*, pages 3530–3539. PMLR, 2017a.

S. Vaswani, B. Kveton, Z. Wen, M. Ghavamzadeh, L. V. S. Lakshmanan, and M. Schmidt. Model-independent online learning for influence maximization. In *ICML*, 2017b.

S. S. Villar, J. Bowden, and J. Wason. Multi-armed bandit models for the optimal design of clinical trials: benefits and challenges. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 30(2):199, 2015.

H. Wang, Q. Wu, and H. Wang. Factorization bandits for interactive recommendation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

R. Wang, S. S. Du, L. Yang, and R. R. Salakhutdinov. On reward-free reinforcement learning with linear function approximation. *Advances in neural information processing systems*, 33: 17816–17826, 2020.

Y. Wang, R. Wang, S. S. Du, and A. Krishnamurthy. Optimism in reinforcement learning with generalized linear function approximation. *arXiv preprint arXiv:1912.04136*, 2019.

Z. Wen, B. Kveton, M. Valko, and S. Vaswani. Online influence maximization under independent cascade model with semi-bandit feedback. *Advances in neural information processing systems*, 30, 2017.

Q. Wu, Z. Li, H. Wang, W. Chen, and H. Wang. Factorization bandits for online influence maximization. In *KDD*, 2019.

Z. Xie, T. Yu, C. Zhao, and S. Li. Comparison-based conversational recommender system with relative bandit feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1400–1409, 2021.

L. Yang and M. Wang. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning*, pages 10746–10756. PMLR, 2020.

J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li. Social influence locality for modeling retweeting behaviors. In *IJCAI*, 2013.

Z. Zhang, W. Chen, X. Sun, and J. Zhang. Online influence maximization with node-level feedback using standard offline oracles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9153–9161, 2022.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Apprentissage séquentiel pour la diffusion d'information

Mots clés: maximisation de l'influence, apprentissage par renforcement, bandits à plusieurs bras, borne supérieure de confiance

Résumé: Motivés par les scénarios de diffusion de l'information et de publicité dans le les réseaux sociaux, nous étudions un problème de maximisation de l'influence (MI) dans lequel on suppose que l'on en sait peu sur le réseau de diffusion ou sur le modèle qui détermine comment l'information peut se propager. Dans un tel environnement incertain, on peut se concentrer sur des campagnes de diffusion à plusieurs tours, avec l'objectif de maximiser le nombre d'utilisateurs distincts qui sont influencés ou activés, à partir d'une base de nœuds influents. Au cours d'une campagne, les graines de propagation sont sélectionnées séquentiellement lors de tours consécutifs, et les commentaires sont collectés sous la forme des nœuds activés à chaque tour. L'impact (récompense) d'un tour est alors quantifié par le nombre de nœuds nouvellement activés. En général, il faut maximiser la propagation totale de la campagne, comme la somme des récompenses des tours. Nous considérons deux sous-classes de d'IM, *Contextual Influence Maximization with Persistence* (CIMP) et *Episodic Contextual Influence Maximization with Persistence* (ECIMP), où (i) la récompense d'un tour d'une campagne en cours consiste uniquement en de nouvelles activations (non observées lors des tours précédents de cette campagne), (ii) le contexte du tour et les données historiques des tours précédents peuvent être exploités pour apprendre la meilleure politique, et (iii) ECIMP est CIMP répété plusieurs fois, ce qui permet d'apprendre également des campagnes précédentes. Ce problème est directement motivé par les scénarios du monde réel de la diffusion de l'information dans le marketing d'influence, où (i) seule la première / unique activation d'un utilisateur cible présente un intérêt (et cette activation persistera comme une activation acquise, latente, tout au long de la campagne). (ii) de précieuses informations secondaires sont disponibles pour l'agent d'apprentissage. Dans ce contexte, une approche d'exploration-exploitation pourrait être utilisée pour apprendre les principaux paramètres de diffusion sous-jacents, tout en exécutant les campagnes. Pour CIMP, nous décrivons et comparons deux méthodes de bandits à bras multiples contextuels, avec des limites supérieures de confiance sur le potentiel restant des influenceurs, l'une utilisant un modèle linéaire généralisé et l'estimateur de Good-Turing pour le potentiel restant (`GLM-GT-UCB`), et l'autre adaptant directement l'algorithme LinUCB à notre cadre (`LogNorm-LinUCB`). Pour ECIMP, nous proposons l'algorithme `LSVI-GT-UCB` qui implémente le principe d'optimisme face à l'incertitude pour l'apprentissage par renforcement, avec approximation linéaire. L'agent d'apprentissage estime pour chaque nœud de départ son potentiel restant avec un estimateur de Good-Turing, modifié par une fonction Q estimée. Nous montrons qu'ils surpassent les performances des méthodes de base utilisant les idées les plus récentes, sur des données synthétiques et réelles, tout en présentant un comportement différent et complémentaire, selon les scénarios dans lesquels ils sont déployés.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Scalable Model-Free Algorithms for Influencer Marketing

Keywords: influence maximization, reinforcement learning, multi-armed bandits, upper confidence bound

Abstract: Motivated by scenarios of information diffusion and advertising in social media, we study an *influence maximization* (IM) problem in which little is assumed to be known about the diffusion network or about the model that determines how information may propagate. In such a highly uncertain environment, one can focus on *multi-round diffusion campaigns*, with the objective to maximize the number of distinct users that are influenced or activated, starting from a known base of few influential nodes. During a campaign, spread seeds are selected sequentially at consecutive rounds, and feedback is collected in the form of the activated nodes at each round. A round's impact (reward) is then quantified as the number of *newly activated nodes*. Overall, one must maximize the campaign's total spread, as the sum of rounds' rewards. We consider two sub-classes of IM, *Contextual Influence Maximization with Persistence* (CIMP) and *Episodic Contextual Influence Maximization with Persistence* (ECIMP), where (i) the reward of a given round of an ongoing campaign consists of only the *new activations* (not observed at previous rounds within that campaign), (ii) the round's context and the historical data from previous rounds can be exploited to learn the best policy, and (iii) ECIMP is CIMP repeated multiple times, offering the possibility of learning from previous campaigns as well. This problem is directly motivated by the real-world scenarios of information diffusion in *influencer marketing*, where (i) only a target user's *first / unique* activation is of interest (and this activation will *persist* as an acquired, latent one throughout the campaign), and (ii) valuable side-information is available to the learning agent. In this setting, an explore-exploit approach could be used to learn the key underlying diffusion parameters, while running the campaigns. For CIMP, we describe and compare two methods of *contextual multi-armed bandits*, with *upper-confidence bounds* on the remaining potential of influencers, one using a generalized linear model and the Good-Turing estimator for remaining potential (`GLM-GT-UCB`), and another one that directly adapts the LinUCB algorithm to our setting (`LogNorm-LinUCB`). For ECIMP, we propose the algorithm `LSVI-GT-UCB`, which implements the *optimism in the face of uncertainty* principle for episodic reinforcement learning with linear approximation. The learning agent estimates each seed node's remaining potential with a Good-Turing estimator, modified by an estimated Q-function. We show that they outperform baseline methods using state-of-the-art ideas, on synthetic and real-world data, while at the same time exhibiting different and complementary behavior, depending on the scenarios in which they are deployed.