GRAPH-BASED RECOMMENDATION SYSTEM

Kaige Yang, Laura Toni

Dept. of Electronic and Electrical Engineering
University College London, London, UK
{kaige.yang.11, l.toni}@ucl.ac.uk

ABSTRACT

In this work, we study recommendation systems modelled as contextual multi-armed bandit (MAB) problems. We propose a graph-based recommendation system that learns and exploits the geometry of the user space to create meaningful clusters in the user domain. This reduces the dimensionality of the recommendation problem while preserving the accuracy of MAB. We then study the effect of graph sparsity and clusters size on the MAB performance and provide exhaustive simulation results both in synthetic and in real-case datasets. Simulation results show improvements with respect to state-of-the-art MAB algorithms.

***Index Terms*—** Recommendation system, contextual multi-armed bandit, community detection.

1. INTRODUCTION

Recommending products to users have been an essential function of commercial websites as Amazon and Netflix, etc. [1]. The aim of a recommendation agent is to propose to a user the product (or item) that will generate a positive reaction: a product purchase in Amazon, a link click in ads website, etc. This user response increases the agent payoff, which ultimately needs to be maximized. The effectiveness of a recommendation system depends on the knowledge of users' preference: the deeper the knowledge, the more the tailored is the recommended item. The challenge is that these preferences usually are not known a priori and need to be built online by trial and error for each user. This learning process can be formalised by multi-armed bandit (MAB) framework [2–5].

The performance of MAB learning strategies scales with the ambient dimension, either linearly or as a square root [6], which makes the problem intractable in scenarios with infinitely large strategy sets, as in recommendation systems.

To overcome the dimensionality limitation, clustering techniques have been proposed to properly quantize the context space (e.g., the user space) [7]. Users preference relationships can be encoded in a graph, where adjacent nodes represents users with similar preferences [8, 9]. This graph can be known a-priori or it can be inferred based on the past users' feedbacks (past payoffs). We are interested in this latter case and in recent works in which the geometrical and irregular structure of the context have been considered [9–16]. In [10], authors proposed CLUB, an online clustering strategy where m clusters are optimized for n ($>> m$) users, where only one recommendation per cluster is optimized. In [14], a similar idea has been implemented on both user and item side to propose COFIBA. In both CLUB and COFIBA, an iterative graph learning process is considered with a fully connected graph as starting point. At each recommendation opportunity, edges are deleted if connecting users with different enough payoff. Any connected component will then form a cluster. The leads to a very simple and yet effective MAB problem, but with the limitation of a i) limited clustering strategy, ii) no possibility of recovering from inaccurate payoff estimate (edges can only be deleted from the known graph and they cannot be added in case of edges wrongly deleted in the past), and iii) a number of clusters rapidly increasing with time, which we show not to be the best trend for MABs. In contrast, DYnUCB [17] groups users via K-means assigning users dynamically into clusters. However, it requires an input as a pre-defined number of clusters. While theoretically K can be optimized with iterative solutions (e.g., elbow method), in practice, an appropriated cluster number is typically unknown, hard to guess, and dynamic over time (as users might appear or disappear).

To overcome these limitations, in this paper we propose SCLUB-CD, a novel graph-based MAB problem that learns and exploits the geometry of the user domain in an online fashion. Specifically, at each recommendation opportunity, a user graph is constructed based on estimated user preference. Then, the graph is divided into clusters based on the community detection algorithm [18]. Our main contributions are:

- to adopt graph clustering into MABs to propose a dynamic graph estimation *and* clustering.
- to show that MABs are more efficient when the number of cluster remains limited over time. Therefore, our proposed recommendation system keeps the number of clusters limited over time, opposite behaviour with respect to CLUB.
- to test the proposed algorithm in both synthetic and

realist dataset, showing improved performance with respect to LinUCB and CLUB state-of-the-art MAB problems.

2. MAB FOR RECOMMENDATIONS

We now describe the basics on recommendation systems and how these problems, when tackled as MAB algorithms, can benefit from context clustering.

Let each user $i \in \mathcal{I}$ be identified by its preferences $\boldsymbol{u}_i \in \mathbf{R}^d$, and let the product $k \in \mathcal{K}$ be identified by its own feature vector $\boldsymbol{x}_k \in \mathbf{R}^d$ (identifying information such as size, colour and price of the product), with d being the dimension of the user and product vector, respectively, and $|\mathcal{I}| = N$. While $\boldsymbol{x}_k$ is known to the agent, the user preference vectors $\boldsymbol{u}_i$ need to be learned on the fly. To this effect, the agent makes sequential recommendation and observes the outcome (appreciation of the recommended product). Formally, at each recommendation opportunity $t = 1, 2, ..., T$, the agent receives a user index $i_t \in \mathcal{I}$ to serve content to, with i_t selected uniformly at random from $\mathcal{I}$. It also receives the set of possible products to recommend $\mathcal{C}_t \subset \mathcal{K}$, with $|\mathcal{C}_t| = C$. The agent then recommends one product out of the available ones to user i_t and observe the user's feedback in the form of instantaneous payoff a_t. The payoff is assumed to be a linear function of the product features $\boldsymbol{x}_k$ and user preference vector $\boldsymbol{u}_{i_t}$ with a noise term $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. [4]. Namely,

$$a_t = {\boldsymbol{u}_{i_t}}^T \boldsymbol{x}_k + \boldsymbol{\epsilon}_{i_t}, \quad k \in \mathcal{C}_t \tag{1}$$

with $a_t \in [0, 1]$, with 1 being the highest appreciation and 0 the lowest, and $\boldsymbol{\epsilon}_{i_t}(\boldsymbol{x})$ being a random Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. Note that ${\boldsymbol{u}_{i_t}}^T \boldsymbol{x}$ is the expected payoff received from user i_t for $\boldsymbol{x}$, while a_t is the instantaneous one.

Let us assume that users are clustered in M non overlapping clusters based on their preferences, with V_j, $j = 1, 2, .., M$ being the jth cluster, and M be unknown a priori. Due to the linear payoff scenario, users in the same clusters will experience similar payoff functions. It follows that rather than having a preference vector per user, the agent can identify and learn a preference vector per cluster. Therefore, the agent needs to learn only M preferences vector rather than N, with $N \gg M$. This comes at the price of an approximation in the estimation of the linear payoff, and therefore a suboptimality in the recommendation. More formally, each user cluster V_j has the preference vector $\boldsymbol{u}_j^{\mathrm{c}}$ representing the common parameter vector shared by users within the cluster, leading to an *estimated* mean payoff given by ${\boldsymbol{u}_{j(i_t)}^{\mathrm{c}}}^T \boldsymbol{x}_k + \boldsymbol{\epsilon}_{j(i_t)}$, with $k \in \mathcal{C}_t$ and $j(i_t)$ being the cluster index whose user i belongs to. Note that the actual payoff (per user) is given by (1), while the agent will estimate the above one per cluster. From here the suboptimality of the clustering-based recommendations.

The agent aims to minimise the cumulative regret R_T over the time horizon T defined as $R_T = \sum_{t=1}^{T} r_t$, with r_t being the regret at time t, defined as the difference between the payoff incurred by the algorithm and the optimal payoff. Formally,

$$r_t = \max_{k \in \mathcal{C}_t} \{ \boldsymbol{u}_{i_t}^T \boldsymbol{x}_k \} - {\boldsymbol{u}_{i_t}}^T \boldsymbol{x}_t \tag{2}$$

where, r_t is the regret at time t. Following the MAB theory, the cumulative regret is minimized if products are selected as follows

$$k_t = \arg\max_{k \in \mathcal{C}_t} (\boldsymbol{u}_{j(i_t)}^{\mathrm{c}} \boldsymbol{x}_k) + CB_{j(i_t)}(\boldsymbol{x}_k) \tag{3}$$

The quantity $CB_{j(i_t)}$ is the upper confidence bound of each arm with respect to cluster $j(i_t)$. Basically, a product is selected if the expected payoff is high and it is low the uncertainty on this estimated payoff.

3. GRAPH-BASED MAB

We now describe the proposed SCLUB-CD algorithm, depicted in Algorithm 1. At the recommendation opportunity t, the agent estimates an unweighted and undirected graph $G_t = (\mathcal{V}, \mathcal{E}_t, W_t)$, with $\mathcal{V}$ being the vertex set representing the N users[1] with $|\mathcal{V}| = N$, $\mathcal{E}_t$ and W_t the edge sets and the $N \times N$ adjacency matrix estimated at t, respectively. The graph G_t is obtained by following a 3 steps iterative method:

STEP 1. First an undirected and *weighted* graph $\tilde{G}_t = (\mathcal{V}, \mathcal{E}_t, \tilde{W}_t)$ is estimated. Given the current knowledge of the system, users preferences are estimated minimizing the linear least-square estimate of $\boldsymbol{u}$ by multiplying the inverse correlation matrix M_t and the bias vector $\boldsymbol{b}_t$. Formally,

$$\hat{\boldsymbol{u}}_{i,t} = M_{i,t}^{-1} \boldsymbol{b}_{i,t}, \;\; i = 1, 2, ..., N \tag{4}$$

Then, the graph weights $\tilde{\mathbf{w}}_{t,i,j}$ in $\tilde{G}_t$ are evaluated as the Gaussian *RBF-distance* between $\hat{\boldsymbol{u}}_{i,t}$ and $\hat{\boldsymbol{u}}_{j,t}$.

STEP 2. Then, the graph is converted in a sparse and unweighted graph. Sparsity is motivated by the need to tune the dimensionality of the user space, while the binary weights are introduced mainly to increase the robustness of the algorithm to graph estimation errors. Both aspects will be discussed in the results section.

We introduce the hyperparameter n, such that the top n, $n \ll N$, largest weights in $w_{t,i,j}, j = 1, 2, ..., N$ are encoded as 1, the remaining are 0. The result graph is G_t. Note that we do not impose sparsity by setting to zero all weights below a given threshold value (more common approach) but we introduce n instead. This is to better control the number of clusters, that needs to remain low for an efficient learning.

STEP 3. Once the graph G_t is estimated, the clusters $\hat{V}_{1,t}$,$\hat{V}_{2,t}$,...,$\hat{V}_{M,t}$ are derived via community detection applying the Louvain Method [18]. Preferences per cluster are then estimated (as shown in Algorithm 1) and the algorithm selects

[1]Without loss of generality, we assume the number of active users constant over time.

Initial: $b_{i,0} = \mathbf{0} \in \mathbf{R}^d$, $M_{i,0} = \boldsymbol{I} \in \mathbf{R}^{d\times d}$, $\hat{\boldsymbol{u}}_{i,0} = \mathbf{0} \in \mathbf{R}^d$, $i \in [1, N]$;
Input: Edge deletion parameter $n \in (0, N]$;
for $t = 1, 2, ..., T$ **do**
  Set $\hat{\boldsymbol{u}}_{i,t-1} = M_{i,t-1}^{-1}\boldsymbol{b}_{i,t-1}, i = 1, 2, ..., N$;
  Find $G_{t-1,weighted}$: Calculate pairwise *Gaussian RBF-distance* $\tilde{\mathbf{w}}_{t-1,i,j}$ between $\hat{u}_{t-1,i}$ and $\hat{u}_{t-1,j}, i, j = 1, 2, ..., N$;
  Find G_{t-1}: Set the largest n values of $\tilde{\mathbf{w}}_{t-1,i,j}, j = 1, 2, .., N$ as 1; Set the remaining to 0;
  Find $\hat{V}_{\hat{j}_t,t-1}$: Apply Louvain Method on G_t;
  Set

$$\bar{M}_{\hat{j}_{t,t-1}} = \boldsymbol{I} + \sum_{i\in\hat{V}_{\hat{j}_t,t-1}} (M_{i,t-1} - \boldsymbol{I})$$

$$\bar{\boldsymbol{b}}_{\hat{j}_t,t-1} = \sum_{i\in\hat{V}_{\hat{j}_t,t-1}} \boldsymbol{b}_{i,t-1}, \bar{\boldsymbol{u}}_{\hat{j}_t,t-1} = \bar{M}^{-1}_{\hat{j}_t,t-1}\bar{\boldsymbol{b}}_{\hat{j}_t,t-1}$$

  Find

$$k_t = \arg\max_{k\in 1,\dots,C} (\bar{\boldsymbol{u}}_{j(i_{t-1})}\boldsymbol{x}_{t,k}) + CB_{\hat{j}_{t-1}}(\boldsymbol{x}_{t,k})$$

$$CB_{\hat{j}_{t-1}}(\boldsymbol{x}) = \alpha\sqrt{\boldsymbol{x}^T\bar{M}_{\hat{j}_t,t-1}\boldsymbol{x}\log(t+1)}$$

  Receive $a_t \in [0, 1]$.
  Update:

$$M_{i_t,t} = M_{i_t,t-1} + \boldsymbol{x}_t\boldsymbol{x}_t^T, \boldsymbol{b_{i_t,t}} = \boldsymbol{b_{i_t,t-1}} + a_t\boldsymbol{x}_t$$

$$M_{i,t} = M_{i,t-1},\ \boldsymbol{b_{i,t}} = \boldsymbol{b_{i,t-1}}, i \neq i_t$$

end

Algorithm 1: SCLUB-CD Algorithm

the product k_t at opportunity t following the UCB method, i.e., following the minimization in (3). Once the product indexed by k_t with feature $\boldsymbol{x}_{k_t}$ is recommended, $\boldsymbol{x}_{k_t}$ is used to update $\hat{\boldsymbol{u}}_{i,t}$ along with the received payoff a_t via a standard linear least-square approximation of $\boldsymbol{u}_{i_t}$ (as shown in Algorithm 1) and a new loop starts.

4. SIMULATION RESULTS

4.1. Simulations Setup

We carried out results both in a synthetic and a realistic dataset. The synthetic case allows us to simulate a scenario in which we can actually control the similarity among users, while the realistic dataset has been implemented to validate our algorithm in real recommendation problems. In the synthetic case, $N = 100$ users are clustered in $M = 5$ clusters and $|\mathcal{K}| = 1000$ products are considered. Out of these 1000 products, at each trial t, a smaller pool C with size $C = 25$ is chosen uniformly at random from $\mathcal{K}$ as candidate for the recommendation[2]. We set the dimension of both the users and product features vectors to $l = d = 25$. To control the similarity among users within a cluster, intra-cluster noise σ_c is introduced. For each user i belonging to V_j, $\boldsymbol{u}_i$ is created by perturbing the $\boldsymbol{u}^c_{j(i)}$ with a white noise term drawn uniformly at random across from a zero mean normal distrbution with variance σ_c^2. The lower σ_c^2 the more compact the clusters. In the following simulation results, we consider both σ_c and σ_ϵ to be in the range $[0.25, 0.5]$, where we recall that σ_ϵ is the standard deviation of the payoff.

For the real-world datasets, we consider **LastFM**, containing tags of artists and record listened by users, and **Delicious**, including URLs bookmarked and tags provided by users, [19]. While **LastFM** represents a scenario named "few-hits" where users' preference are coherent (therefore it is reasonable to assume that users can be clustered), **Delicious** represents a "many-hits" scenario in which users' preferences are diverse (therefore the clustering is a strong approximation). Simulation results are averaged over 10 runs and provided in the following.

LastFM and **Delicious** datasets were processed following the same procedure in [12] which we brief here. First, tags that appear less than 10 times were removed. Second, all tags related to each specific item was formed as a TF-IDF vector. To reduce the dimension, PCA was applied and only the top 25 principle components were retained.

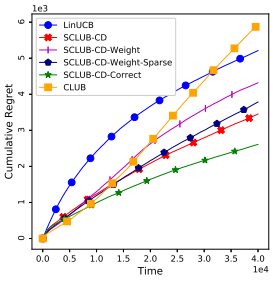
The proposed SCLUB-CD[3] is compared with respect to the state-of-the-art algorithms, namely LinUCB [20], CLUB [10]. In the synthetic dataset, we provide simulation results also for the SCLUB-CD in the case in which user clusters is known, but user preference is unknown. We label this method SCLUB-CD-Correct and it represents a lower bound in terms of cumulative regret. To study the effect of the weigted and sparse graph, we provide results also for two other baseline methods (modified version of the proposed SCLUB-CD): SCLUB-CD-Weight performs the clustering based on $\hat{G}_t$, while SCLUB-CD-Weight-Sparse keeps the top n largest Gaussian *RBF-distance* edges but preserving their weights.

4.2. Results

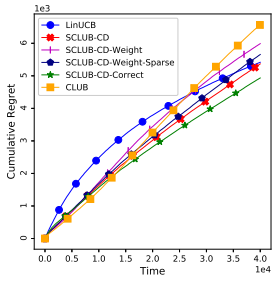
Fig. 1 shows the cumulative regret as a function of the recommendation opportunities in the case of synthetic dataset and under various combinations of reward noise and intra-cluster noise. In the long-term, SCLUB-CD outperforms its competitors consistently over all scenarios, with a substantial gap (in terms of cumulative regret) for lower intra-cluster noise ($\sigma_c = 0.25$), Fig. 1(a) and Fig. 1(c). It is interesting to observe that also when the intra-cluster noise increases (Fig. 1(b) and Fig. 1(d)), clustering the user space with the proposed approach still leads to a system improvement. With respect to LinUCB, which does not cluster the users, the gain is experienced because of the faster learning process: LinUCB learns N preference vector while SCLUB-CD learns M ones. With respect to CLUB, the gain is motivated by i) the online learning estimation of the graph at each time opportunity, ii) the graph-based clustering not limited to identify con-

[2]This is a common assumption in recommendation systems, therefore we apply it in both synthetic and realistic scenarios.

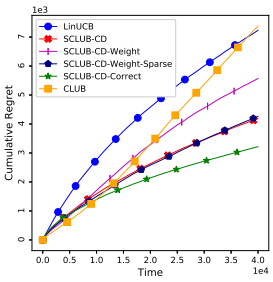
[3]Available at https://github.com/LASP-UCL/KaigeYang/tree/graph-based-recommendation-system.



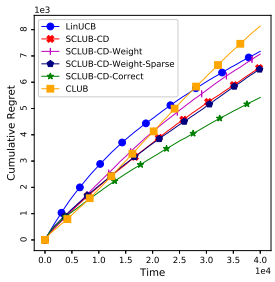
(a) $\sigma_c = 0.25$, and $\sigma_\epsilon = 0.25$.



(b) $\sigma_c = 0.5$, and $\sigma_\epsilon = 0.25$.

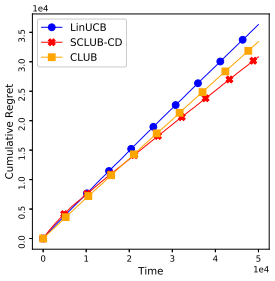


(c) $\sigma_c = 0.25$, and $\sigma_\epsilon = 0.5$.

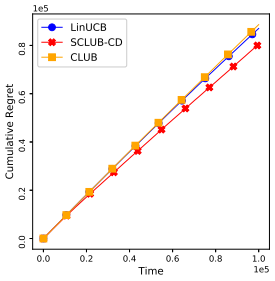


(d) $\sigma_c = 0.5$, and $\sigma_\epsilon = 0.5$.

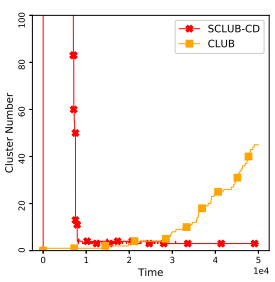
Fig. 1. Cumulative regret for the synthetic dataset.



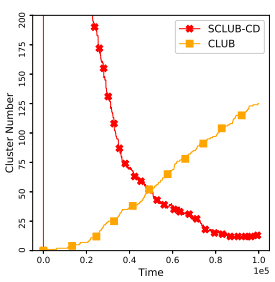
(a) LastFM



(b) Delicious



(c) LastFM Clustering



(d) Delicious Clustering

Fig. 2. Cumulative regret and cluster number for the real-world dataset.

nected components, iii) the binary and sparse modelling of the graph. In Fig. 1(c), LinUCB might outperform SCLUB-CD for time horizon greater than 50000. However, we considered simulations with $N = 100$, to have a fair comparison with LinUCB. By increasing the dimensionality of the user space this potential crossing point is shifted far away in time [21].

Finally, the gap between SCLUB-CD and SCLUB-CD-Correct shows the potential room for improvement of the proposed algorithm, as discussed at the end of this session. The comparison with SCLUB-CD-Weight and SCLUB-CD-Weight-Sparse shows the gain in controlling the sparsity level (and therefore number of clusters) in the proposed algorithm.

Fig. 2 presents results on real world datasets. In **LastFM**, SCLUB-CD maintains a leading margin, this is due to a better clustering methodology. In particular, CLUB tends to indentify many clusters, while SCLUB-CD identifies $M = 3$ user clusters only with $n = 300$, as shown in Fig.2(c). This means that the proposed approach is able to find the right tradeoff between dimensionality reduction and approximation in clustering users.

In **Delicious**, SCLUB-CD still outperforms baseline algorithms, but the leading margin is smaller. Results show it groups users into $M = 11$ clusters with $n = 700$. Overall, delicious represents the "many-hits" scenario, in which each user is interested in a small amount and quite dissimilar websites. This means that each user will select few website only, therefore the agent can gather a small amount of feedbacks per user, which translates in a limited training set per user. Clustering users together allows to increase the dimension of the training set. Therefore SCLUB-CD outperforms LinUCB and CLUB, however due to the highly heterogenous scenario the approximation introduced by the clustering is affecting more the overall system. This justifies the reduced gain.

5. CONCLUSION

We proposed a graph-based bandit algorithm, which encodes users' similarity in preference by an undirected and unweighted graph and groups users into clusters. The key aspects of the proposed algorithm are that i) it adopts graph-based clustering to extract meaningful clusters, ii) the unweigted graph makes the system more robust to weights estimation errors, iii) the proposed method keeps the number of clusters limited (through n), unlike CLUB that has a number of clusters constantly increasing over time. All these components lead to an overall gain in terms of cumulative regret with respect to state-of-the-art algorithms. These results also opened new questions such as "What is the sensitivity of the MAB algorithm to the cluster size?", "Could we adopt graph signal processing to further improve the graph knowledge (exploiting also the smoothness of the reward function on the user graph)?" As future works, we will be addressing these open questions.

6. REFERENCES

[1] "Amazon," Https://www.amazon.co.uk/, accessed: 2018-04-15.

[2] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–536, 1952.

[3] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized qos-aware web service recommendation and visualization," *IEEE Transactions on Services Computing*, vol. 6, no. 1, pp. 35–47, 2013.

[4] J. Liu, P. Dolan, and E. R. Pedersen, "Personalized news recommendation based on click behavior," in *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 2010, pp. 31–40.

[5] H. K. Kim, J. K. Kim, and Y. U. Ryu, "Personalized recommendation over a customer network for ubiquitous shopping," *IEEE Transactions on Services Computing*, vol. 2, no. 2, pp. 140–151, 2009.

[6] S. Bubeck, N. Cesa-Bianchi *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[7] A. Slivkins, "Contextual bandits with similarity information." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2533–2568, 2014.

[8] T. T. Nguyen and H. W. Lauw, "Dynamic clustering of contextual multi-armed bandits," in *Proc. ACM Int. Conf. on Information and Knowledge Management*, ser. CIKM '14, 2014.

[9] F. Rezaeimehr, P. Moradi, S. Ahmadian, N. N. Qader, and M. Jalili, "Tcars: time-and community-aware recommendation system," *Future Generation Computer Systems*, vol. 78, pp. 419–429, 2018.

[10] C. Gentile, S. Li, and G. Zappella, "Online clustering of bandits," *CoRR*, vol. abs/1401.8257, 2014. [Online]. Available: http://arxiv.org/abs/1401.8257

[11] S. Li, C. Gentile, A. Karatzoglou, and G. Zappella, "Data-dependent clustering in exploration-exploitation algorithms," *arXiv preprint arXiv:1502.03473*, 2015.

[12] N. Cesa-Bianchi, C. Gentile, and G. Zappella, "A gang of bandits," in *Advances in Neural Information Processing Systems*, 2013, pp. 737–745.

[13] S. Li, C. Gentile, A. Karatzoglou, and G. Zappella, "Online context-dependent clustering in recommendations based on exploration-exploitation algorithms," *ArXiv*, vol. abs/1608.03544, 2016.

[14] S. Li, A. Karatzoglou, and C. Gentile, "Collaborative filtering bandits," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 539–548.

[15] N. Korda, B. Szorenyi, and S. Li, "Distributed clustering of linear bandits in peer to peer networks," in *ICML*, 2016.

[16] S. Caron, B. Kveton, M. Lelarge, and S. Bhagat, "Leveraging side observations in stochastic bandits," *ArXiv*, vol. abs/1210.4839, 2012.

[17] T. T. Nguyen and H. W. Lauw, "Dynamic clustering of contextual multi-armed bandits," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 1959–1962.

[18] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[19] I. Cantador, P. L. Brusilovsky, and T. Kuflik, *Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011)*. ACM, 2011.

[20] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 661–670.

[21] K. Yang and L. Toni, "Graph-based recommendation systems," *ArXiv*, vol. submit/2313688, 2018.