MCS+: An Efficient Algorithm for Crawling the Community Structure in Multiplex Networks

RICKY LAISHRAM, Syracuse University, USA
JEREMY D WENDT, Sandia National Laboratories, USA
SUCHETA SOUNDARAJAN, Syracuse University, USA

In this paper, we consider the problem of crawling a multiplex network to identify the community structure of a layer-of-interest. A multiplex network is one where there are multiple types of relationships between the nodes. In many multiplex networks, some layers might be easier to explore (in terms of time, money etc.). We propose MCS+, an algorithm that can use the information from the easier to explore layers to help in the exploration of a layer-of-interest that is expensive to explore. We consider the goal of exploration to be generating a sample that is representative of the communities in the complete layer-of-interest.

This work has practical applications in areas such as exploration of dark (e.g., criminal) networks, online social networks, biological networks etc. For example, in a terrorist network, relationships such as phone records, email records etc are easier to collect; in contrast, data on the face-to-face communications are much harder to collect, but also potentially more valuable.

We perform extensive experimental evaluations on real-world networks, and we observe that MCS+ consistently outperforms the best baseline – the similarity of the sample that MCS+ generates to the real network is up to three times that of the best baseline in some networks. We also perform theoretical and experimental evaluations on the scalability of MCS+ to network properties, and find that it scales well with the budget, number of layers in the multiplex network and the average degree in the original network.

CCS Concepts: • **Mathematics of computing** → **Graph algorithms**; • **Information systems** → *Data mining*; *World Wide Web*; Social networks.

Additional Key Words and Phrases: multplex networks, community detection, network crawling, multi-armed bandit

ACM Reference Format:
Ricky Laishram, Jeremy D Wendt, and Sucheta Soundarajan. 2020. MCS+: An Efficient Algorithm for Crawling the Community Structure in Multiplex Networks. *ACM Trans. Knowl. Discov. Data.* 1, 1 (August 2020), 32 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Multilayer networks are used to model various complex relationships including transportation systems [12, 23, 67], social networks [35, 69], biological networks [7, 25], and many others. A *multiplex network* is a type of multilayer network in which nodes participate in multiple types of interactions or layers [43, 49]. For example, consider the NoordinTop multiplex network [21, 26], as shown in Figure 1. In this network, the nodes represent people and the edges represents relationships

Authors' addresses: Ricky Laishram, Syracuse University, Syracuse, NY, USA, 13210, rlaishra@syr.edu; Jeremy D Wendt, jdwendt@sandia.edu, Sandia National Laboratories, Albuquerque, NM, USA; Sucheta Soundarajan, susounda@syr.edu, Syracuse University, Syracuse, NY, USA.

1556-4681/2020/8-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

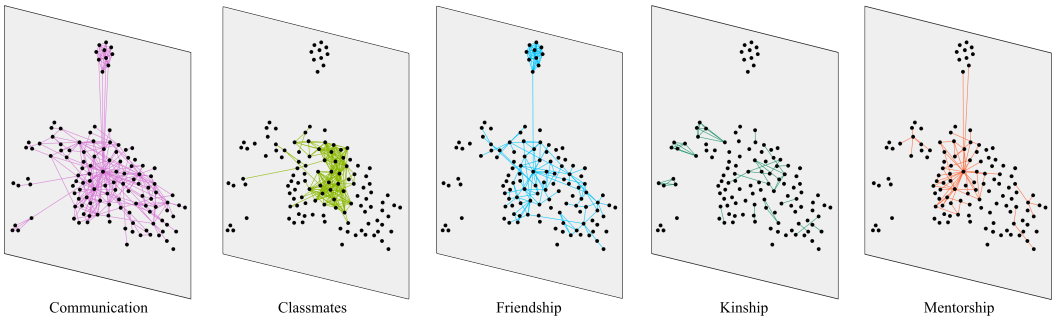


Fig. 1. The different layers in the NoordinTop network. The nodes are people and the edges are the relationships (depending on the layer) between the nodes.

between the people. There are multiple types of relationships, corresponding to the different layers, including communication, classmate, friendship, kinship, and mentor-mentee. Another example is a network of protein interactions – the layers represent different types of interactions such as direct interaction, physical interaction, association, co-localization, etc [60]. Multilayer networks can be categorized according to whether their layers are node aligned, disjoint, of the same size, and so on [36].

To explore a network, we start with a set of seed nodes and expand on it by *crawling* – collecting the neighbors of the observed nodes until the sample has the desired property [2, 28, 38]. In this paper, we consider the problem of *observing a multiplex network by crawling*, with the end goal of generating a community representative sample of the layer of interest. When crawling a multiplex network, one begins with a small number of nodes (or a single node) and gradually expands the observed sample by iteratively 'querying' for the neighbors of the observed nodes in a particular layer.

Multiplex networks pose a special challenge, as the cost of gathering data might be different for different layers. For instance, in a communication multiplex network, some modes of communications might be easier to follow than others; and in the protein interaction network the cost of experiments to discover different types of interactions might be different. As an example, in the NoordinTop terrorist network it is much easier to collect data on the classmate relationship compared to the face-to-face communication; and in protein interactions network the reliability of the experiments to obtain the different types of interactions is not the same – direct interaction generally requires more experimental data to infer [57, 59, 63].

Collecting a community representative sample from a multiplex network is a practical and important problem due to the prevalence of multiplex networks in the real world. The multiplex nature of the network gives rise to opportunities not found when crawling single-layer networks: even if the layer of interest is costly to observe (e.g., face-to-face interactions), we may be able to use cheaper layers (e.g., phone calls) to infer information about the layer of interest, and thus perform a more efficient crawl. However, when doing so, we must balance the trade-off between exploration of the layer of interest and the other layers – after all, we do not know the edge overlap between the layers beforehand. Another challenge is that that there might be different reliabilities associated with different layers – for example, people are not likely to remember all people with whom they had a personal interaction, but may keep a complete record of their email communication.

We consider two variations on this same basic problem, depending on the response to a neighborhood query on a node – *Reliable Query Response* and *Unreliable Query Response* (Section 3.1). In the Reliable Query Response model, when a node is queried in a layer, all of its neighbors in

the layer are returned in the response. This setting is observed in the case of a query against a database. The second type of query response we consider is the Unreliable Query Response model. In this case, a query for the neighbors of a node in a layer returns only some of the neighbors,[1] and multiple queries on the same node in the same layer might be needed to get all the neighbors.

There are various challenges associated with this problem:

(1) Data in the layer of interest is costly to gather. Thus, neighborhood queries on this layer must be made intelligently.
(2) Initially, we know only a few nodes and which layers exist. We do not know the true properties of most nodes, or even the size of the network.
(3) We do not know the edge overlap between the layer of interest and the other layers. The algorithm must learn this during the sampling process and balance the amount of budget allocated to exploration of the different layers.
(4) Because the community structure of the layer of interest is not known ahead of time, it is challenging to directly evaluate performance during the sampling process.

In this paper we propose *MultiComSample+* (MCS+), an algorithm for generating a community representative sample of the layer of interest from a multiplex network. MCS+ uses multi-armed bandits to guide a network crawler to explore the layer of interest.

The main contributions of this paper are:

(1) We consider the problem of generating a community representative sample of a layer of interest by making use of the information from the other layers that are easier to explore.
(2) We propose *MultiComSample+*, an algorithm to guide a crawler in exploring the layer of interest by making use of the other layers.
(3) Through extensive experimental evaluations across networks from different domains, we show that MCS+ outperforms a wide variety of baseline algorithms. In some networks, it outperforms the best baseline by up to a factor of 3.

This work is an extension of [39]. In this paper, we improve upon the earlier algorithm in the following ways:

(1) In the earlier paper, the proposed algorithm (MCS) could not handle node attributes. In this paper, we propose MCS+, which is able to take into account the node attributes (if available). We additionally improved the budget allocation and the reward calculation.
(2) In the earlier paper, we considered the *Random Unreliable Query Response* model. In this paper, we also consider a *Hierarchical Unreliable Query Response* model. This is a more realistic unreliable query response model in many situations.
(3) We include theoretical analysis of the running time and sensitivity of the algorithm to various network properties.
(4) We include experiments to analyze the behavior of MCS+ with respect to (a) the new *Hierarchical Unreliable Query Response*, (b) node attributes, (c) running time as network properties vary, (d) sensitivity to various network properties and (e) performance under different community detection algorithms (for example Louvain, Infomap, OSLOM etc.).

The source code for MCS+[2] and all the datasets used for experiments are publicly available.

[1] The neighbors returned might be random, as in the case of data collection thorough face-to-face communication, or paginated in the case of many online API.

[2] https://github.com/rlaishra/MCS-

2 RELATED WORK

Although there has been little work addressing the problem of crawling multiplex networks, there has been a significant amount of work in the areas of multiplex networks in general, community detection on both single layer and multiplex networks, and crawling single layer networks. In this section, we provide an overview of these works.

2.1 Multiplex Networks

In many networks, the edges can be classified as belonging to different categories representing different type of relationships, actions, interactions etc. In such networks, the edges belonging to the same category can be grouped together into a individual network or *'layer'*; and the network can be represented by a collection of such 'layers'. Such networks are referred to as *multiplex networks* [43, 49].

In [36], Kivela *et al.* differentiate multiplex networks from multilayer networks. Multiplex networks are node aligned: that is, all nodes exist in all layers. Multilayer networks do not have this constraint. Multiplex networks can thus be considered to be a special type of multilayer network.

In the context of social science, multiplex networks have been used to model the complex relationships between people. In [66] Verbrugge studied friendship networks based on three types of relationships – kin, neighbor and coworker. Krackhardt [35] proposed studying social structures by representing them as three-dimensional networks with the third dimension being the type of relationship (the first two dimensions are the rows and columns of adjacency matrix). McPherson, Smith-Lovin & Cook [48] studied the effects of different types of relationships with regards to the homophily observed in human relationships. Due the difficulty in collection of such data, these works are generally limited to only small networks.

Multiplex networks have been used to study online social networks such as Twitter, Facebook, etc. For example, in the Twitter network, the different layers can correspond to different types of interactions – follower, friend, reply, mention, retweet, etc. In [52], Omodel, De Domenico & Arenas studied the properties of the different layers in the Twitter multiplex network. Szell, Lambiotte & Thurner [61] studied a network of players in a massively multiplayer online game. In this network there are six different interaction types between players, corresponding to a multiplex network with six networks. The authors explored how these different layers interact to give rise to the social system organization observed in the game. In [30, 33] the authors extended link prediction methods to multiplex networks.

Social networks are not the only type of networks that have seen widespread adoption of the multiplex network model. In [20], De Domenico *et al.* modeled transportation network as a multiplex network with the layers being mode of transport. Cardillo *et al.* [12] also used multiplex networks to study the air transport network – in this case, the layers are the different airlines.

Multiplex networks have also been used extensively to represent biological networks. In [19] De Domenico *et al.* represented protein-protein interaction networks with multiplex networks. Wang *et al.* [68] proposed a method of characterization of cancer sub-types by making use of the multiplex network model.

2.2 Community Detection

The community detection problem has been extensively studied. Girvan & Newman [24] defined a community as a group of nodes that have more edges between them, but fewer edges to nodes in other communities and proposed a method of finding communities by iterative removal of edges with highest betweeness centrality. In [51] Newman & Girvan proposed modularity as a measure

of the strength of community structure, and algorithms based on modularity maximization are among the most popular algorithms for detecting communities in networks.

Various community detection methods based on modularity maximization have been proposed [9, 14, 50, 53]. One of the most popular is the *Louvain modularity maximization* method proposed by Blondel *et al.* This is a greedy algorithm to find communities in a network by grouping nodes that maximize modularity. Community detection methods based on modularity suffer from the resolution limit [40]; nonetheless they remain popular for their effectiveness and efficiency. Other methods of community detection include random walk based methods [10, 56], statistical inference [34, 54], etc.

The idea of community has also been extended to include overlapping communities. This is motivated by the observation that in many social networks communities may represent different types of groups – for example co-workers, friends, etc. Various community detection methods have been extended to apply in the case of overlapping communities [42, 56].

In the context of multilayer networks, there have been fewer works on community detection. It is not trivial to generalize community detection methods for single layer networks to the multilayer case due to the absence of a null model that takes the inter-layer edges into consideration [36]. So, most of the works have been on special types of multilayer networks.

In [49], Mucha *et al.* extended modularity to consider node-layer tuples and extended community detection methods to a type of multilayer networks called multislice networks. Tang, Wang & Liu [62] approached the problem of community detection in multi-dimensional networks by approaching the problem as a multi-objective optimization problem with the different objectives being the modularity in each layer. There are also several works [6, 8] that approach the problem by first identifying communities in each individual layer. In [8], Berlingerio, Pinelli & Calabrese defined a multiplex community as a closed frequent item-set where each item is a tuple of the layer and the community assigned to the node.

2.3 Network Sampling

In many applications, a network of interest might be too large to analyze. So, we need to reduce it to a sample that preserves the structure we are interested in studying. There has been a great deal of work on generating samples that satisfy certain requirements.

In [31, 44], the authors proposed a sampling algorithm to generate a subgraph that preserves the degree distribution and clustering coefficient. In [46], Maiya & Berger-Wolf proposed a method of generating a sample of a network that is representative of the community structure of the original network. These works can be considered as down-sampling – we have a known large graph and we want to generate a smaller sample of that graph, due to constraints on computation, storage etc.

A more challenging network sampling setting is the case of online sampling or '*crawling*'. In this case, the entire network is not known and we start off with a small sub-graph. The goal is to expand on this sub-graph by observing the neighbors of known nodes with the objective of generating a sub-graph that fulfills certain requirements. In such settings, a request of neighbors of a node is generally called a query; and there is generally a constraint on the number of queries that can be made.

In [1], Abiteboul, Preda & Cobena proposed an online sampling technique to calculate PageRank [11]. Maiya & Berger-Wolf [45] worked on a more generalized problem of crawling a network to find nodes with high centrality and proposed a technique based on expander graphs. Avrachenkov *et al.* [5] proposed an algorithm to generate a sample of an undirected network with the goal of maximizing the node coverage; i.e., the number of observed and queried nodes in the sample. In [38], the authors extended this to the directed case where there are limitations on the number of

Table 1. Notations used in this paper.

Notation	Description
$M = \langle V, E_0, \ldots, E_l \rangle$	Multiplex network with nodes V, and edges E_i in layer i.
$G_i = \langle V, E_i \rangle$	Layer G_i in multiplex network M.
$G'_i = \langle V', E'_i \rangle$	Sample of layer G_i.
$K(G)$	Communities found in graph G.
$\Psi(X, Y)$	Community similarity between X and Y.
Q_i	Set of nodes queried in G_i.
$\overline{Q}_i$	Set of nodes in G'_i that have not been queried in G_i.

neighbors returned (such as when there is an API rate limit). In [3], the authors studied the effect of various network properties on the performance of different crawling algorithms.

There have also been a few works exploring online sampling in multiplex networks. De Domenico *et al.* [18] explored the effect of random walks on multiplex networks and discovered that compared to single layer networks, the additional layers makes it easier for the walker to move between different regions of the network. In [71], Wendt *et al.* studied the problem of data collection through crawling in a multiplex network (Twitter) when the different layers have different exploration costs.

In this paper, we are concerned with the communities in the individual layers. Unlike earlier work, we are not proposing a new community detection algorithm; rather, we introduce a method to generate a representative sample whose community structure is similar to that of a layer of interest.

3 PROBLEM

Consider a multiplex network, $M = \langle V, E_0, E_1, \ldots, E_l \rangle$, where V is the set of nodes and E_i is the set of edges in layer i, with $l + 1$ layers. The individual layers are represented by $G_i = \langle V, E_i \rangle$.

The different layers can correspond to different types of relationships between the nodes in V. For example in the Twitter network, different layers may contain follower, mention, reply, or retweet relationships; in the case of gene interaction networks, the different layers might correspond to additive interaction, suppressive interaction, co-localization, etc. [57, 59, 63].

In many of these real-world networks, the cost of data collection might not be the same on different layers. In the Twitter network, due to API rate limits, it is faster to collect data on mentions, replies, etc. than followers; and in the gene interaction network, the experimental cost of different interactions might be different. Let $C = \{c_0, c_1, \ldots, c_l\}$ be the costs of data collection for layers $\{G_0, G_1, \ldots, G_l\}$ i.e. c_i is the cost required to make one query on a node in layer G_i. We use the term cost as a generalized term – in different applications it may refer to different resources. In the case of online social networks it is generally time because the API rate limits for different layers are different, in the case of protein interaction networks, it can be monetary because experiments to uncover different types of interactions requires different types of experiments.

Assume that we have a layer of interest and we have a limited budget (time, money, etc.) *how can we generate a sample that is representative of the community structure of the layer of interest through crawling*? In the rest of this paper, we will assume that layer 0 is the layer of interest.

Because we are collecting the sample through crawling, we assume that we begin with an initial set of nodes $V' \subset V$, and initial set of edges E'_i in each layer i. As the nodes in V' serve as a starting points to begin the exploration, $V' \neq \emptyset$, but it is possible that E'_i can be empty (for any or all i). Although there is no restriction on the size of V', we will assume that $|V'| \ll |V|$ since

in many real-world applications, we know of only a very few (or only a single) initial nodes. As we proceed through the crawling process, let $G_i^b = \langle V^b, E_i^b \rangle$ represent the sample of layer i found after spending a budget of b.

The communities in layer i of the multiplex network M are simply the communities in the graph G_i. Let K_i and K_i^b be the communities in G_i and G_i^b. We can represent the similarity between K_i and K_i^b as $\Psi\left(K_i, K_i^b\right)$. We describe our similarity measure in Section 3.2.

Formally, we consider the following problem: *Given an initial set of nodes V' and edges E' (between V'), and a total query budget B, how can we select which B nodes to query (and the layer to query them on) so that $\Psi\left(K_i, K_i^B\right)$ is maximized through crawling?*

Note that if there is a layer $i \neq 0$ and $c_i \geq c_0$, we can ignore layer i in favor of querying layer 0 directly. So for the rest of the paper, we will assume that for all the layers $i \neq 0$, $c_i < c_0$. We also assume that all the edges are undirected – in the case of directed edges, it can be easily extended as in [38].

3.1 Query Response Models

We consider two possible responses when a node u is queried for its neighbors in G_i – *Reliable Query Response* and *Unreliable Query Response*.

3.1.1 Reliable Query Response. In the *Reliable Query Response* model, when node u in layer i is queried, all the neighbors of u in G_i are returned. This is generally observed when we query against a database and all the information is available and accurate. As an example, if we consider a network of students connected by who took the same classes, we can accurately get all the neighbors from the school records. We will denote neighbors of u in G_i, returned through reliable query response model, with $N(u, G_i)$.

3.1.2 Unreliable Query Response. In this query response model, not all the neighbors of u in layer G_i are returned when queried. In this model, we assume that a node u in layer i has some *node uncertainty factor* associated with it, denoted by $\beta_i(u)$. When u in layer i, for every $v \in N(u, i)$ the probability of v being included in the response is $\beta_i(u)$. It should be noted that due to the probabilistic nature, repeat queries on u in layer i might not return the same results, and the uncertainty for any node is not known beforehand.

An example of this kind of query response is when we survey people about who they had face-to-face communication with – it is unlikely that they will remember everyone they talked to. On the other hand, when asked again on another day, a survey responder might recall someone they did not remember last time.

We can have different types of unreliable query response depending on the distribution of the node unreliability. In this paper, we consider two types of distribution – random and hierarchical. We will denote the random case by `R-UQR` and the hierarchical case by `H-UQR`.

In `R-UQR`, we assume that the node uncertainty factors are assigned randomly; and there is a relationship between the uncertainty of a node in different layers. For `H-UQR`, we make the assumption that the node uncertainty factor depends on some hierarchical structure in the network – nodes higher up in the hierarchy may have higher uncertainty factor than those lower or the reverse.

In our experimental analysis (Section 6.3.1), we assume that the hierarchy is based on the k-core structure – nodes in higher k-cores are higher up in the hierarchy; but any other notion of hierarchy are applicable.

In all cases, $\beta_i(u)$ is bounded between 0 and 1. We will denote the set of nodes returned on a query on u in G_i by $N'(u, G_i)$.

Another type of query response model is paginated response. In that case, all the neighbors might not be returned in a query response, but it is guaranteed that multiple queries on a node return different neighbors. This case can be easily handled as in [38] and is not discussed in this paper.

3.2 Community Similarity Measure

Suppose G_0 is the original layer of interest, and G_0' is some subgraph of G_0. How can we measure the similarity between the communities $K(G_0)$ and $K(G_0')$?

Various measures have been proposed in the literature to measure the similarity between two community partitions - such as Normalized Mutual Information (NMI) [17, 22, 37, 74], Jaccard Index [32, 72, 73], F1 Score [72], etc. These measures quantify the *community quality* – are communities grouped together in $K(G_0)$ also grouped together in $K(G_0')$, and the reverse? In this paper, we use NMI to measure the community quality since it is one of the most widely used. To perform a fair comparison, we restrict the comparison to only those nodes included in both graphs.

Because we are dealing with a sampled graph, in addition to the community quality, we also need to measure the community representation – *how many communities in* $K(G_0)$ *are represented in* $K(G_0')$? To measure this we use the a measure similar to the one proposed in [46]. We measure the community representation as the ratio of the number communities in $K(G_0)$ represented in $K(G_0')$ to the total number of communities in $K(G^0)$, weighted by the log of size of the communities. The intuition behind the weight is that, the penalty for missing larger communities should be higher than that of small communities.

To arrive at the final similarity measure, $\Psi(K(G_0), K(G_0'))$, we take the harmonic mean of the community quality and the community similarity.

4 METHODOLOGY

To solve the problem described in Section 3, we propose a novel algorithm called *MultiComSample+* (MCS+). In MCS+, there are multiple iterations with each iteration consisting of three major steps – Budget Allocation (Section 4.2), Sampling $G_{i\in(0,l]}$ (Section 4.3) and Sampling G_0 (Section 4.4). In the rest of the discussion, G_+ will be used to referred to any layer G_i with $i > 0$; i.e. any layer other than the one of interest. We will also use G_*' to denote the sample of some layer G_* found so far.

If some layers can be explored cheaply while still providing accurate information about which nodes in the G_0 should be queried, we should allocate more budget to the exploration of those layers (as compared to other cheap but less informative layers); and use this information to guide the exploration of G_0. This is the intuition behind MCS+.

Initially we do not know the similarities between G_0 and G_+. So, the budget for querying these various layers needs to be allocated carefully. If we allocate too much budget to G_+, it may be wasteful if G_+ does not contain information useful for selection of good nodes to query in G_0. On the other hand, if we allocate too much budget to G_0, we might miss information in G_+ that would have helped in better node selection. We describe the budget allocation process in Section 4.2.

After the budget has been allocated, in the next step (Sampling $G_{i\in(0,l]}$), MCS+ samples the layers $G_{i\in(0,l]}$ (i.e. the ones that are not of interest) through random walk with jump. Then, using the information gathered, nodes to query in layer G_0 are selected to generate sample G_0'. In Section 4.1 we bring together everything and describe *MultiComSample+* in detail.

In the discussion that follows, we will focus on the Reliable Query Response model. In Section 4.6, we will discuss the modifications to handle the different cases of Multiple Query Response model.

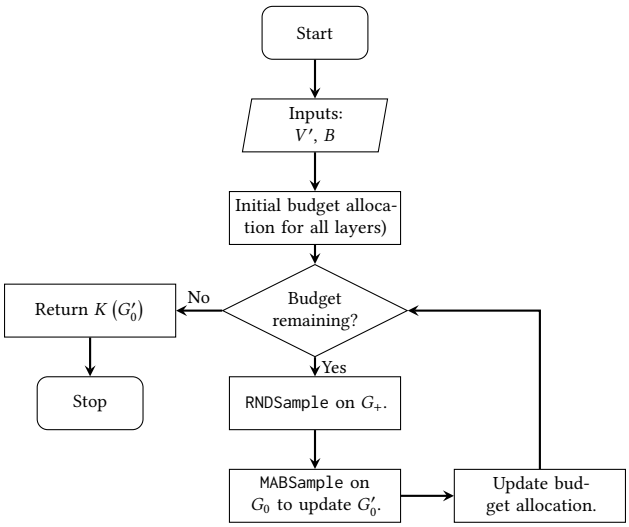


Fig. 2. The flowchart of the MCS+ algorithm.

4.1 *MultiComSample+* (MCS+)

As mentioned before, *MultiComSample+* (MCS+) consists of three major steps: (1) Budget Allocation (Section 4.2), (2) Sampling $G_{i\in(0,l]}$ (Section 4.3), and (3) Sampling G_0 (Section 4.4).

The budget allocation step splits up the maximum allowed budget into 'slices' and splits it for Sampling $G_{i\in(0,l]}$ and Sampling G_0. Then, the allocated budget is used to sample the 'cheaper layers', and the information from these layers is used to select nodes to query in G_0. If there is still budget left, MCS+ goes back to budget allocation where the updated edge overlap can be used to allocate budget more intelligently; else the algorithm returns $K\left(G'_0\right)$ and terminates.

4.2 Budget Allocation

Initially, before any queries have been made, MCS+ reserves a portion b of the total budget B for the first iteration of RNDSample and MABSample. In our experiments, we select b so that in is 10% of B. The budget b is divided so that each layer has enough budget to query the same number of nodes. That is, the budget allocated to layer G_i is,

$$b_i = \frac{c_i \cdot b}{\sum_{j\in[0,l]} c_j}$$

After running RNDSample and MABSample, MCS+ uses the information about the different layers to allocate budgets to G_+ appropriately. To do this, MCS+ computes the 'freshness' and 'layer importance' of each observed layer G'_+.

The freshness of layer G'_+ is defined as the average number of edges found in the last x queries on G'_+ during RNDSample. Similarly, *the layer importance of G'_+ is defined as the average of the last x community update distances due to nodes queried on G_0 selected due to their property in G'_+*. We discuss the community update distance in more details in Section 4.5. We will denote the freshness of G'_+ with $\phi\left(G'_+\right)$, and the layer importance with $\lambda\left(G'_+\right)$.

$\phi\left(G'_+\right)$ gives a measure of how many unobserved edges are expected on querying G_+, based on past observations, and $\lambda\left(G'_+\right)$ gives a measure how good the information from G'_+ was in selecting nodes to query in G_0. So, more budget is allocated to layers that have high freshness and layer

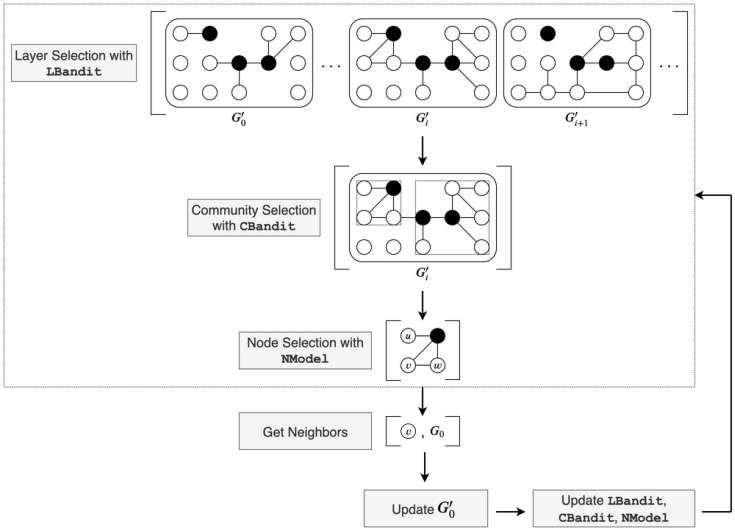


Fig. 3. Outline of MABSample with a toy example. Here the G'_0 is the sample of the layer of interest and the rest are the sample of the cheaper layers. The black nodes are the ones that has already been queried in G_0. LBandit, CBandit and NModel together outputs a node (not already queried in G_0). This node is queried in G_0, and G'_0 is updated. LBandit, CBandit and NModel are then updated with the results of the last query. The different parts involved in MABSample are described in Sections 4.4.1,4.4.2 and 4.4.3.

importance; and the budget is allocated as,

$$b_i = \frac{c_i \cdot \phi\left(G'_i\right) \cdot \lambda\left(G'_i\right)}{\sum_{j \in (0,l]} \left(c_j \cdot \phi\left(G'_j\right) \cdot \lambda\left(G'_j\right)\right)} \cdot (b - b_0)$$

4.3 RNDSample: Sampling $G_{i \in (0,l]}$

In this step, the aim is to sample the layers $G_{(0,l]}$ with the ultimate goal that these samples will be useful in finding good nodes to query in G_0. It has been demonstrated that random walk can be used to discover the community structure of single-layer networks [27, 56]. So, in this step MCS+ uses random walk with jump to separately crawl each of the layers in $G_{(0,l]}$.

Assume that the budget allocated for querying G_+ in this iteration is b, and Q_+ is the set of nodes already queried in layer G_+. Let $G'_+ = \langle V', E'_+ \rangle$ be the sample of G_+ obtained so far. Then, a random walk is started from some node in $V' \setminus Q_+$ such that with probability α it selects an unqueried neighbor of the current node uniformly at random, and with probability $(1 - \alpha)$ it jumps to a node in $V' \setminus Q_+$.

After each query, Q_+ and G'_+ are updated with the newly discovered edges and nodes. The random walk on G_+ continues until budget b has been used up or $V' \setminus Q_+ = \emptyset$. Then, MCS+ moves on to another layer and starts the random walk. When all the layers in $G_{(0,l]}$ have been sampled with the allocated budget for the current iteration, this step ends.

In Section 4.6, we describe how we handle the unreliable query response model.

4.4 MABSample: Sampling G_0

After sampling $G_{i \in (0,l]}$, the next step is to leverage the information from the layers $G_{i \in (0,l]}$ to help in selection of good nodes to query for neighbors in G_0. MCS+ starts by initializing the sample G'_0.

Because we are dealing with a multiplex network, all the nodes that are in in G'_+ are also in G_0. So, we update the set of unqueried nodes $\overline{Q}_0$ as, $\overline{Q}_0 = V' \setminus Q_0$.

As for the edges, all the edges that involve at least one node from Q_0 are already in G'_0.[3] These are the edges directly *observed* from G_0, and denote them by E_α. For edges between $u, v \in \overline{Q}_0$, we cannot be sure if they actually exist in G_0 without querying either u or v in G_0 (thereby putting it in Q_0). In MCS+, we *infer* that this edge exists if it has been observed in a sample G'_+ and the edge overlap of that sample is greater than some threshold, θ_i. Denote the set of all such edges with E_β. Then, G'_0 is initialized as $\langle V', E_\alpha \cup E_\beta \rangle$. Algorithm 1 describes this in more details.

Algorithm 1: Algorithm for MABSample Initialization

Input: $G'_{i \in [0,l]}$
Data: $Q_{i \in [0,l]}$, $\overline{Q}_{i \in [0,l]}$,$\theta_i$
Output: G'_0
$E_\alpha \leftarrow E'_0 \cap (Q_0 \times V')$
$E_\beta \leftarrow \emptyset$
for $i \in (0, l]$ **do**
  if $\lambda(G_i) \geq \theta_i$ **then**
    $E_\beta \leftarrow E_\beta \cup E'_i \cap \left(\overline{Q}_i \times \overline{Q}_i\right)$
  end
end
$G'_0 \leftarrow \langle V', E_\alpha \cup E_\beta \rangle$
return G'_0

Once the initialization is over, MCS+ goes through *Layer Selection* (Section 4.4.1), *Community Selection* (Section 4.4.2) and *Node Selection* (Section 4.4.3). These steps select nodes nodes to query on in G_0 – effectively bringing in new edges and helping to 'clean up' the edges added from E_β.

4.4.1 Layer Selection with LBandit. Different layers will contain varying information about the layer of interest. As an example, consider the NoordinTop Terrorist network (Section 6.1), where Face-to-Face communication is the layer of interest. In this network, the Mentors layer is more likely to contain more information about the layer of interest than the Classmates layer.

Since we do not know these layer correlations before hand, during the *Layer Selection* step, we use a multi-armed bandit with the arms corresponding to $G_{i \in (0,l]}$. We refer to this as the LBandit. Ideally, we want to select the layer that leads to selection of nodes that move the similarity between $K\left(G'_0\right)$ and $K\left(G_0\right)$ closer towards 1. However that requires knowledge of $K\left(G_0\right)$ – which we do not have.

As a result, we use the *Community Update Distance* (Section 4.5) as the reward. Let u be the node selected to query on, then the community update distance is denoted by $\Delta\left(G'_0, u\right)$. If $\Delta\left(G'_0, u\right)$ is very small, the query on u did not help in the 'clean up' of G'_0 or did not bring in edges that are important to the community structure. So, we want to have high $\Delta\left(G'_0, u\right)$ and use this as a proxy for actual comparison with $K\left(G_0\right)$ to assign the reward.

4.4.2 Community Selection with CBandit. After some layer G_i has been selected by LBandit, the next task is to narrow down the potential nodes to query. Depending on the type of multiplex network, different communities in G_i might have different relationships with those in G_0. For

[3]This is not true in the case of *Unreliable Query Response Model*. We will address it in Section 4.6.

example, consider the Twitter Followers and Replies network, and celebrities A and B. If A interacts a lot with her fans but B does not, the community around A in the Replies layer will have more information about the one in Followers network compared to that of B.

This information is not available *a priori*, and so we again have another level of a multi-armed bandits to narrow down on the candidate nodes to query on. In these multi-armed bandits, the communities observed in G'_i are the arms. We call this the `CBandit` and every layer has its own version.

Like in the case of *Layer Selection*, if $\Delta\left(G'_0, u\right)$ is low, querying nodes in the selected community does not yield new information; if it is high, it indicates that there is more uncertainty around this community (in G'_0). So, we want to select more of the communities that have high $\Delta\left(G'_0, u\right)$ and use that as the reward for `CBandit`.

Hierarchical `CBandit`: If there are many communities in G'_i, `CBandit` might require many queries to stabilize to a good selection of arm. To address this, we take advantage of hierarchical community structure [16, 41]. Initially `CBandit` starts with a cut in the hierarchical clustering that gives a small number of communities. At some point, if it is found that one arm has significantly higher rewards, it indicates that it contains useful nodes to query on. It is then split into its children communities if they exist. This scheme makes it so that the arms that are not useful do not take up queries individually.

4.4.3 Node Selection with `NModel`. Let G_i be the layer selected by `LBandit`, and C be the set of candidate nodes returned by `CBandit`. `MCS+` needs to select one from among them to query in G_i. Because we are dealing with *Reliable Query Response*, the effective candidates are $C \cap \overline{Q}_0$. `MCS+`, then, assigns structural features $f(u, G'_i)$ (described later) to all the nodes in $u \in C \cap \overline{Q}_0$ based on the sample found G'_i. We will refer to this step as `NModel`.

Based on the previously queried nodes, `MCS+` trains a linear regression model $\mathcal{M}$ with the observed features of the queried nodes to predict the community update distance. Then, the node u^*, with the highest predicted community update distance is selected and queried in G_0. If the nodes have external attributes associated with them, they can also be used as features for the linear regression model.

$$u^* = \arg\max_{u \in C \setminus Q_0} \mathcal{M}\left(f\left(u, G'_i\right)\right) \tag{1}$$

After u^* is queried in G_0, G'_0 is updated. Then, the community update distance found is used to update $\mathcal{M}$. The rewards for the arms G_i `LBandit` and C in `CBandit` are also assigned at this point. If there is still budget left for this iteration, `MCS+` goes back to *Layer Selection*; otherwise, it goes to *Sampling* G_0.

In our experiments, the features assigned are degree centrality, core number, betweeness centrality, and clustering coefficient. It should be noted that the earlier version of this algorithm described in [39] cannot handle node attributes; however `MCS+` can handle node attributes by incorporating them into the feature vector for the nodes during the node selection. We also experimented different node attributes [27, 29] as features; and the results are provided in Section 6.4.

4.4.4 Termination Condition. `MABSample` will continue as long as the budget used during the current iteration is less than the allocated budget b_0. If the average community update distance is still high after b_0 budget has been used up, the information from the current samples $G'_{i\in(0,l]}$ is still good enough to help in selection of good nodes to query. So, we do not need to terminate `MABSample`; and it will continue until the average community update distance of the last x queries drops below some threshold θ_m.

Thus, MABSample will continue until any of the following conditions are satisfied:

(1) The total cost (of the entire sampling process) has exceeded B, the maximum allowed.
(2) The cost of the current iteration of MABSample is greater than b_0, and the average community update distance of the last x queries is less than θ_m.

4.5 Community Update Distance

As explained earlier, we use the *Community Update Distance* as the reward in MABSample. Consider a sample X^0 of some graph G. After querying for the neighbors of a node u and updating X^0, let X^1 be the resulting sample. Then, we define,

$$\delta_m\left(X^0, X^1\right) = 1 - Similarity\left(K\left(X^0\right), K\left(X^1\right)\right).$$

Here $Similarity\left(\cdot,\cdot\right)$ is any measure of community similarity, normalized between 0 and 1. In our implementations, we use *Normalized Mutual Information* [17].

Although $\delta_m\left(X^0, X^1\right)$ gives us a magnitude of how different $K\left(X^1\right)$ is from $K\left(X^0\right)$, it does not tell us if it is because $K(X^1)$ is getting closer to $K(G)$ or if it is getting farther. Our goal is to assign higher reward if $K(X^1)$ is closer to $K(G)$ than $K(X^0)$ is to $K(G)$ – which is difficult since we do not know $K(G)$. So, to assign this 'direction', we use a heuristic.

Assume that X^{-1} is the sample before X^0, and that $K\left(X^0\right)$ is closer to $K\left(G\right)$, than $K\left(X^{-1}\right)$ is to $K\left(G\right)$. Then we assign direction as,

$$\delta_d\left(X^0, X^1\right) = \begin{cases} +1 & \text{if } \delta_m\left(X^{-1}, X^1\right) > \delta_m\left(X^0, X^1\right) \\ -1 & \text{otherwise} \end{cases}.$$

So, we define the *Community Update Distance* of node u with respect to sample X^0 as,

$$\delta\left(X^0, u\right) = \delta_d\left(X^0, X^1\right) \cdot \delta_m\left(X^0, X^1\right)$$

4.6 Handling Unreliable Query Response Model

In the discussion so far, we have considered only the *Reliable Query Response* model. In this section, we will describe the modifications to account for *Unreliable Query Response*.

Unlike in the case of *Reliable Query Response*, a previously queried node might still have unobserved neighbors. So, we need to estimate the probability that a node u still has unobserved neighbors in G_i.

Let $q_i\left(u\right)$ be the number of times node u has been queried in G_i; and let $r_i\left(u, v\right)$ be the number of times v was present in $N'\left(u, G_i\right)$. We estimate the uncertainty of u as,

$$\hat{\beta}_i\left(u\right) = \frac{1}{\left|N\left(u, G_i'\right)\right|} \sum_{v \in N\left(u, G_i'\right)} \frac{r_i\left(u, v\right)}{q_i\left(u\right)}$$

Now, we can calculate the probability that u still has unobserved neighbors in G_i is given by,

$$\rho_i\left(u\right) = \left(1 - \hat{\beta}_i\left(u\right)\right)^{q_i(u)}$$

A node u is included in the set $\overline{Q}_i$ with probability $\rho_i\left(u\right)$; a node that has never been queried in G_i is always included in $\overline{Q}_i$. This is the only modification to MCS+ to account for Unreliable Query Response. Note that we we do not need to know the distribution of node uncertainty factor.

4.7 Handling Overlapping Communities

During the *Community Selection* step, MCS+ can handle both non-overlapping and overlapping communities. In the case of overlapping communities, there will be overlap between some arms of the community bandit. As described in [58], the problem is still very similar to the traditional multi-armed bandit problem from an algorithmic perspective. So, MCS+ can be used without any modification in the case of overlapping communities as well; and we show the experimental results in Section 6.10.

4.8 Choice of Multi-Armed Bandit Algorithm

For both the *Layer Selection* and *Community Selection*, the multi-armed bandit algorithm we use is ϵ-greedy. In the ϵ-greedy algorithm, with probability ϵ a random arm is selected, otherwise the arm with the best estimated reward is selected. We use ϵ-greedy because of its simplicity and we observe experimentally that it performs as well as the different multi-armed bandit algorithms (Section 6.8).

5 THEORETICAL ANALYSIS OF MCS

In this section we present theoretical analysis of MCS+. In Section 5.1 we preset an analysis of the running time of MCS+ to the network property and budget; and in Section 5.2 we explore the sensitivity of MCS+ to different network properties.

5.1 Running Time of MCS

Here, we discuss the running time of MCS with respect to the budget and other properties of the multiplex network M. Assume that RNDSample and MABSample are performed η times during the entire sampling process[4]. For simplicity, we assume that that the budget B is split evenly during each iteration. Since the total budget is unchanged by this assumption, it should not have an effect on the running time.

We have $b = \frac{B}{\eta}$. Let $\sigma = \sum_{j\in[0,l]} c_j$. For layer G_i, define ζ_i such that,

$$\zeta_i \stackrel{\text{def}}{=} \frac{\phi\left(G_i'\right)\lambda\left(G_i'\right)}{\sum_{j\in(0,l]}\left(c_j\phi\left(G_j'\right)\lambda\left(G_j'\right)\right)}$$

$$\hat{\zeta} = \max_i \zeta_i$$

The budget allocated to sampling G_0 for one iteration is, $b_0 = \frac{c_0 b}{\sigma} = \frac{c_0 B}{\sigma\eta}$, and for $i > 0$, the budget allocated for sampling G_i is,

$$b_i = c_i\zeta_i\left(b - b_0\right) = \frac{c_i\zeta_i\left(\sigma - c_0\right)B}{\sigma\eta} = \frac{c_i\zeta_i\tilde{\sigma}B}{\sigma\eta},$$

where $\tilde{\sigma} = \sigma - c_0$.

Then, the number of nodes queried are,

$$n_0 = \frac{b_0}{c_0} = \frac{B}{\sigma\eta},$$

$$\text{and for } i > 0,\ \ n_i = \frac{b_i}{c_i} = \frac{\zeta_i\tilde{\sigma}B}{\sigma\eta}.$$

[4]In MCS, since the termination condition for MABSample is not dependent on the query budget consumed (for $< B$), η is not a fixed constant. We assume its is fixed for simplicity.

Let, $\hat{n} = \frac{\hat{\zeta}\tilde{\sigma}B}{\sigma\eta}$. The total number of nodes queried during η iterations of RNDSample is

$$N_r = \eta \sum_{i\in(0,l]} n_i \leq \eta \sum_{i\in(0,l]} \hat{n} = \frac{(l-1)\,\tilde{\sigma}\hat{\zeta}B}{\sigma} = \hat{N}_r.$$

If a neighborhood query takes constant time, the running time due to RNDSample is,

$$O\left(\hat{N}_r\right) = O\left(\frac{(l-1)\,\tilde{\sigma}\hat{\zeta}B}{\sigma}\right) \approx O\left(\frac{l\tilde{\sigma}\hat{\zeta}B}{\sigma}\right).$$

If the average degree is d_i, a queried node brings in $(d_i - 1)$ other nodes. So, the number of nodes in G'_i is,

$$n'_i = n_i + (d_i - 1)\,n_i - \epsilon(n_i) = d_i n_i - \epsilon(d_i, n_i), \tag{2}$$

where $\epsilon\,(d_i, n_i)$ accounts for the number of nodes observed repeatedly. It is clear that $\epsilon(d_i, n_i)$ grows with d_i and n_i. If we assume that d_i is low enough (compared to the size of the network) so that $\epsilon(*,*) \approx 0$, we have $n'_i = d_i n_i$.

During the initialization of MABSample, community detection is performed on all layers. Recall that since we are dealing with multiplex networks, if a node is found in a layer, it is added to all the layers. Assume that u is one such node added to G'_i, i.e. node u had been added to G'_i because it was observed in another layer, but not in G'_i. So, u exists in G'_i, but no edges to u are observed. We can safely ignore such nodes during community detection. So the running time due to the initialization in η iterations is,

$$\begin{aligned} O\left(\eta n'_0 \log(n_0) + \eta \sum_{i\in(0,l]} \left(n'_i \log\left(n'_i\right)\right)\right) &= O\left(\frac{d_0 B}{\sigma}\log\left(\frac{d_0 B}{\sigma\eta}\right) + \frac{\tilde{\sigma}B}{\sigma}\sum_{i\in(0,l]}\left(d_i\zeta_i \log\left(\frac{\zeta_i d_i \tilde{\sigma}B}{\sigma\eta}\right)\right)\right) \\ &= O\left(\frac{d_0 B}{\sigma}\log\left(\frac{d_0 B}{\sigma\eta}\right) + \frac{\tilde{\sigma}l\hat{d}\hat{\zeta}B}{\sigma}\log\left(\frac{\hat{\zeta}\hat{d}\tilde{\sigma}B}{\sigma\eta}\right)\right) \end{aligned}$$

where $\hat{d} = \max_{i\in(0,l]} d_i$.

During the node selection process, we need to train a linear regression model η times for all the layers. Assuming the number of node features is e, the running time due to this step is

$$O\left(\eta e^2 \sum_{i\in[0,l]} n'_i\right) = O\left(\eta e^2\left(\frac{d_0 B}{\sigma\eta}\right) + \frac{(l-1)\hat{\zeta}\hat{d}\tilde{\sigma}B}{\sigma\eta}\right) = O\left(\frac{e^2 B}{\sigma}\left(d_0 + l\hat{\zeta}\hat{d}\tilde{\sigma}\right)\right).$$

So, the running time of MCS is,

$$O\left(\frac{d_0 B}{\sigma}\log\left(\frac{d_0 B}{\sigma\eta}\right) + \frac{l\tilde{\sigma}\hat{d}\hat{\zeta}B}{\sigma}\log\left(\frac{\hat{\zeta}\hat{d}\tilde{\sigma}}{\sigma\eta}\right) + \frac{e^2 B}{\sigma}\left(d_0 + l\hat{\zeta}\hat{d}\tilde{\sigma}\right)\right) \tag{3}$$

5.1.1 Running Time with B. From [3], we can observe that (with everything else being constant) the running time of MCS+ with respect to the budget, B, is $O\,(B\log(B))$. So, the running time of MCS+ is super-linear with the budget, but grows very slowly.

5.1.2 Running Time with l. There are various terms in [3] that depends on l. So we need to consider them first. Let,

$$\alpha_\zeta^\top = \max_{j\in(0,l]} \phi(G_i')\lambda(G_i')$$
$$\alpha_\zeta^\perp = \min_{j\in(0,l]} c_i\phi(G_i')\lambda(G_i').$$

If we denote $\alpha_\zeta = \frac{\alpha^\top}{\alpha^\perp}$,

$$\hat{\zeta} \leq \frac{\alpha_\zeta^\top}{\alpha_\zeta^\perp(l-1)} = \frac{\alpha_\zeta}{l-1}.$$

Now, let $\alpha_c^\perp = \min_{i\in(0,l]} c_i$. Then, $\sigma \leq c_0 + \alpha_c^\perp(l-1) = c_0\,(1+\alpha_c(l-1))$ where $\alpha_c = \frac{\alpha_c^\perp}{c_0} \leq 1$, and $\frac{\tilde{\sigma}}{\sigma} < 1$. Then the running time of MCS+ is given by,

$$O\left(\frac{1}{1+\alpha_c(l-1)}\log\frac{1}{1+\alpha_c(l-1)} + \frac{\alpha_\zeta l}{l-1}\log\frac{\alpha_\zeta}{l-1} + \frac{1}{1+\alpha_c(l-1)} + \frac{\alpha_\zeta l}{l-1}\right)$$
$$= O\left(\frac{l}{l-1}\left(1+\log\frac{\alpha_\zeta}{l-1}\right)\right).$$

5.1.3 Running Time with Properties of the Original Network. .

Since we are dealing with samples, the running time of MCS+ is independent of the number of nodes or edges in the original network.

From [3], we can see that the running time depends on the average degree of the original network. Let $d^* = \max_i d_i$. Then, the running time of MCS+ in terms of d^* is given by $O(d^* \log d^*)$.

5.2 Sensitivity to Number of Layers

In this section, we evaluate how the performance of MCS is affected by the number of layers in the multiplex network M. For simplicity, we make some assumptions: (1) the layer importance of all the layers is above the threshold[5] and (2) all the layers (except G_0) have the same query costs.

Given two samples generated with the same algorithm, if one has more nodes, it is likely to have more communities that are representative of the ones in the original graph [46]. On the other hand, if a sample has more edges, the community quality of that sample can be better inferred.

Consider a layer G_i. Because the network is undirected,[6] there will be duplication in the observed edges. The factors that affect the amount of duplication are: (1) density of edges in the layer, and (2) the number of queries already made in G_i. Of these factors, the only variable during the sampling process is the number of queries already made. So, let $o_i\,(x)$ be the expected number of unobserved edges (in G_i') and found after x queries have been made. Then, it is easy to see that, $o_i\,(x) \geq o_i\,(x+k)$ where k is a positive integer. Then, the expected number of edges found after using up the allocated budget is $\sum_{x=0}^{\frac{b_i}{c_i}} o_i(x)$.

Since there are multiple layers being queried (and we assume that the layer importance of all layers is above the threshold), there is a possibility that an edge observed in G_i' had been observed in another sample G_j'. This probability is dependent on the *edge overlap* between G_i and G_j,

$$\omega_{i,j} = \frac{|E_i \cap E_j|}{|E_i|}.$$

[5]If this is not the case for a layer, we can simply ignore that layer.

[6]In the case of a directed network, there well be no edge duplication, and this analysis can be extended easily.

Assuming the layers are sampled in sequence starting from 0, the probability of finding an edge in G'_i not observed in any other layer is $\prod_{j\in[0,i)}\left(1-\omega_{i,j}\right)$.

So, the expected number of 'unique' edges in G'_i, i.e. not observed in other layers, is

$$\left(\sum_{x=0}^{\frac{b_i}{c_i}} o_i(x)\right)\left(\prod_{j\in[0,i)}\left(1-\omega_{i,j}\right)\right).$$

Recall that we are interested in only the layer G_0. So, out of the unique edges in G'_i, the expected number of edges that actually exist in G_0 is,

$$\omega_{i,0}\left(\sum_{x=0}^{\frac{b_i}{c_i}} o_i(x)\right)\left(\prod_{j\in[0,i)}\left(1-\omega_{i,j}\right)\right).$$

So, after all the layers are queried, the expected number of edges in G_0 found is,

$$\mathbb{E}\left[|E'_0|\right] = \sum_{i\in[0,l]}\omega_{i,0}\left(\sum_{x=0}^{\frac{b_i}{c_i}} o_i(x)\right)\left(\prod_{j\in[0,i)}\left(1-\omega_{i,j}\right)\right).$$

Similarly, the expected number of nodes can also be calculated.

Assume that k new layers are added to M, and $\mathbb{E}\left[|\overline{E'_0}|\right]$ is the expected number of edges in the sample obtained from the resulting network. Let $\overline{b_i}$ be the budget allocated for layer G_i. Then, it is easy to show that, $b_i \geq \overline{b_i}$.

$$\begin{aligned}
&\mathbb{E}\left[|E'_0|\right] - \mathbb{E}\left[|\overline{E'_0}|\right] \\
&= \sum_{i\in[0,l]}\omega_{i,0}\left(\sum_{x=\frac{\overline{b_i}}{c_i}+1}^{\frac{b_i}{c_i}} o_i(x)\right)\left(\prod_{j\in[0,i)}\left(1-\omega_{i,j}\right)\right) - \sum_{i\in(l,l+k]}\omega_{i,0}\left(\sum_{x=0}^{\frac{\overline{b_i}}{c_i}} o_i(x)\right)\left(\prod_{j\in[0,i)}\left(1-\omega_{i,j}\right)\right). \qquad (4)
\end{aligned}$$

The term $\prod_{j\in[0,i)}\left(1-\omega_{i,j}\right)$ grows smaller as i grows, and for high enough k it approaches 0. The term $\overline{b_i}$ also decreases with increasing values of k. However, decreasing $\overline{b_i}$ grows the term $\sum_{x=\frac{\overline{b_i}}{c_i}+1}^{\frac{b_i}{c_i}} o_i(x)$ (upto a certain extent).

So, if the number of additional layers increases, at some point $\mathbb{E}\left[|E'_0|\right] \geq \mathbb{E}\left[|\overline{E'_0}|\right]$. That is, if the number of layers is beyond some threshold, the performance of MCS will drop. This threshold can be calculated by using (4).

Similar analysis of the sensitivity of MCS to other factors such as the query cost ratio, edge overlap etc. can be performed.

6 EXPERIMENTAL ANALYSIS OF MCS+

In this section we perform experimental evaluations to compare the performance of MCS to various baselines (Section 6.3.2). In Section 5.1 and 5.2, we present the theoretical analysis of MCS+; and in Section 6.5 and 6.6 we validate these experimentally. In Sections 6.7, we show the comparison between *MultiComSample+* and different type of oracle algorithms – the oracle algorithms are ones where we assume the algorithm has access to information not generally available. We show what

Table 2. Statistics for the different networks used for experiments. The type of network is denoted by OSN (Online Social Network), SN (Social Network), BIO (Biological) and CA (Co-authorship). The number of communities, connected components and modularity (Mod) are for the layer of interest.

Network	Abbr.	Type	Layers	Nodes	Edges	Communities	Modularity	Connected Components
TwitterKP	TK	OSN	4	2.4 K	9.2K	32	0.63	19
TwitterTR	TT	OSN	4	3.0 K	11.5K	36	0.67	13
NoordinTop	NT	SN	5	124	782	11	0.56	5
CKMPhysicians	CP	SN	3	117	669	20	0.51	12
GPIArabidopsis	GA	BIO	4	6.6 K	16.6 K	1523	0.77	1486
GPIHomo	GH	BIO	4	17.8 K	152.2 K	5882	0.48	5562
ArxivNetSci	AN	CA	4	7.2 K	35.5 K	1694	0.84	1635

effect different multi-armed bandit algorithms have on the performance of *MultiComSample+* in Section 6.8. Finally, in Section 6.9 we evaluate the contributions of the different components of MCS+ to the overall performance.

6.1 Datasets

For the experimental evaluation of MCS+ we use networks of various size and from various domains. The statistics of the the networks are given in Table 2. In the table, the type of the network is indicated by OSN (Online Social Network), SN (Social Network), BIO (Biological Networks), and CA (Co-authorship Networks). The description of the different layers are given in Table 8, and Table 9 shows the edge overlap between the different layers to the layer of interest.

Table 2 shows the number of communities, modularity, and connected components for the layer of interest. Due to the non-deterministic nature of the community detection method, the communities found in the layer of interest are not always identical. So, to quantify this we show the maximum similarity in the table. We do this by running community detection 30 times and measuring the similarity between them. While the method presented in this paper is not inherently tied to any particular community detection method, we use the Louvain community detection unless otherwise stated.

The first type of networks we have are the online social networks. The Twitter networks (TwitterKP and TwitterTR) were collected using the Twitter API [71] and they are available publicly[7]. The last Twitter network we have is the TwitterHiggs network.[8] In these networks, the friends-followers networks is considered to be the layer of interest.

We also consider two real-world social network datasets. In the NoordinTop network,[9] the face-to-face communications layer is considered to be those of interest. The Physicians network[8] contains three layers, and we consider the advice layer as the layer of interest.

The third type of networks we use are the biological networks.[8] These networks are protein-protein interaction networks – the nodes represents proteins and the edges are the interactions. The layers denotes the different types of interactions: direct, physical, association and co-localization [60]. In these layers, we consider the direct interaction as the layer of interest.

Finally, we have the co-authorship networks – ArXivNetsci.[8] In this network, the nodes are the authors and the layers are different domains (Physics, Math, Biology, Computer Science). Here, the layer of interest is the Physics network.

[7] http://rickylaishram.com/data/

[8] https://comunelab.fbk.eu/data.php

[9] https://sites.google.com/site/sfeverton18/research/appendix-1

6.2 Experimental Setup

For all of the experiments, unless otherwise stated, we assign a query cost of 1 for each node in the layer of interest and 0.5 for all the other layers.[10] In many real-world networks the cost of querying the other layers are even cheaper compared to the layer of interest. For example, the Twitter API allows for 15 friends-followers request every 15 minute, but allows 1500 for the other layers [65].

We show the total query budget as a fraction of the number of nodes in the multiplex network. Unless otherwise stated, the total budget for each experiment is 0.1 times the number of nodes, except for the NoordinTop and CMKPhysicians where we use 0.25 (these networks are smaller, and 0.1 was too small). All experiments start from the same set of 10 randomly selected nodes, and we run each experiment 10 times.

As for the community detection method, we use the Louvain modularity maximization [9] unless otherwise stated. We selected the Louvain method because of it is very widely used. However, MCS+ is not dependent on the community detection method used, and any suitable one can be used. In Section 6.10 we show experimental results for different community detection methods, including overlapping communities. We use the community found from the layer of interest as the ground truth to compare the community found in the sample.

For the experiments with R-UQR model, we assign the values of β_i^{μ} and β_i^{σ} as 0.5 and 0.1. For H-UQR, we set the uncertainty of the nodes in the highest level of the hierarchy as 0.75 and the lowest is set to 0.25. The uncertainty of those in between are between these values depending on their position in the hierarchical structure.

6.3 Performance Comparison against Baselines

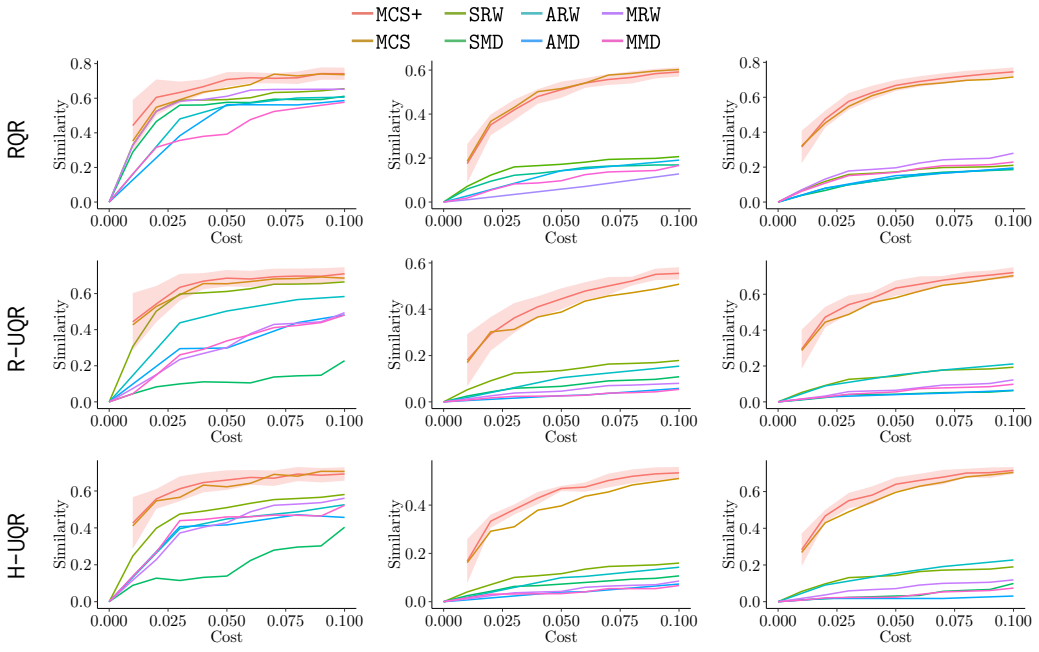
In this section we evaluate the performance of MCS+ against several baseline algorithms (Section 6.3.1) on the networks described in Table 2. For completeness, we also consider an earlier version of the algorithm (called MCS) described in [39]. We consider both types of query response models described in Section 3.1.

6.3.1 Baseline Algorithms. We are not aware of any previous works that address the problem described in this paper (other than our own in [39]). So to evaluate the performance of MCS+ we compare it to three variants each of random walk and Maximum Degree Sampling [46]. Random walk has been shown to be effective at finding community structures in networks [55, 56]; and in [46] the authors demonstrated that samples generated with Maximum Degree sampling are representative of the community structure of a single layer network.

The first set of baselines are aware of only the layer of interest – that is, these algorithms crawl the layer of interest and ignore the rest. Because the true degree of the nodes are not known, in the case of *Single Max Degree* (SMD), the node (from the candidate set) with the highest observed degree is selected as the node to query on. This is equivalent to MOD [5]. In *Single Random Walk* (SRW) a random walk is performed on the layer of interest with reset probability of 0.2. Although these algorithms have been demonstrated to perform well in single layer networks, they suffer from a huge drawback in the case of multiplex networks because there are areas of the layer of interest that can be reached only by traversing through multiple layers.

The second set of baselines operates on an aggregate of all the layers – that is, all the different layers are aggregated into a single layer and *Aggregate Max Degree* (AMD) and *Aggregate Random Walk* (ARW) are performed on the aggregated network. These algorithms do not suffer from the drawbacks above; however, queries are more expensive because nodes are queried in all the layers.

[10] We also performed experiments for other values of query costs. The results follows a similar pattern and are omitted.



(a) TwitterKP (b) ArXiv (c) GPIArabidopsis

Fig. 4. Similarity comparison between MCS+ and other baselines on different networks for different budgets under RQR (Row 1), R-UQR (Row 2) and H-UQR (Row 3). The red line is the performance for MCS+ and yellow line is the result for MCS. It can be observed that both MCS+ and MCS outperform all the baselines. The performance of MCS+ and MCS are close, but MCS+ consistently outperforms MCS. (The shaded area is the standard deviation of MCS+. The standard deviation for the other algorithms are presented in Table 3 but are omitted here for visual clarity.)

In the last set of baselines, the algorithms treat the different layers separately and make use of the multiplex structure. Before a node is selected to be queried on, a layer is first selected. The probability of selecting a layer G_i is proportional to $\frac{\omega_{i,0}}{c_i}$, where $\omega_{i,0}$ is the observed edge overlap of G_i' compared to G_0' and c_i in the cost of a query in G_i. Once the layer is selected, the node with the highest observed degree in that layer is queried in the case of *Multiplex Max Degree* (MMD); and in *Multiplex Random Walk* (MRW), a random neighbor of the current node in the selected layer is queried. These baselines do not suffer from either of the two previous drawbacks.

In the case of Reliable Query Response model, the candidate set consists of only the nodes that have never been queried in the layer of interest. In the case of Unreliable Query Response model, the candidate set consists of queried nodes and queried nodes which returned new neighbors in the last query.

In the case of Reliable Query Response Model, query cost is incurred only if a node that has not been previously queried is queried. In Unreliable Query Response, query cost is incurred for every query.

6.3.2 Results. Figure 4 shows results of MCS+, MCS and other baseline algorithms on TwitterKP, ArXiv and GPIArabidopsis. The first row represents the results for *Reliable Query Response*, the second and third rows are for *Random Unreliable Query Response* and *Hierarchical Unreliable Query Response*, respectively. In these figures, the red lines represent MCS+ and the yellow lines represent

MCS. (The plots for MCS+ and MCS do not start at (0, 0) because these algorithms spend some budget early on to query the cheaper layers.)

We observe that in all the networks under all the query response models, MCS+ outperforms all the baseline methods significantly under all the query response models. If there are a lot of connected components in the layer of interest, we can observe that the baseline algorithms perform very poorly. This is because the crawler might be required to traverse several layers to get to the different connected components, and the baseline algorithms simply fail to reach these components.

Although the performance of MCS is closer to MCS+ than that of the baselines, MCS+ still outperforms it. This indicates that the changes introduced in MCS+ – the freshness during budget allocation and the node selection, improves the performance of the algorithm.

Table 3. Similarity comparison between MCS and other baselines on different networks for different budgets under *Reliable Query Response Model*. The highest value for each network is denoted in bold, and the p-value is denoted by * (< 0.05), **(< 0.01) or ***(< 0.001). It can be observed that MCS+ outperforms all the baselines significantly.

	Query	Proposed		Baselines					
Network	Response	MCS+	MCS	SRW	SMD	ARW	AMD	MRW	MMD
TK	RQR	**0.74 ± 0.03*****	0.73 ± 0.03	0.65 ± 0.02	0.61 ± 0.00	0.60 ± 0.05	0.58 ± 0.00	0.65 ± 0.03	0.57 ± 0.01
	R-UQR	**0.71 ± 0.03*****	0.68 ± 0.04	0.66 ± 0.02	0.22 ± 0.01	0.58 ± 0.02	0.48 ± 0.02	0.49 ± 0.03	0.48 ± 0.03
	H-UQR	**0.69 ± 0.03*****	0.70 ± 0.05	0.62 ± 0.01	0.58 ± 0.00	0.62 ± 0.01	0.18 ± 0.00	0.55 ± 0.07	0.60 ± 0.01
TT	RQR	**0.71 ± 0.03*****	0.69 ± 0.02	0.69 ± 0.02	0.58 ± 0.01	0.59 ± 0.04	0.46 ± 0.01	0.58 ± 0.05	0.48 ± 0.06
	R-UQR	**0.65 ± 0.05*****	0.65 ± 0.02	0.61 ± 0.02	0.36 ± 0.02	0.57 ± 0.03	0.23 ± 0.06	0.49 ± 0.07	0.42 ± 0.07
	H-UQR	0.65 ± 0.01	**0.66 ± 0.03*****	0.53 ± 0.13	0.33 ± 0.03	0.50 ± 0.01	0.42 ± 0.01	0.46 ± 0.04	0.34 ± 0.07
NT	RQR	**0.82 ± 0.01*****	0.74 ± 0.05	0.77 ± 0.07	0.64 ± 0.00	0.60 ± 0.14	0.41 ± 0.00	0.62 ± 0.10	0.36 ± 0.12
	R-UQR	**0.80 ± 0.06*****	0.71 ± 0.05	0.72 ± 0.06	0.38 ± 0.3	0.47 ± 0.19	0.15 ± 0.05	0.16 ± 0.14	0.10 ± 0.12
	H-UQR	**0.78 ± 0.01*****	0.69 ± 0.03	0.71 ± 0.03	0.54 ± 0.02	0.61 ± 0.11	0.20 ± 0.00	0.61 ± 0.10	0.55 ± 0.05
CP	RQR	**0.86 ± 0.02*****	0.79 ± 0.05	0.69 ± 0.04	0.56 ± 0.00	0.67 ± 0.08	0.60 ± 0.00	0.69 ± 0.06	0.48 ± 0.01
	R-UQR	0.76 ± 0.03	**0.77 ± 0.06*****	0.64 ± 0.05	0.27 ± 0.06	0.62 ± 0.07	0.34 ± 0.00	0.21 ± 0.11	0.35 ± 0.11
	H-UQR	**0.77 ± 0.00*****	0.72 ± 0.08	0.62 ± 0.07	0.69 ± 0.00	0.59 ± 0.01	0.41 ± 0.00	0.56 ± 0.03	0.54 ± 0.01
GA	RQR	**0.74 ± 0.02*****	0.71 ± 0.02	0.21 ± 0.01	0.18 ± 0.00	0.19 ± 0.01	0.19 ± 0.00	0.27 ± 0.05	0.23 ± 0.01
	R-UQR	**0.72 ± 0.03*****	0.70 ± 0.02	0.19 ± 0.01	0.06 ± 0.01	0.21 ± 0.02	0.06 ± 0.01	0.12 ± 0.01	0.09 ± 0.01
	H-UQR	**0.71 ± 0.01*****	0.70 ± 0.02	0.18 ± 0.01	0.09 ± 0.00	0.22 ± 0.01	0.03 ± 0.00	0.11 ± 0.00	0.07 ± 0.01
GH	RQR	0.70 ± 0.01	**0.71 ± 0.01*****	0.06 ± 0.00	0.06 ± 0.00	0.10 ± 0.01	0.06 ± 0.00	0.12 ± 0.03	0.09 ± 0.01
	R-UQR	**0.66 ± 0.01*****	0.65 ± 0.02	0.06 ± 0.00	0.04 ± 0.00	0.08 ± 0.00	0.05 ± 0.00	0.05 ± 0.00	0.05 ± 0.00
	H-UQR	**0.68 ± 0.01*****	**0.68 ± 0.01**	0.06 ± 0.00	0.03 ± 0.00	0.08 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.05 ± 0.00
AN	RQR	0.59 ± 0.02	**0.60 ± 0.01*****	0.20 ± 0.00	0.16 ± 0.00	0.19 ± 0.01	0.12 ± 0.00	0.21 ± 0.01	0.16 ± 0.01
	R-UQR	**0.55 ± 0.03*****	0.51 ± 0.02	0.17 ± 0.00	0.10 ± 0.00	0.15 ± 0.01	0.05 ± 0.01	0.08 ± 0.01	0.05 ± 0.01
	H-UQR	**0.53 ± 0.02*****	0.51 ± 0.03	0.15 ± 0.00	0.10 ± 0.00	0.14 ± 0.01	0.07 ± 0.01	0.08 ± 0.01	0.06 ± 0.01

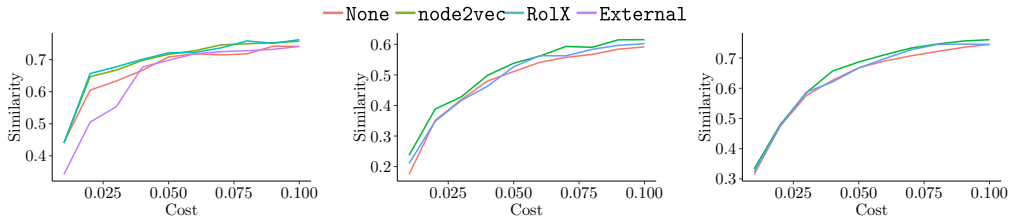
In Table 3, we present the performance comparison between MCS+ and the other algorithms for a budget of 0.1 under different query response models for all the networks described in Section 6.1. These are the results of 10 experiments, and the values shown are the means and standard deviations. The highest value for each network is in bold and * denotes the p-value (compared to the baseline algorithms). Here as well, we observe that MCS+ outperforms all the baselines under all the query response models; and in most of the cases it outperforms MCS as well.

6.4 MCS+ with Node Attributes

In Section 4.4.3, features for the candidate nodes are extracted from the observed sample and are used for node selection. However, if there are node attributes available, they can simply be concatenated to the feature vector of the nodes during the node selection step.

We consider three types of node attributes:

(1) The first one is when the attribute are not related to the network structure. For the TwitterKP network, this feature is a binary vector of the top 20 topics in the texts of the tweets – 1



(a) TwitterKP (b) ArXiv (c) GPIArabidopsis

Fig. 5. Community similarity comparison between MCS without node attributes (red), and with various type of node attributes. Observe that with node attributes the algorithm generally performs well, but the difference is not significantly higher.

denotes that the user (node) has tweeted about the topic and 0 denote otherwise. The other networks do not have these attributes.

(2) For the second attribute, we consider the node embedding through node2vec [27]. The attribute of each node is its vector representation in the embedding.
(3) The last type of attribute we consider is the structural roles of the nodes. We generate this through RolX [29], and the attribute of each node is a vector that defines its structural role in the network.

Note that the node attributes through node2vec and RolX are calculated from the original graph (not the sample), and are returned along with the nodes as results of neighborhood queries. This is intended to emulate cases such as Twitter, where in addition to the user id, the nodes returned to a query includes information such as number of followers, friends, tweets, location, bio, etc.

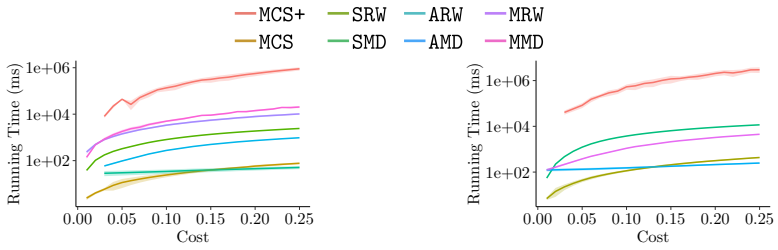
Figure 5 shows the performance of MCS+ with the different attributes for TwitterKP, ArXiv and GPIArabidopsis. We can observe that the performance of MCS+ in the presence of node attributes (especially node2vec and RolX) is better than that without the attributes. However, the results are not significantly different – this might be due to the fact the MCS+, even without the attributes, performs very close to the oracle (Section 6.7).

6.5 Running Time

In this section, we compare the running time of MCS+ against the other algorithms. In most real-world settings, the running time of the algorithm is not a constraint since the query cost is generally tied to some query time; for example in a real-world setting we might be given a query budget of x queries every y time units. However, for completeness, we compare the running time in this section. For the more realistic case where the query budget is tied to some query time, refer to Section 6.11.

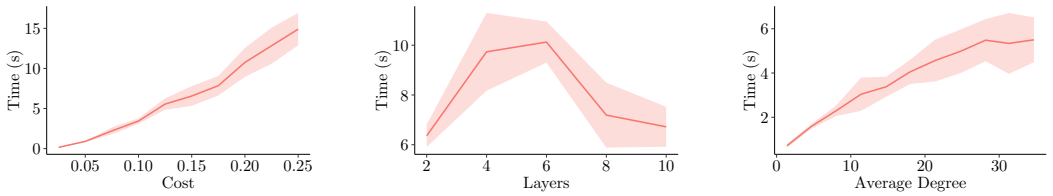
6.5.1 Running Time in Real Networks. In this section, we compare the running time of MCS+ against the baseline algorithms for the TwitterKP and ArXiv networks. Figure 6 shows the running time for the various algorithms for different query costs. As compared to the baselines, MCS+ is the slowest. However, for real-world applications, this is not a limiting factor, since the query cost is the constraint.

6.5.2 Running Time in Synthetic Networks. In Section 5.1, we discussed the theoretical running time of MCS+. In this section, we perform an experimental evaluation of the running time by considering synthetic networks. We consider synthetic networks because we can control for various properties such as number of layers, average degree etc.



(a) TwitterKP (b) ArXiv

Fig. 6. Running time comparison (against the query cost) between MCS+ and the baselines for TwitterKP and ArXiv multiplex networks.



(a) Running time vs budget. (b) Running time vs number of layers. (c) Running time vs average degree.

Fig. 7. Running time analysis of MCS+. Figure 7a, 7b and 7c shows the running time of MCS+ vs budget, number of layers, and average degree respectively.

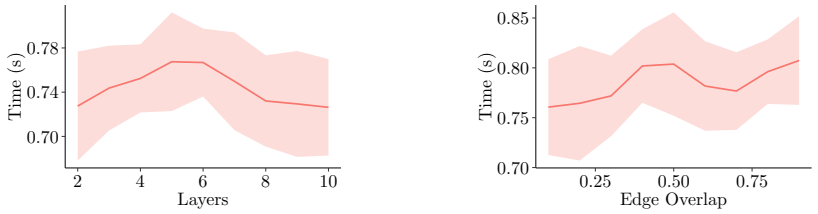
To generate the synthetic networks, we generate a random graph using the planted partition model [15]. (See Table 10 for details of the parameters used.) We generated the other layers such that they have an edge overlap of 0.25 with the layer of interest. To do this, we keep 25% of the edges (randomly selected), and rewire the remaining edges. Table 10 shows the parameters used for generating the synthetic networks.

Table 4. Parameters of the synthetic network generated with planted partition model for experiments.

Parameters	Value	Description
l	5	The number of communities.
k	100	The number of nodes in each community.
p_{in}	0.1	Probability of connection to a node in the same community.
p_{out}	0.01	Probability of connection to a node in other community.

In Figure 7, the running time of MCS+ on these synthetic networks are shown. These are the results of 10 synthetic networks and 10 runs of MCS+ are performed on each graph. For the experiment on running time vs average degree, we change p_{in}, while keeping the ratio $\frac{p_{in}}{p_{out}} = 10$.

In Figure 7a and 7c, we can observe that the running time increases with the budget and average degree. These are consistent with the theoretical observations from Section 5.1. In the case of number of layers, we can see in Figure 7b that the running time increases initially and then decreases as the number of layers increases. The decreasing part is consistent with the theoretical



(a) Sensitivity to number of layers.

(b) Sensitivity to edge overlap.

Fig. 8. Sensitivity of MCS+ to the number of layers (Figure 8a) and edge overlap (Figure 8b). We can observe that the experimental results agrees with what we expected from the theoretical analysis.

analysis. As for the increasing part, this is due to the simplifications we made – for example we assumed that $l - 1 \approx l$ and $\frac{\tilde{\sigma}}{\sigma} \approx 1$.

6.6 Sensitivity Analysis

The theoretical analysis of the sensitivity of the performance of MCS+ to various network properties is discussed in Section 5.2. Here, we provide an experimental analysis of the sensitivity by generating synthetic networks with different properties. The synthetic networks are generated as described in Section 6.5. For all the experiments described below, the total budget is 0.1.

Figure 8a shows the performance of MCS+ against the number of layers. For this experiment, the edge overlap between the layer of interest and the other layers is fixed at 0.25. As expected from the theoretical analysis, the performance of MCS+ increases initially, but after the number of layers crosses some threshold, the performance starts to drop. This is due to the fact that below a certain number of layers, new nodes and edges are observed and contribute to the layer of interest. However, as the number of layers increases, the number of duplicates increases, and budget is wasted on these. Note that the change in performance is not very significant. This is because of the 'freshness' that is taken into consideration during the budget allocation (Section 4.2). As the duplicates observed from a layer increases, it is allocated lower budget in the next iteration.

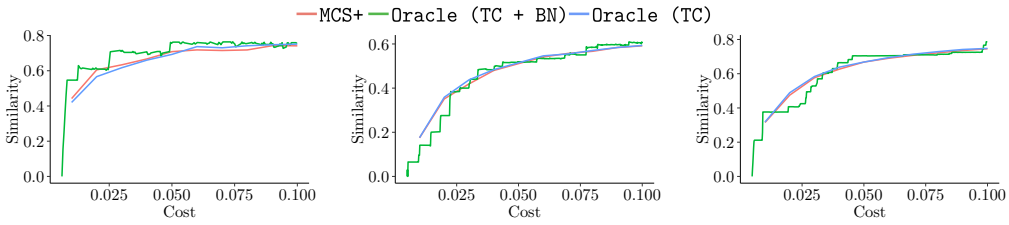
In Figure 8b, the performance of MCS+ for different values of edge overlap is shown. For these experiments, the number of layers is fixed at 2 and the edge overlap is changed. It can be observed that as the edge overlap increases, the performance of MCS+ increases as expected.

6.7 Comparison with Oracles

In this section we show the comparison between MCS+ and different types of oracles. We consider two types of oracles:

(1) In the first oracle, denoted by `Oracle (TC)`, we assume that there is a method to compare the communities in the current sample to the original. That is, instead of the *Community Update Distance* (Section 4.5), the `Oracle (TC)` can actually calculate the reward to be used for the multi-armed bandits. The rest of the algorithm is similar to MCS+.
(2) For the second oracle, denoted by `Oracle (TC + BN)`, we assume that the algorithm can find the node out of the candidates (without incurring any cost) which, if queried and added to the sample, results in the greatest increase in the community similarity. `Oracle (TC + BN)` also has all the capabilities of `Oracle (TC)`.

The comparisons between the oracles and MCS+ is shown in Figure 9 for three networks. In these figures, the red line shows the results for MCS+; the green and blue lines are the results for `Oracle`



(a) TwitterKP (b) ArXiv (c) GPIArabidopsis

Fig. 9. Comparison of the performance of MCS+ to two types of oracles. Figures 9a, 9c and 9c shows the results for TwitterKP, ArXiv and GPIArabidopsis respectively. In these figures, the red, green and blue lines represent the results for MCS+, Oracle (TC + BN) and Oracle (TC) respectively. Observe that the results for MCS+ is very close to the oracles.

Table 5. Performance of MCS+ with different multi-armed bandit algorithms.

	TwitterKP	ArXiv	GPIArabidopsis
ϵ-Greedy [70]	0.74 ± 0.03	0.59 ± 0.02	0.74 ± 0.02
ϵ-Decreasing [13]	0.69 ± 0.04	0.60 ± 0.01	0.72 ± 0.02
VDBE [64]	0.74 ± 0.04	0.60 ± 0.02	0.75 ± 0.02
UCB1 [4]	0.73 ± 0.03	0.59 ± 0.02	0.73 ± 0.03

Table 6. Performance of MCS+ without different steps of MCS+. The results indicates that without either the Layer Selection, Community Selection or Node Selection, the performance of MCS+ degrades. This indicates that all are important and contribute to the overall performance of MCS+.

	TwitterKP	Arxiv	GPIArabidopsis
MCS+	**0.74 ± 0.03**	**0.59 ± 0.02**	**0.74 ± 0.02**
MCS+ without Layer Selection	0.67 ± 0.02	0.57 ± 0.01	0.68 ± 0.01
MCS+ without Community Selection	0.65 ± 0.03	0.59 ± 0.00	0.71 ± 0.05
MCS+ without Node Selection	0.66 ± 0.01	0.56 ± 0.03	0.72 ± 0.02

(TC + BN) and Oracle (TC). We can observe that the results for MCS+ is very close to the that of the oracles.

6.8 Choice of Multi-Armed Bandit Algorithms

As mentioned in Section 4, we use ϵ-greedy as the multi-armed bandit algorithm in MCS+. In Table 5, we show the comparison of MCS+ with different multi-armed bandit algorithms. We compare the results of MCS+ with different multi-armed algorithms – ϵ-greedy [70], ϵ-decreasing [13], VDBE [64] and UCB1 [4]. We can observe that for the different networks considered, the results are very close to each other. So, in MCS+ we use ϵ-greedy due to its simplicity and performance.

6.9 Contribution of Different Steps of MCS+

As described in Section 4, MCS+ consist of multiple steps – layer selection, community selection and node selection. To evaluate the contribution of these steps to MCS+, we consider different versions of MCS+ without each of these steps. That is they are each replaced by random selection. Table 6 shows the results without the different steps (that is when they are replaced by random selection).

Table 7. Results of MCS+ with different community detection methods. In the table, the first column shows the name of the community detection used, the second column is the type of community detected and the rest of the columns are the different networks.

		TwitterKP	ArXiv	GPIArabidopsis
Louvain [9]	Non-Overlapping	0.74 ± 0.03	0.59 ± 0.02	0.74 ± 0.02
Infomap [56]	Non-Overlapping	0.60 ± 0.03	0.48 ± 0.02	0.57 ± 0.01
Infomap [56]	Overlapping	0.90 ± 0.02	0.97 ± 0.00	0.96 ± 0.01
OSLOM [42]	Overlapping	0.81 ± 0.02	0.93 ± 0.01	0.92 ± 0.02

It can be observed that in all the networks considered, removing any of the steps results in substantially worse performance. This indicates that all the steps are necessary for the performance of MCS+.

6.10 MCS+ under Different Community Detection Methods

In the experiments described so far, the community detection method used is the Louvain modularity maximization [9]. However, MCS+ is not dependent on the community detection method and it works for both overlapping, and non-overlapping communities. That is if the original community in G_0 was detected with method X, we can simply swap the Louvain method with X in MCS+.

To show this, in this section we present the results of MCS+ under different community detection methods. In addition to the Louvain modularity maximization [9], we also consider Infomap [10] and OSLOM [42]. The Louvain method detects non-overlapping communities and OSLOM detects overlapping communities. Infomap can detect either types depending on the setting selected by the user.

The original definition of community representation (Section 3.2) holds up for overlapping communities. However to measure the community quality we use the modified NMI as described in [47].

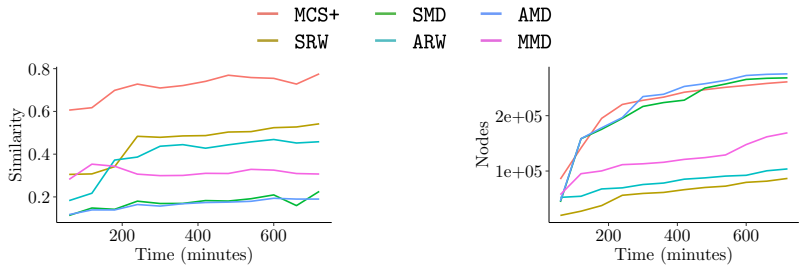
The results for MCS+ under these different community detection algorithm are shown in Table 7; and it can be observed that MCS+ works well with all the different type of community detection algorithms. Note that these results are not directly comparable to each other – that is, the results for Louvain cannot be compared to Infomap, since they measure the similarity to different types communities in G_0.

6.11 Case Study: Twitter API

As mentioned before, when collecting Twitter data, one will generally use the Twitter API. As of the time of writing this paper, this API has a rate limit of 15 queries for every 15 minute window for the *friendship* network, and 75 each for the *mentions*, *retweet* and *reply*.

We perform an experimental comparison between MCS+ and the baseline algorithms by simulating the Twitter API. To do this, we use the Twitter-Higgs network. This network contains 456K nodes and 13M edges. In the layer of interest (friendship) there are 66 communities (as detected by Louvain method) with a modularity of 0.61. To do the simulation, we reduce the window from 15 minutes to 1 minute (this has no effect on the results and serves only to speed up the experiments).

In the Twitter API, queries between different layers cannot be transferred – that is, one cannot save queries for the retweet network for use on the friendship network. Because of this, the budget allocation is always fixed.



(a) Community Similarity. (b) Number of Nodes.

Fig. 10. Results of the case study. Figure 10a shows the similarity between the communities found by the different algorithms; and figure 10b shows the number of nodes in the samples. In both figures, the x-axis is the time in minutes. We can observe that MCS+ gives a sample which has very high community similarity with the original and the number of nodes is also very high. In contrast, among the baselines algorithms, the ones that finds a lot of nodes does not have high community similarity and vice-verse.

Figure 10 shows the results of this case study. In these figures, the x-axis is the total time spent (sum of all the 15 minute windows). In all the experiments, the algorithms always start from the same set of 5 seed nodes. The results shown are the results of 5 runs for each set of seed nodes.

We can see in Figure 10a that the sample that MCS+ generates has the highest community similarity with the original network while also being comparable in terms of the number of nodes with the best baseline. Among the baselines, we can observe that the algorithms that performs well in terms of the community similarity produces samples that have relatively fewer nodes; and those that generate large samples have lower community similarity.

If we compare this result with those in Section 6.3.2, we can observe that the community similarity of the baselines is much closer to MCS+ in our earlier experiments than what we observe in this case study. This is because the budget that we deal with in Section 6.3.2 is much higher than what we deal with in this case study. Specifically, after 720 minutes the total budget consumed is equivalent to 0.0015, while in Section 6.3.2 the total budget consumed is 0.1. So, with the higher budgets in Section 6.10, the baseline algorithms have a better chance to 'catch up.'

7 CONCLUSION

In this paper we address the problem of sampling for the community structure of a layer of interest in a multiplex network under a constraint on the budget that can be used for querying for neighbors of nodes. We consider variations of the problems – when the layers have different exploration (or query) cost and when they have different reliability.

To solve this problem, we proposed MCS+, a novel online sampling algorithm based on multi-armed bandits, which selects the nodes to query in the layer of interest by making use of the information uncovered in the other layers that are less expensive to explore.

We provided theoretical and experimental analysis of the running and sensitivity of MCS+. We also compared the performance of MCS+ against various baseline algorithm on multiple real-world networks from different domains. Experimental evidence shows that MCS+ consistently outperforms the baseline algorithm; in some networks it outperforms the best baseline by around 3 times.

ACKNOWLEDGMENTS

Laishram and Soundarajan are supported by U. S. Army Research Office under grant number #W911NF1810047. Wendt's work is supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525. This research was supported in part through computational resources provided by Syracuse University.

REFERENCES

[1] Serge Abiteboul, Mihai Preda, and Gregory Cobena. 2003. Adaptive On-Line Page Importance Computation. In *Proceedings of the Twelfth International World Wide Web Conference, WWW*. 280–290. https://doi.org/10.1145/775152.775192

[2] Rajendra Akerkar (Ed.). 2011. *Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS 2011, Sogndal, Norway, May 25 - 27, 2011*. ACM.

[3] Katchaguy Areekijseree, Ricky Laishram, and Sucheta Soundarajan. 2018. Guidelines for Online Network Crawling: A Study of Data Collection Approaches and Network Properties. In *Proceedings of the 10th ACM Conference on Web Science, WebSci*. 57–66. https://doi.org/10.1145/3201064.3201066

[4] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 2-3 (2002), 235–256. https://doi.org/10.1023/A:1013689704352

[5] Konstantin Avrachenkov, Prithwish Basu, Giovanni Neglia, Bruno F. Ribeiro, and Donald F. Towsley. 2014. Pay Few, Influence Most: Online Myopic Network Covering. In *Proceedings IEEE INFOCOM Workshops*. 813–818. https://doi.org/10.1109/INFCOMW.2014.6849335

[6] Matteo Barigozzi, Giorgio Fagiolo, and Giuseppe Mangioni. 2011. Identifying the Community Structure of the International-Trade Multi-Network. *Physica A: Statistical Mechanics and its Applications* 390, 11 (2011), 2051 – 2066. https://doi.org/10.1016/j.physa.2011.02.004

[7] Danielle S Bassett and Olaf Sporns. 2017. Network Neuroscience. *Nature Neuroscience* 20 (feb 2017), 353. https://doi.org/10.1038/nn.4502

[8] Michele Berlingerio, Fabio Pinelli, and Francesco Calabrese. 2013. ABACUS: frequent pAttern mining-BAsed Community discovery in mUltidimensional networkS. *Data Min. Knowl. Discov.* 27, 3 (2013), 294–320. https://doi.org/10.1007/s10618-013-0331-0

[9] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (2008), P10008. https://doi.org/10.1088/1742-5468/2008/10/p10008

[10] Ludvig Bohlin, Daniel Edler, Andrea Lancichinetti, and Martin Rosvall. 2014. Community Detection and Visualization of Networks with the Map Equation Framework. In *Measuring Scholarly Impact: Methods and Practice*, Ying Ding, Ronald Rousseau, and Dietmar Wolfram (Eds.). Springer International Publishing, 3–34. https://doi.org/10.1007/978-3-319-10377-8_1

[11] Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks* 30, 1-7 (1998), 107–117. https://doi.org/10.1016/S0169-7552(98)00110-X

[12] Alessio Cardillo, Massimiliano Zanin, Jesús Gómez-Gardeñes, Miguel Romance, Alejandro J. García del Amo, and Stefano Boccaletti. 2013. Modeling the Multi-Layer Nature of the European Air Transport Network: Resilience and Passengers Re-scheduling under Random Failures. *The European Physical Journal Special Topics* 215, 1 (01 Jan 2013), 23–33. https://doi.org/10.1140/epjst/e2013-01712-8

[13] Nicolò Cesa-Bianchi and Paul Fischer. 1998. Finite-Time Regret Bounds for the Multiarmed Bandit Problem. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*. 100–108.

[14] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. 2004. Finding Community Structure in Very Large Networks. *Phys. Rev. E* 70 (Dec 2004), 066111. Issue 6. https://doi.org/10.1103/PhysRevE.70.066111

[15] Anne Condon and Richard M. Karp. 2001. Algorithms for Graph Partitioning on the Planted Partition Model. *Random Struct. Algorithms* 18, 2 (2001), 116–140. https://doi.org/10.1002/1098-2418(200103)18:2%3C116::AID-RSA1001%3E3.0.CO;2-2

[16] Vinicius da Fonseca Vieira, Carolina Ribeiro Xavier, Nelson F. F. Ebecken, and Alexandre G. Evsukoff. 2014. Modularity Based Hierarchical Community Detection in Networks. In *Computational Science and Its Applications - ICCSA - 14th International Conference, Proceedings, Part VI*. 146–160. https://doi.org/10.1007/978-3-319-09153-2_11

[17] Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing Community Structure Identification. *Journal of Statistical Mechanics: Theory and Experiment* 2005, 09 (sep 2005), P09008–P09008. https://doi.org/10.1088/1742-5468/2005/09/p09008

[18] Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. 2015. Identifying Modular Flows on Multilayer Networks Reveals Highly Overlapping Organization in Interconnected Systems. *Phys. Rev. X* 5 (Mar 2015), 011027. Issue 1. https://doi.org/10.1103/PhysRevX.5.011027

[19] Manlio De Domenico, Vincenzo Nicosia, Alexandre Arenas, and Vito Latora. 2015. Structural Reducibility of Multilayer Networks. *Nature Communications* 6 (apr 2015), 6864. https://doi.org/10.1038/ncomms7864

[20] Manlio De Domenico, Albert Solé-Ribalta, Sergio Gómez, and Alex Arenas. 2014. Navigability of interconnected networks under random failures. *Proceedings of the National Academy of Sciences* 111, 23 (2014), 8351–8356. https://doi.org/10.1073/pnas.1318469111

[21] Sean F Everton. 2012. *Disrupting Dark Networks*. Vol. 34. Cambridge University Press.

[22] Ana L. N. Fred and Anil K. Jain. 2003. Robust Data Clustering. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 128–136. https://doi.org/10.1109/CVPR.2003.1211462

[23] Riccardo Gallotti and Marc Barthelemy. 2015. The Multilayer Temporal Network of Public Transport in Great Britain. *Scientific Data* 2 (jan 2015), 140056. https://doi.org/10.1038/sdata.2014.56

[24] Michelle Girvan and Mark EJ Newman. 2002. Community Structure in Social and Biological Networks. *Proceedings of the National Academy of Sciences* 99, 12 (2002), 7821–7826. https://doi.org/10.1073/pnas.122653799

[25] Marko Gosak, Rene Markovič, Jurij Dolenšek, Marjan Slak Rupnik, Marko Marhl, Andraž Stožer, and Matjaž Perc. 2018. Network Science of Biological Systems at Different Scales: A Review. *Physics of Life Reviews* 24 (2018), 118 – 135. https://doi.org/10.1016/j.plrev.2017.11.003

[26] International Crisis Group. [n.d.]. Terrorism in Indonesia: Noordin's Networks. https://www.crisisgroup.org/asia/south-east-asia/indonesia/terrorism-indonesia-noordin-s-networks

[27] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864. https://doi.org/10.1145/2939672.2939754

[28] Wook-Shin Han, Divesh Srivastava, Ge Yu, Hwanjo Yu, and Zi Helen Huang (Eds.). 2010. *Advances in Web Technologies and Applications, Proceedings of the 12th Asia-Pacific Web Conference, APWeb 2010, Busan, Korea, 6-8 April 2010*. IEEE Computer Society. https://ieeexplore.ieee.org/xpl/conhome/5473891/proceeding

[29] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. RolX: Structural Role Extraction & Mining in Large Graphs. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*. 1231–1239. https://doi.org/10.1145/2339530.2339723

[30] Desislava Hristova, Anastasios Noulas, Chloë Brown, Mirco Musolesi, and Cecilia Mascolo. 2016. A Multilayer Approach to multiplexity and link prediction in online geo-social networks. *EPJ Data Sci.* 5, 1 (2016), 24. https://doi.org/10.1140/epjds/s13688-016-0087-z

[31] Christian Hübler, Hans-Peter Kriegel, Karsten M. Borgwardt, and Zoubin Ghahramani. 2008. Metropolis Algorithms for Representative Subgraph Sampling. In *Proceedings of the 8th IEEE International Conference on Data Mining*. 283–292. https://doi.org/10.1109/ICDM.2008.124

[32] Paul Jaccard. 1912. The Distribution of the Flora in the Alpine Zone. *New Phytologist* 11, 2 (1912), 37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

[33] Mahdi Jalili, Yasin Orouskhani, Milad Asgari, Nazanin Alipourfard, and Matjaž Perc. 2017. Link Prediction in Multiplex Online Social Networks. *Royal Society open science* 4, 2 (2017), 160863. https://doi.org/10.1098/rsos.160863

[34] Brian Karrer and Mark EJ Newman. 2011. Stochastic Blockmodels and Community Structure in Networks. *Phys. Rev. E* 83 (Jan 2011), 016107. Issue 1. https://doi.org/10.1103/PhysRevE.83.016107

[35] Michael N. Katehakis and Arthur F. Veinott Jr. 1987. The Multi-Armed Bandit Problem: Decomposition and Computation. *Math. Oper. Res.* 12, 2 (1987), 262–268. https://doi.org/10.1287/moor.12.2.262

[36] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. 2014. Multilayer Networks. *J. Complex Networks* 2, 3 (2014), 203–271. https://doi.org/10.1093/comnet/cnu016

[37] Ludmila I. Kuncheva and Stefan Todorov Hadjitodorov. 2004. Using Diversity in Cluster Ensembles. In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*. 1214–1219. https://doi.org/10.1109/ICSMC.2004.1399790

[38] Ricky Laishram, Katchaguy Areekijseree, and Sucheta Soundarajan. 2017. Predicted Max Degree Sampling: Sampling in Directed Networks to Maximize Node Coverage through Crawling. In *IEEE Conference on Computer Communications Workshops, INFOCOM Workshops*. 940–945. https://doi.org/10.1109/INFCOMW.2017.8116502

[39] Ricky Laishram, Jeremy D. Wendt, and Sucheta Soundarajan. 2019. Crawling the Community Structure of Multiplex Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*. 168–175. https://aaai.org/ojs/index.php/AAAI/article/view/3782

[40] Andrea Lancichinetti and Santo Fortunato. 2011. Limits of Modularity Maximization in Community Detection. *Phys. Rev. E* 84 (Dec 2011), 066122. Issue 6. https://doi.org/10.1103/PhysRevE.84.066122

[41] Andrea Lancichinetti, Santo Fortunato, and Janos Kertesz. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics* 11, 3 (2009), 033015. https://doi.org/10.1088/1367-2630/11/3/033015

[42] Andrea Lancichinetti, Filippo Radicchi, José J Ramasco, and Santo Fortunato. 2011. Finding Statistically Significant Communities in Networks. *PLOS ONE* 6, 4 (04 2011), 1–18. https://doi.org/10.1371/journal.pone.0018961

[43] Kyu-Min Lee, Byungjoon Min, and Kwang-Il Goh. 2015. Towards Real-World Complexity: An Introduction to Multiplex Networks. *The European Physical Journal B* 88, 2 (16 Feb 2015), 48. https://doi.org/10.1140/epjb/e2015-50742-1

[44] Jure Leskovec and Christos Faloutsos. 2006. Sampling from Large Graphs. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 631–636. https://doi.org/10.1145/1150402.1150479

[45] Arun S. Maiya and Tanya Y. Berger-Wolf. 2010. Online Sampling of High Centrality Individuals in Social Networks. In *Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference. Proceedings. Part I.* 91–98. https://doi.org/10.1007/978-3-642-13657-3_12

[46] Arun S. Maiya and Tanya Y. Berger-Wolf. 2010. Sampling Community Structure. In *Proceedings of the 19th International Conference on World Wide Web, WWW*. 701–710. https://doi.org/10.1145/1772690.1772762

[47] Aaron F. McDaid, Derek Greene, and Neil J. Hurley. 2011. Normalized Mutual Information to Evaluate Overlapping Community Finding Algorithms. *CoRR* abs/1110.2515 (2011). arXiv:1110.2515 http://arxiv.org/abs/1110.2515

[48] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology* 27, 1 (2001), 415–444. https://doi.org/10.1146/annurev.soc.27.1.415 arXiv:https://doi.org/10.1146/annurev.soc.27.1.415

[49] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. 2010. Community Structure in Time-Dependent, Multiscale, and Multiplex Networks. *Science* 328, 5980 (2010), 876–878. https://doi.org/10.1126/science.1184819

[50] Mark EJ Newman. 2006. Modularity and Community Structure in Networks. *Proceedings of the National Academy of Sciences* 103, 23 (2006), 8577–8582. https://doi.org/10.1073/pnas.0601602103 arXiv:https://www.pnas.org/content/103/23/8577.full.pdf

[51] Mark EJ Newman and Michelle Girvan. 2004. Finding and Evaluating Community Structure in Networks. *Phys. Rev. E* 69 (Feb 2004), 026113. Issue 2. https://doi.org/10.1103/PhysRevE.69.026113

[52] Elisa Omodei, Manlio De Domenico, and Alex Arenas. 2015. Characterizing Interactions in Online Social Networks during Exceptional Events. *Frontiers in Physics* 3 (2015), 59. https://doi.org/10.3389/fphy.2015.00059

[53] Michael Ovelgönne and Andreas Geyer-Schulz. 2012. An Ensemble Learning Strategy for Graph Clustering. (2012), 187–206. http://www.ams.org/books/conm/588/11701

[54] Tiago P. Peixoto. 2014. Hierarchical Block Structures and High-Resolution Model Selection in Large Networks. *Phys. Rev. X* 4 (Mar 2014), 011047. Issue 1. https://doi.org/10.1103/PhysRevX.4.011047

[55] Pascal Pons and Matthieu Latapy. 2006. Computing Communities in Large Networks Using Random Walks. *J. Graph Algorithms Appl.* 10, 2, 191–218. https://doi.org/10.7155/jgaa.00124

[56] Martin Rosvall and Carl T. Bergstrom. 2008. Maps of Random Walks on Complex Networks Reveal Community Structure. *Proceedings of the National Academy of Sciences* 105, 4 (2008), 1118–1123. https://doi.org/10.1073/pnas.0706851105

[57] Mihaela E Sardiu and Michael P Washburn. 2011. Building Protein-Protein Interaction Networks with Proteomics and Informatics Tools. *Journal of Biological Chemistry* 286, 27 (2011), 23645–23651.

[58] Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. 2019. Overlapping Multi-Bandit Best Arm Identification. In *The IEEE International Symposium on Information Theory (ISIT)*. https://infoscience.epfl.ch/record/265112/files/MultiBandit_FULL.pdf

[59] Martin H Schaefer, Jean-Fred Fontaine, Arunachalam Vinayagam, Pablo Porras, Erich E Wanker, and Miguel A Andrade-Navarro. 2012. HIPPIE: Integrating Protein Interaction Networks with Experiment Based Quality Scores. *PloS one* 7, 2 (2012), e31826.

[60] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. 2006. BioGRID: A General Repository for Interaction Datasets. *Nucleic acids research* 34, suppl_1 (2006), D535–D539.

[61] Michael Szell, Renaud Lambiotte, and Stefan Thurner. 2010. Multirelational Organization of Large-Scale Social Networks in an Online World. *Proceedings of the National Academy of Sciences* 107, 31 (2010), 13636–13641. https://doi.org/10.1073/pnas.1004008107

[62] Lei Tang, Xufei Wang, and Huan Liu. 2009. Uncoverning Groups via Heterogeneous Interaction Analysis. In *ICDM, The Ninth IEEE International Conference on Data Mining*. 503–512. https://doi.org/10.1109/ICDM.2009.20

[63] Bo Tian, Can Zhao, Feiyang Gu, and Zengyou He. 2017. A Two-Step Framework for Inferring Direct Protein-Protein Interaction Network from AP-MS Data. *BMC systems biology* 11, 4 (2017), 82.

[64] Michel Tokic and Günther Palm. 2011. Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax. In *KI : Advances in Artificial Intelligence, 34th Annual German Conference on AI*. 335–346. https://doi.org/10.1007/978-3-642-24455-1_33

[65] Twitter. [n.d.]. API Rate Limits - Twitter Developers. https://dev.twitter.com/rest/public/rate-limiting. Accessed: 2018-05-25.

[66] Lois M Verbrugge. 1979. Multiplexity in Adult Friendships*. *Social Forces* 57, 4 (jun 1979), 1286–1309. https://doi.org/10.1093/sf/57.4.1286

[67] S. Verbrugge, D. Colle, P. Demeester, R. Huelsermann, and M. Jaeger. 2005. General Availability Model for Multilayer Transport Networks. In *DRCN). Proceedings. 5th International Workshop on Design of Reliable Communication Networks.* 8 pp.–. https://doi.org/10.1109/DRCN.2005.1563848

[68] H. Wang, H. Zheng, J. Wang, C. Wang, and F. Wu. 2016. Integrating Omics Data With a Multiplex Network-Based Approach for the Identification of Cancer Subtypes. *IEEE Transactions on NanoBioscience* 15, 4 (June 2016), 335–342. https://doi.org/10.1109/TNB.2016.2556640

[69] Stanley Wasserman and Katherine Faust. 1994. *Social Network Analysis: Methods and Applications*. Cambridge University Press. https://doi.org/10.1017/CBO9780511815478

[70] Christopher JCH Watkins. 1989. Learning from delayed rewards. (1989). http://www.academia.edu/download/50360235/Learning_from_delayed_rewards_20161116-28282-v2pwvq.pdf

[71] J. D. Wendt, R. Wells, R. V. Field, and S. Soundarajan. 2016. On Data Collection, Graph Construction, and Sampling in Twitter. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 985–992. https://doi.org/10.1109/ASONAM.2016.7752360

[72] Jaewon Yang and Jure Leskovec. 2013. Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach. In *Sixth ACM International Conference on Web Search and Data Mining, WSDM*. 587–596. https://doi.org/10.1145/2433396.2433471

[73] Jaewon Yang, Julian J. McAuley, and Jure Leskovec. 2013. Community Detection in Networks with Node Attributes. In *IEEE 13th International Conference on Data Mining*. 1151–1156. https://doi.org/10.1109/ICDM.2013.167

[74] Pan Zhang. 2015. Evaluating Accuracy of Community Detection Using the Relative Normalized Mutual Information. *Journal of Statistical Mechanics: Theory and Experiment* 2015, 11 (2015), P11006. https://doi.org/10.1088/1742-5468/2015/11/P11006

A DATASETS

In this section more detailed description of the datasets used for the experiments is provided. Table 8 shows the descriptions of the different layers of the networks used, and Table 9 shows the number of edges and the edge overlap (compared to the layer of interest) of the different layers.

Table 8. Description of the different layers and the cost of query we assume for our experiments. The layer of interest is denoted in italics.

Network	Layers Description
TwitterKP	*Friends-Followers*, Mentions, Replies, Retweet
TwitterTR	*Friends-Followers*, Mentions, Replies, Retweet
NoordinTop	*Communication*, Classmates, Friendship, Kinship, Mentors
CKMPhysicians	*Advice*, Discussion, Friends
GPIArabidopsis	*Direct Interaction*, Physical Association, Association, Co-localization
GPIHomo	*Direct Interaction*, Physical Association, Association, Co-localization
ArXiv	*Physics*, Math, Biology, Computer Science

B EXPERIMENTAL SETUP

In this section, more details of the experimental setups are provided. In Table 10 the parameters used for generating the synthetic networks with planted partition model is provided.

Table 9. Number of edges in the different layers for different networks.

Network	Number of Edges	Edge Overlap
TwitterKP	7262, 1609, 69, 338	1.0, 0.567, 0.855, 0.814
TwitterTR	9123, 1917, 83, 467	1.0, 0.518, 0.855, 0.702
NoordinTop	311, 205, 153, 49, 64	1.0, 0.2, 0.824, 0.531, 0.781
CKMPhysicians	230, 246, 193	1.0, 0.496, 0.244
GPIArabidopsis	12318, 4154, 48, 141	1.0, 0.144, 0.688, 0.284
GPIHomo	48508, 83404, 1952, 18378	1.0, 0.143, 0.478, 0.074
ArXiv	17349, 4120, 4162, 9961	1.0, 0.553, 0.301, 0.84

Table 10. Parameters of the synthetic network generated with planted partition model for experiments.

Parameters	Value	Description
l	5	The number of communities.
k	100	The number of nodes in each community.
p_{in}	0.1	Probability of connection to a node in the same community.
p_{out}	0.01	Probability of connection to a node in other community.