Clustering Algorithms for Influence Maximisation

Student Name:

Supervisor Name:

Submitted as part of the degree of MEng Computer Science to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract —

Context/Background

Locating the members of a network that have the greatest influence is important when there is a desire to accelerate or suppress the spread of information through it. Doing so optimally is a computationally hard task, creating a need for effective heuristics.

Aims

This project aims to investigate the usefulness of techniques from cluster analysis in their application to influence maximisation. This is done with the goal of identifying which approaches work well based on the analysis of empirical evidence.

Method

The method employed was to implement a range of clustering algorithms for finding multiple influential spreaders and compare their quality of output using an epidemic spreading simulation. The heuristics draw on a range of techniques from network analysis and data mining to find a set of well-distributed, influential nodes.

Results

Testing on a set of real and synthetic networks shows two novel algorithms we contribute, Density-based Graph Clustering and Embedded K-means, to be competitive with existing influence maximisation approaches. To this end, having a good distribution of initial spreaders is shown to be crucial.

Conclusions

Cluster analysis is a useful tool for maximising influence spread. While the adaptation of existing clustering algorithms to networks is a viable approach, learning graph embeddings and then clustering these offers superior performance without increased computational cost.

Keywords — Network analysis, influence maximisation, heuristic clustering, data mining, spreading model

I INTRODUCTION

The study of influence spreading in the context of networks relates to how information is propagated throughout their constituent nodes. In this paper, we focus on finding the network nodes which, when chosen as the starting points for information diffusion, produce the greatest spread according to a given model. This optimisation problem is known as influence maximisation (Kempe et al. 2003). Motivation for finding influential network nodes ranges from identifying which social media accounts to sponsor for viral marketing campaigns, to which locations to quarantine to stop the spread of a disease, to which media outlets to block to stop the spread of fake news. In fact, given sufficient data on any network, the knowledge of the top influential spreaders is invariably useful. As a result of its myriad applications, particularly in relation to social networks, the problem has been widely studied (Castillo et al. 2012).

This project stems from a paper on identifying multiple influential spreaders in networks by a heuristic clustering algorithm, henceforth referred to as HC (Bao et al. 2017). The results of this research are replicated before we take a deeper look into using cluster analysis in influence maximisation heuristics. This includes the implementation and testing of additional algorithms that operate in a network space, as well as a more advanced approach that uses representational learning as a preprocessing step. The rest of this section gives an outline of the topic of influence maximisation and the motivation for our work.

A Background

Formally, given a graph $G = \{V, E\}$ and a parameter k, influence maximisation is defined as finding the set of activator nodes A that maximises some influence function $\sigma(A)$, such that $A \subseteq V$ and $|A| = k$. Kempe et al. (2003) note that the difficulty of finding such a solution is dependent on the spreading model used to calculate σ. In such a model, nodes' states may be described by Boolean variables as either active or inactive and σ often equates to the number of activated nodes. An example of this is the linear spreading model defined by Richardson and Domingos (2002). Here, the likelihood of a node being influenced to buy a certain product (i.e. move from an inactive to an active state) given some marketing action is a weighted sum of its own internal probability and its neighbours' respective values and influence maximisation is solvable in polynomial time. Contrastingly, the problem is NP-hard when using the much more general spreading model previously defined by these authors in which the set of variables denoting nodes' states form a Markov random field (Richardson & Domingos 2001).

While these theoretical models provide convenient definitions of node influence probability, a prescriptive diffusion model that explicitly defines how an agent spreads through a network is desirable when computing node influence experimentally. Our chosen spreading model is the compartmental SIR epidemic model (Kermack & McKendrick 1927), explained in a later section, for which the finding an optimal solution is also NP-hard. Despite its simplicity, this model remains prevalent in the literature on influence maximisation, highlighting the numerous challenges that exist in capturing the complex behaviour of real networks. While the SIR model can be easily extended to incorporate edge weights and directions, other network characteristics are far harder to simulate and account for in maximisation algorithms. For example, the node and edge sets of real social networks such as Facebook and Twitter are dynamic, contain multiple (possibly competing) agents and components, and are so large that global properties become prohibitively expensive to compute, forcing heuristic algorithms to rely solely on local information.

Designing a sophisticated spreading model is one of many complex problems in influence maximisation. As is commonplace with data mining tasks, acquiring a sufficient amount of good quality data is another key issue. Determining what data to collect and devising a suitable method of doing so are problems dependent on the type of network at hand and how influence maximisation is being leveraged. Assimilating these data into a useful model can also be tricky, especially when there are multiple sources to combine or when contextual knowledge of the network must be incorporated. In this project, to provide a fair comparison between all implemented algorithms, we work exclusively with the same undirected, unweighted, static, connected graphs used by Bao et al. (2017). Another challenge in influence maximisation is establishing whether a spread of behaviour across a network is a result of nodes directly influencing their neighbours or similar nodes independently adopting this same behaviour. Correctly factoring in this balance of correlation and causation further increases the complexity of network modelling. Due to these multitudinous issues, real world usage of influence maximisation technology is very much in its infancy (Castillo et al. 2012).

B Motivation

Finding multiple influential spreaders is of much greater practical use than single spreaders. Often, individually influential nodes occur in groups from which repeatedly choosing seeds yields rapidly diminishing returns in terms of the total number of nodes influenced by a simulated epidemic. In such a scenario, selecting less influential nodes favourably positioned in more loosely packed clusters or on bridges between network communities as additional spreaders is usually a more effective strategy. It is therefore established that when selecting multiple sources of information diffusion they should be well-distributed throughout the network. As a result of this need for good network coverage, cluster analysis naturally lends itself to the problem of multiple spreader influence maximisation as a tool for discovering these groups of important nodes so that the knowledge can be funnelled into a mechanism for seed selection.

Algorithms for influence maximisation are diverse, drawing on a wide range of graph theoretic concepts. Those that make use of traditional graph clustering techniques such as finding graph cuts and counting edges to recover modular network structure may be generally categorised by how they define and detect clusters. However, while the HC algorithm does use graph structure to compute the similarity of vertices, it most closely resembles the well-known K-means algorithm for clustering real-valued vectors in terms of how its centroid-based clusters converge. Given the success of HC in finding multiple influential spreaders, we decided to explore this link between network and spatial clustering in more depth in an attempt to further exploit the potential of this class of algorithm for influence spreading in networks.

C Objectives

This paper addresses the research question: *How may cluster analysis be applied to the problem of influence maximisation in networks?* The specific objectives for the project were divided into three stages, all of which were successfully completed. These are as follows:

Minimum: Emulate the implementation and results of Bao et al. (2017). This includes the HC algorithm given by the paper, as well as six other multiple spreader selection algorithms based on node centrality measures and graph colouring. Also, write an SIR epidemic simulation and supporting code for graph manipulation and I/O.

Intermediate: Implement a more sophisticated algorithm using graph clustering. This could perhaps be an adaptation of Highly Connected Subgraphs (HCS) (Hartuv & Shamir 2000) to influence maximisation or a novel algorithm that applies techniques from Density-based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al. 1996) to the problem.

Advanced: Investigate the feasibility of clustering a feature vector representation of a network. This will require a suitable embedding technique and methods for subsequent clustering and seed extraction. Additionally, evaluate the influence spreading ability of all algorithms on a range of test networks and develop appropriate data and network visualisations.

The rest of the paper is organised as follows: section II contains a review of related work on influence maximisation, section III details the solution developed, before results, analysis and final conclusions are presented in sections IV and V.

II RELATED WORK

Many algorithms for finding influential nodes can be found in the literature. These commonly use heuristics to efficiently compute a good approximate solution since it is not computationally feasible to exhaustively check the vast search space encountered when working with common spreading models on a network of reasonable size. In some cases, these algorithms have provable guarantees that the returned solution is within a multiplicative approximation factor of the optimum. But, as is often the case with network analysis, the use of empirical testing as an alternative method of evaluation is commonplace. Each algorithm outputs a set of k network nodes that are the seeds of influence spreading. This section summarises some well-known methods.

A Single Spreader Case ($\mathrm{k} = 1$*)*

The simplest case of the influence maximisation problem is trying to find the best *single* spreader in a network. A group of node properties collectively known as centrality measures often crop up in the literature as tools with which one may reasonably expect to find this activator node. The most common of these are degree, betweenness and closeness centrality, defined in Table 1 (where $\sigma_{tu}(v)$ denotes the number of shortest paths from t to u containing v). The node with greatest centrality according to one of these measures is chosen as the single spreader.

Table 1: Common centrality measures of a vertex v

Degree	Betweenness	Closeness
$\lvert N(v)\rvert$	$\displaystyle\sum_{t\neq u\neq v} \frac{\sigma_{tu}(v)}{\sigma_{tu}}$	$\dfrac{1}{\sum_u d(u,v)}$

Perhaps surprisingly, these well-known centrality measures are often not reliable indicators of influence spreading ability, leading to the development of more sophisticated measures. One notable example is non-backtracking centrality (Radicchi & Castellano 2016), which, under certain conditions, provably finds the best spreader as the node that maximises the sum of the elements of the principal eigenvector of the non-backtracking matrix of the graph that correspond to its outgoing edges. Another technique to find a good single spreader is using the k-shell decomposition (Seidman 1983) of a network, which favours wider network topology over local measures. This algorithm finds maximal induced subgraphs of minimum degree by iteratively pruning the node of lowest remaining degree. Research indicates that the k-shell index of a node returned by this analysis is a better predictor of influence than both degree and betweenness centrality (Kitsak et al. 2010). Even so, this approach has a problem in that the number of shells found is often low relative to the order the graph, giving a wide range of node influence within each shell.

One suggested mitigation for this issue is mixed degree decomposition (Zeng & Zhang 2013), which aims to increase the number of shells. This works in the same way as k-shell decomposition except that nodes are removed according to sum of their remaining degree and an additional ‘exhausted’ degree term that equates to a fraction of the neighbours already removed from that node. Liu et al. (2013) provide an empirically better technique that distinguishes nodes within the same k-shell using the sum of the shortest distances from each to all of the nodes in the innermost shell. Yet another improvement on k-shell centrality is gravity centrality (Ma et al. 2016). Resembling Newton’s law of universal gravitation, this is summed over nearby nodes with k-shell index replacing mass. Coreness centrality, defined as the sum of a node's neighbours' k-shell indices (Bae & Kim 2014), has also been shown to perform better as a predictor of spreading influence than simpler procedures, particularly in networks with a community structure that are scale-free (meaning the degree distribution asymptotically follows a power law).

Other notable node influence metrics shown to be predictive of epidemiological spreading include node accessibility, which is calculated using the probabilities of nearby nodes being reached with a self-avoiding random walk (Travençolo & Costa 2008), and expected force, which is computed from the normalized cluster degrees of all possible transmission clusters resulting from two transmissions from a node (Lawyer 2014). However, none of the aforementioned influence measures have experienced the same success as the PageRank algorithm (Brin & Page 1998) famously designed for ranking pages on the World Wide Web to be indexed and searched by Google. At a high-level, this algorithm assigns a score to each webpage based on its importance by using the quality and number of the incoming and outgoing links to assess the relative probability of a random surfer arriving there directly or by following an upstream link.

Despite their wide coverage in the literature, the measures in this section alone are insufficient when aiming to find multiple influential spreaders; the top k nodes of greatest centrality by any metric are generally poorly distributed throughout the network and tend not to be a good set of activator nodes. Algorithms for the multiple spreader case, which additionally take into account seeds' collective network coverage, are of greater practical interest and are the focus of the rest of this paper. The following section describes some existing approaches.

B Multiple Spreader Case ($\mathrm{k} > 1$*)*

Aside from choosing seeds ranking most highly by centrality, arguably the simplest approach to multiple spreader influence maximisation is greedy hill-climbing whereby the activator node that gives the greatest increase in total influence spread is repeatedly added to the set of seeds until k are chosen. It was proven by Kempe et al. (2003) that this algorithm returns a spreader set within a factor arbitrarily close to $(1 - \frac{1}{e})$ of the global optimum on the condition that the influence function σ is non-negative, monotone and submodular (i.e. gives diminishing returns as the number of seeds increases). However, despite this reasonable guarantee, the greedy algorithm is not a silver bullet due to its $O(|V|^2)$ time complexity, which is generally considered to be too high for use on large-scale networks.

Literature published in subsequent years offered improvements on the hill-climbing approach. By exploiting submodularity to only compute influence spread when necessary, researchers designed the Cost-Effective Lazy Forward selection algorithm (CELF) that is hundreds of times more efficient than the basic greedy algorithm (Leskovec et al. 2007). The implementation, based on the observation that the marginal influence gain of a node can never increase upon the previous round, uses a max-heap to significantly reduce the number of these computations. A further improvement of this algorithm is CELF++, which requires even fewer computations of influence spread by efficiently computing the spread increase yielded by a adding a node to both the current and expected next seed sets (Goyal & Lakshmanan 2011).

As well as reducing the number of spread evaluations, heuristics exist to increase the efficiency of these computations. For example, the Maximum Influence Arborescence (MIA) heuristic (Chen et al. 2010) computes paths of influence to each node and thresholds them such that those along which spreading is unlikely are ignored. The resulting trees offer fast approximation of node influence along the remaining routes. Another heuristic that makes use of local influence regions is SimPath (Goyal et al. 2011). This estimates the marginal influence gain from adding a node to the seed set as the sum of influence spread probabilities of other nodes reachable by simple paths in a subgraph without the current activator set. The initial estimation of node spreading power is made efficient by leveraging a vertex cover of the graph. Additionally, repeated enumeration of paths is avoided using an lookahead optimisation whereby the influence of the current seed set is computed separately to the marginal gain from each one of a given number of promising seed candidates, allowing it to be reused.

The literature also contains techniques to scale up influence maximisation that are pertinent to a range of algorithms. One such approach partitions the network into influence-based communities before applying a maximisation algorithm within each (Wang et al. 2010). The top k seeds are then selected using a dynamic programming approach that greedily mines the top remaining influential node from the partitioned graph. Other research (Mathioudakis et al. 2011) shows network sparsification to be effective. This technique selects edges along which influence is likely to spread to build an induced subgraph of the network using a greedy process that attempts to maximise the combined log-likelihood of a set of observed traces of influence spread. This trimming process reduces the amount of computation performed when subsequently executing an influence maximisation algorithm. Simulated annealing can also be applied to influence maximisation (Jian et al. 2011), whereby an initial set of k seeds is randomly chosen before members are iteratively replaced such that the diffusion quality of the set converges. This approach of introducing nondeterminism guided by heuristics can be generally applied to augment other algorithms' seed selection processes.

More recent literature contains diverse strategies for influence maximisation. One interesting example is the quasilinear algorithm from Morone et al. (2015) that optimally solves the problem, but only on random graphs. This method maps influence maximisation to the problem of optimal percolation, that is, finding the minimal set of nodes to remove from the network to cause it to be fragmented. Here, a measure of the collective influence of a node is used that is proportional to its degree multiplied by the sum of the degrees of nodes on a 'frontier' some radial distance away. The node maximising this quantity is repeatedly calculated and transferred to the seed set until the giant component of the graph vanishes (i.e. when the node count of all connected components is logarithmic in the total number of vertices). The authors show that as the radius of the frontier tends to infinity, the solution better approximates the optimum.

Another approach similar to Wang et al. (2010) utilises the community structure of a network in order to find influential spreaders (Hu et al. 2014). The Girvan-Newman community detection algorithm is used to find a network decomposition of maximal modularity before the node of highest degree from each community is added to the seed set.

While graph clustering algorithms are numerous, those targeted at finding influential spreaders are scarce. Indeed, the HC algorithm from Bao et al. (2017) is not evaluated against any other clustering algorithms. Instead, the graph colouring procedure introduced by Zhao et al. (2015) is used for comparison, in which the chosen seeds form an independent set, guaranteeing a minimum distance of two between any pair of spreaders. A couple of noteworthy algorithms which adapt DBSCAN to graphs, like one of the algorithms we introduce, are DenGraph and SCAN (Schlitter et al. 2014). However, while all three share the same inspiration, they independently give markedly different definitions for the network equivalents of DBSCAN concepts. Furthermore, being the only approach designed for influence maximisation, our algorithm is alone in supplying a mechanism to gather initial spreaders from the clusters formed.

C Solution Analysis

When developing an algorithm for influence maximisation, it is necessary to have a strategy to gauge the influence spreading ability of the returned activator nodes to facilitate a quantitative comparison with other methods. The models used for this task often involve a trade-off between realistic spreading behaviour and computational complexity.

Two generic spreading models that frequently crop up in the literature are the linear threshold and independent cascade models defined by Kempe et al. (2003) that describe the spreading process in terms of discrete timesteps. The linear threshold model assigns each node a threshold in the interval $[0, 1]$, with a node becoming (and remaining) active at a given timestep if the edge-weighted sum of its active neighbours is greater than this value. In the independent cascade model, each edge is assigned a probability $p_{u,v}$ of an inactive node v being influenced by active node u at a given timestep. Solving influence maximisation in both of these models is proven to be NP-hard by reductions to vertex and set cover respectively, and they are both shown to be instances of a more general diffusion model. The independent cascade model in particular bears a resemblance to the SIR model used in this project.

With the influence of a set of activators collected via simulation of a spreading model, a method of comparison is required to judge algorithms' relative success. In the single spreader case, a common procedure is to compare a list of nodes ranked by some centrality measure against a list ordered by spreading influence, defined as the average number of node activations produced when a node is the sole initial spreader. Kendall's tau coefficient (Kendall 1938) is a popular statistic that measures the correlation between two such lists by counting the number of concordant and discordant pairs of nodes with the same index, with a higher value signalling a more effective centrality measure. However, while this method is useful when $k = 1$, it is computationally infeasible to apply it to the exponentially large lists of subsets of nodes in the multiple spreader case, meaning other methods of comparison are used.

In addition to the model of influence spreading used, the networks on which algorithms' performance is measured are also key. These fall into two categories. Real networks are drawn from datasets collected on various real-life systems. While these have the benefit of hopefully being indicative of the networks in which influence maximisation algorithms may be deployed, using them obviously incurs the usual challenges of collecting large amounts of accurate data.

On the other hand, synthetic networks are computer-generated datasets engineered to exhibit some of the characteristics of their real-life equivalents. For example, the scale-free networks generated according to the Barabási-Albert (BA) model (Barabási & Albert 1999) are produced by a preferential attachment process that aims to capture the phenomenon (observed in multiple real networks) of new nodes being more likely to connect to those with higher degree. Another common example is the Watts-Strogatz (WS) model (Watts & Strogatz 1998) that describes graphs with small-world properties, meaning nodes generally have high local clustering coefficients but the average path length is low. Both of these models are used to generate test networks in this project. Also in the literature is the Lancichinetti–Fortunato–Radicchi (LFR) benchmark for creating scale-free networks with irregular community structure (Lancichinetti et al. 2008).

While synthetic networks are convenient for evaluating influence maximisation algorithms due to their freely variable size, they must not be treated as realistic. For example, the BA model does not capture the same levels of clustering observed in real networks and the WS model is not scale-free, giving it an unrealistic degree distribution. The importance of algorithm performance in these models is conditional on their relevance to the domain.

III SOLUTION

This section provides an overview the development work completed during the project. A brief summary of the high-level design choices is given before a more detailed description of the implementation and testing of the clustering algorithms in the solution.

A Development Tools

The primary programming language used to develop the solution was Python (version 3.6.8). It was chosen for its flexibility, support for functional programming and built-in data structures including sets and dictionaries that are useful for manipulating networks. Python 3 also has an extensive ecosystem of open-source third party packages. This project predominantly used the Graph-tool (Peixoto 2014) and Matplotlib packages for network manipulation and data visualisation respectively, with NetworkX and NumPy providing additional functionality. The Stanford Network Analysis Platform (Leskovec & Sosič 2016) was also utilised in the later stages of implementation.

All development work was completed on a personal laptop (Intel i7 6820HK @ 2.7GHz) with code executed using an Ubuntu 18.04 terminal running on the Windows Subsystem for Linux.

B Datasets

As mentioned in section I, the network datasets used in this project are the same as those used by Bao et al. (2017). These are summarised in Table 2. All are connected, with any edge weights and directions removed.

Table 2: Test dataset breakdown

Name	Order, $\|V\|$	Size, $\|E\|$	Average Degree	Diameter	Average Path Length
SMAGRI	1024	4919	9.60	6	2.981
EMAIL	1133	5451	9.62	8	3.606
BLOGS	1222	16714	27.36	8	2.738
HEP	5835	13815	4.74	19	7.026
PGP	10680	24316	4.55	24	7.463
SEX (bipartite)	15810	38540	4.88	17	5.785

Two synthetic networks, both on 2000 nodes, are also used. One is a BA graph with each node preferentially attached to 8 others during generation such that neighbours of higher degree are favoured. The other is a WS small-world graph generated by taking a circle lattice of nodes connected to their 8 nearest neighbours and rewiring each edge with probability 0.1. These parameters were selected to produce networks with a similar average degree and connectivity as the real networks.

This selection of networks is good for generally testing maximisation algorithms as it includes a range of types and sizes. Nevertheless, it is worth noting that if developing an algorithm for a specific application, such as finding which k intersections of a road network to redesign in order to minimize congestion, tuning code based on performance on networks of this type alone will presumably yield superior results.

C Algorithms

This section describes the algorithms implemented during the course of the project. Each takes two arguments, a graph and an integer k, and returns a list of k spreader nodes. The implementations of each are grouped together in a Python module decoupled from other parts of the source code, an approach also taken with the various functions required for graph I/O, SIR epidemic simulation and plotting. This solution architecture was chosen to minimise dependencies between different parts of the codebase, making it easier to develop and test.

C.1 Heuristic Clustering

We begin with a review of the HC algorithm (Bao et al. 2017), which finds centroid-based clusters defined as the set of nodes near to some central node. Defining what constitutes a nearby node is hard. The obvious choice of using the shortest path distance between nodes as a measure of closeness is insufficient because the average shortest path distance is often low even in large networks. As a result, more complex measures are employed to define the similarity between a pair of nodes for clustering. The HC algorithm relies on a metric known as local path (LP) similarity (Lü et al. 2009) that is shown to perform well in predicting missing links in networks (Zhou et al. 2009). This is defined for a pair of vertices, (u, v), using the adjacency matrix, A, of a graph in Equation 1, where the coefficient λ takes a value in the interval $[0, 1]$.

$$LP(u, v) = (A^2)_{uv} + \lambda(A^3)_{uv} \tag{1}$$

As with previous results, no clear trend in the reach of influence spreading was observed when varying this coefficient across the different networks, so we choose a moderate value of $\lambda = 0.5$ in our research. As with the test data, the choice of node similarity measure and any parameters could be tuned for application-specific use.

In our implementation of HC and later algorithms, computing LP similarity by multiplying adjacency matrices is wasteful because only a relatively small fraction of all possible node pairs' values are used during execution. Instead, similarity is computed lazily by directly counting the number of paths of length two and three between a pair of nodes.

Like K-means clustering, HC starts by choosing k random nodes as the initial centroids. These are then iteratively adjusted until convergence, at which point the final centroids are extracted as the set of initial spreaders. During each iteration, k clusters are first formed by assigning nodes to the centroid to which they are most similar based on the LP metric. Within each cluster, the most significant node is then assigned as the new centroid, with significance defined as the sum of the LP similarity of a node with all others in the cluster. Pseudocode for the HC algorithm is given in Algorithm 1.

The HC algorithm given is subtly different to that specified by Bao et al. (2017). When updating the cluster centroids, the original paper defines the significance of a node as a sum over *all* cluster nodes (i.e. including itself), whereas in our implementation self-similarity is ignored. The reasons for omitting this value are that it allowed us to better replicate the original results (see section IV), noticeably decreased the time to convergence and also reduced the bias towards vertices with high degree, since each neighbour of a node corresponds to a path of length two.

In the original paper, HC is compared against six other algorithms also reimplemented as part of our solution. The first three of these, referred to as DC, BC and KC, select spreaders as the set of k nodes with the greatest global degree, betweenness and k-shell centrality respectively. The other three instead retrieve the nodes scoring the highest by these centrality measures from the largest independent set found by the Welsh-Powell colouring algorithm (Xiang-Yu Zhao 2015). These are referred to as DCC, BCC and KSC. It is assumed that the size of the maximal independent set is greater than or equal to k.

Algorithm 1 Heuristic Clustering (HC)

```
Require: Graph $G = \{V, E\}$, number of spreaders $k$
Require: Function LP denoting local-path similarity
 1: function HC($G$, $k$)
 2:     $C \leftarrow \texttt{RANDOM\_CHOICE}(V, k)$                ▷ Initialise k random centroids
 3:     $C_{prev} \leftarrow \varnothing$                          ▷ Initialise set to track previous centroids
 4:     while $C \neq C_{prev}$ do                                 ▷ Until convergence
 5:         for $c \in C$ do                                       ▷ Compute clusters
 6:             $clusters[c] \leftarrow \{v \in V : c = \text{argmax}_u \, \texttt{LP}(u, v),\ u \in C \wedge \texttt{LP}(c, v) > 0\}$
 7:         $C_{prev},\ C \leftarrow C,\ \varnothing$                ▷ Update centroid sets
 8:         for $c \in C_{prev}$ do                                ▷ Compute new centroids
 9:             $S \leftarrow clusters[c]$
10:             $C \leftarrow C \cup \{\text{argmax}_v \sum_{u \in S \setminus \{v\}} \texttt{LP}(v, u) : v \in S\}$
11:     return $C$                                                 ▷ Return influential spreaders
```

The drawback of K-means clustering that the number of clusters found is user-determined and potentially difficult to specify also applies to HC. Additionally, the semi-local nature of LP similarity means that nodes greater than distance three from any centroid are not assigned to clusters. While K-means does not categorise samples as noise, HC does not guarantee that such non-clustered nodes are indeed in parts of the graph that may be considered as the network equivalent. In fact, it is possible that a poor initial random selection of centroids could entirely miss a well-connected part of the graph where influence could easily spread. On a related note, HC activator nodes could sometimes be effectively random if an initial centroid's neighbours are all more similar to others. The next section investigates the potential of using density-based clustering to solve such issues.

C.2 Density-based Clustering

We tested the suitability of the well-known HCS algorithm (Hartuv & Shamir 2000) for use in influence maximisation. Exemplifying some traditional graph clustering techniques, HCS finds clusters by recursively partitioning the graph along the minimum cut to find subgraphs on V nodes with a minimum cut of greater than $\frac{|V|}{2}$ edges.

Unfortunately, HCS frequently partitioned single nodes from the rest of our test networks, producing decompositions of predominantly singleton clusters that were very computationally expensive to obtain. The authors' proposed solution to this behaviour is preprocessing less dense graphs by repeatedly removing all vertices with degree below a threshold, since the size of the minimum cut may not exceed a graph's minimum degree. However, in the context of multiple influential spreader selection this strategy is unreliable as lower centrality does not imply inferior influence spreading ability.

We mention this experiment to highlight the subtle distinction between the densely internally connected communities recovered by many existing graph clustering algorithms and the density-based clusters we wish to obtain for influence maximisation where nodes are in continuous close proximity. The limited work in this area, particularly in relation in influence maximisation, prompted us to develop a novel clustering algorithm, henceforth referred to as DBGC (Density-based Graph Clustering), that applies concepts from the well-known DBSCAN to a network space (Ester et al. 1996). DBSCAN was a natural next step from K-means as a basis for graph clustering due to the original algorithm's success in clustering spatial data. A high-level summary of DBGC, which proceeds in three key stages, is given in Algorithm 2.

Algorithm 2 Density-based Graph Clustering (DBGC)

```
Require: Graph $G = \{V, E\}$, number of spreaders $k$
1: function DGBC($G$, $k$)
2:     $N \leftarrow \{v \in V : \text{CORE\_NODE}(v)\}$          ▷ Collect core nodes
3:     $C \leftarrow \text{CLUSTER}(N)$          ▷ Group the vertex set into clusters
4:     return $\text{EXTRACT}(C, k)$          ▷ Retrieve $k$ seeds from the set of clusters
```

The first stage of DBGC finds core nodes in the graph that are to be grouped into clusters. In DBSCAN, core points are defined as those with at least n nearby points within a radius ϵ. Other clustering algorithms based on DBSCAN compute core nodes in a similar fashion by counting the number of neighbours at different distances d from a vertex. However, we found $d = 1$ was too restrictive as the only potential spreaders were nodes with high degree, and with $d > 1$ the effect a single high degree node had on its neighbours' counts made n difficult to tune. We eventually settled on a more intuitive definition for core nodes based on a measure ψ, defined for a node v in Equation 2, that is computed in the same way as the local clustering coefficient but over the closed neighbourhood of v, $N[v] = N(v) \cup \{v\}$. Specifically, all vertices with $\psi(v)$ at least one standard deviation above the mean are selected as core nodes.

$$\psi(v) = \frac{|\{(u,t) \in E : u,t \in N[v]\}|}{|N[v]| \times |N[v]| - 1} \tag{2}$$

All edges between v and its neighbours are of course present by definition, but including them allows DBGC to handle networks where the local clustering coefficient is zero everywhere, such as trees and bipartite graphs. Equivalently, DBSCAN also includes each data point towards its own count of points within the radius ϵ. During testing, ψ proved to be a reliable indicator of local spreading ability, giving better results than alternative metrics based on common centrality measures when used to select potential seeds.

Next, DBGC groups core points into density-based clusters. The implementation of this step is again similar to DBSCAN, which repeatedly initialises a cluster with an arbitrary core point and continually adds new core points reachable within a distance ϵ of a cluster member until none remain. However, rather than using distance, our graph clustering version adds core nodes to a cluster if their LP similarity with an existing member is above a threshold set as a multiple of the mean ψ value of nodes computed in the previous stage. Scaling the similarity threshold using this measure, which is indicative of the overall density of a network, helps keep the granularity of clusters in particularly dense or sparse graphs more consistent. Nevertheless, as with the corresponding ϵ parameter in DBSCAN, the final clustering is decidedly sensitive to changes in the threshold value.

After clusters of core points are formed, density-reachable borders are added which consist of non-core fringe nodes that are above the similarity threshold with at least one cluster member. Any remaining nodes are classed as noise and will not be selected as initial spreaders.

The final phase of the DBGC algorithm returns k initial spreaders from the clusters obtained, with the number of spreaders, k', extracted from each cluster proportional to its size. Working on the assumption that all clusters are relatively well-connected, achieving a good distribution of activator nodes within each is crucial to maximising the spread of influence.

The seed selection function uses a recursive breadth-first search (BFS) algorithm for this task. Recall that BFS iteratively explores a queue of unexplored nodes and adds any newly encountered neighbours to the queue for the next iteration. Within each cluster, we select our first seed randomly and add it to the queue for the first round of BFS. The cluster is then searched until the size of queue for the next iteration decreases, at which point a new seed is randomly selected from the previous (largest) queue processed. BFS is then called recursively on a queue initialised with the set of seeds chosen so far. This continues until k' spreaders are chosen. If a cluster is not connected and BFS is blocked from progressing (note that a pair of nodes may have high LP similarity without being adjacent), the algorithm resorts to random selection for the next seed. Empirically, this seed selection algorithm performed better than numerous other methods tested using colouring, centrality measures and fully random selection.

C.3 Representational Learning

Both algorithms described so far adapt techniques from spatial clustering to networks. In this section, we take the opposite approach, instead investigating the feasibility of converting graphs to a format that can be directly input to existing algorithms operating on vectors. Such a representation of a graph as a set of real-valued vectors is referred to as an embedding. We propose an algorithm called Embedded K-means (EKM) that is shown, in section IV, to improve upon the previously discussed graph clustering methods for multiple spreader influence maximisation, particularly in dense networks.

In the first step of the EKM algorithm, an embedding of the input network $G = \{V, E\}$ is produced that consists of $|V|$ d-dimensional vectors corresponding to the nodes of the original graph. The spectral embedding of graphs was tested, but we obtained better results the node2vec framework (Grover & Leskovec 2016) which learns continuous feature representations of the graph nodes by optimising a neighbourhood preserving objective. For an embedding $f : V \to \mathbb{R}^d$ of the node set to feature vectors, node2vec uses stochastic gradient descent to optimise the objective function given in Equation 3, where $N_S(u)$ is a neighbourhood of a node u generated via some sampling strategy S.

Two assumptions are made that simplify the calculation of the objective function. The first, shown in Equation 4, is that the probability of observing a node in $N_S(u)$ is conditionally independent of observing any other, given the feature representation of u. The latter is that any two nodes have a symmetric effect on each other in the feature space, meaning each term $P(n_i|f(u))$ may be computed as a softmax function of the dot product of the nodes' feature vectors, shown in Equation 5. A significant strength of node2vec is its use of the negative sampling technique popularised by word2vec (Mikolov et al. 2013) to efficiently approximate the normalising term of Equation 5 in large networks. Rather than computing the dot product of $f(u)$ with the potentially huge number of vectors $f(v)$, a relatively small number of randomly chosen negative samples are used (as few as 5-20 in small networks and 2-5 in larger ones). Clearly, given the size of many real networks, this is critical to node2vec's scalability.

$$\max_f \sum_{u \in V} \log P(N_S(u)|f(u)) \tag{3}$$

$$P(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} P(n_i|f(u)) \tag{4}$$

$$P(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))} \tag{5}$$

The sampling strategy used to determine the neighbourhood, $N_S(u)$, of each node is key to obtaining useful feature representations. Grover & Leskovec (2016) note that two extreme options for S are BFS and depth-first search (DFS), which result in embeddings favouring structural and community knowledge respectively. The actual sampling strategy of node2vec uses biased random walks to interpolate between these two search algorithms. In our results, 20 random walks of length 100 are generated for each vertex in the input graph. Conforming to previous results (Grover & Leskovec 2016), further increasing these parameters gave diminishing returns in terms of the influence of the spreader set returned after clustering the learned features.

Figure 1 shows how return parameter p and in-out parameter q add bias to the random walks. With the walk having moved from node t to v, the edge weights give the (unnormalised) transition probabilities of the next node visited being distance zero, one or two away from t as $\frac{1}{p}$, 1 and $\frac{1}{q}$ respectively. By decreasing the return parameter, p, the walk is more likely to immediately backtrack from v and more closely approximate BFS. Contrarily, by decreasing q, the random walk has a greater chance of moving distance two away from v and exploring more of the network akin to DFS. In our tests, values of $p = 10$ and $q = 0.1$ were used; to maximise influence, it is desirable to have the feature vectors of each cluster corresponding to nodes belonging to the same community in G.

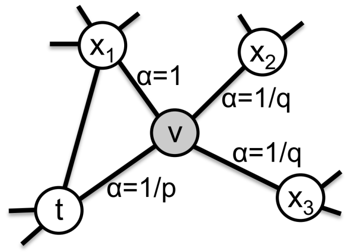

Figure 1: The node2vec random walk procedure (Grover & Leskovec 2016)

A key strength of node2vec is that the transition weight computations, random walk sampling and optimisation steps may all be parallelised, giving the algorithm an average running time that is linear in $|V|$.

After the node feature representations are computed, they are clustered using K-means. We use the scikit-learn implementation (Pedregosa et al. 2011) that employs the K-means++ initialisation scheme to try and prevent the clustering process from converging on a local minima. This works by assigning the starting cluster centroids such that they have a good distribution throughout the dataset, a property established to be critical for a successful multiple spreader heuristic. Specifically, the first centroid, c_1, is chosen randomly from the dataset, $\mathcal{X}$, with each subsequent centroid, c_i, chosen as sample $x' \in \mathcal{X}$ with probability given in Equation 6, where $D(x)$ denotes the shortest distance from x to the closest centroid already assigned. This method is proven to be better than random initialisation (Arthur & Vassilvitskii 2007).

$$P(c_i = x') = \frac{D(x')^2}{\sum_{x \in \mathcal{X}} D(x)^2} \tag{6}$$

The K-means algorithm is executed ten times in parallel to reduce the chance of it failing in local minima. The clustering of lowest inertia (the sum of the squared distances of samples to their closest centroids) is returned. The node in each cluster whose corresponding feature vector minimises the Euclidean distance to the centroid is added to the set of initial spreaders in G.

The average running time of K-means is also linear in $|V|$ (Pedregosa et al. 2011), making the EKM algorithm a practical choice for finding multiple spreaders.

The results of EKM on our test networks presented in section IV were produced with $d = 2^9$. While this is a couple of orders of magnitude above the threshold at which performance saturated in previous tests, we found that it led to higher quality clustering that yielded more influential seeds. Dimension reduction of these vectors using UMAP (McInnes et al. 2018) was tested to try and speed up K-means, but the time saved did not compensate for the additional overhead.

D Testing and Evaluation

The influence spreading ability of the seeds returned by the various influence maximisation algorithms implemented was assessed using the compartmental SIR model (Kermack & McKendrick 1927). In this epidemic model, which proceeds in discrete timesteps on a contact network, nodes take one of three states. ***Infected*** nodes have contracted the disease, which may spread to neighbouring ***susceptible*** nodes with fixed transmission probability, β, such that they, too, become infected. Infected nodes may also move, with a constant recovery probability μ, to a ***recovered*** state in which they are no longer contagious and immune to infection, meaning they can henceforth be ignored. The epidemic ends when no infected nodes remain.

In our simulations, networks are initialised with the activator nodes returned by a maximisation algorithm in an infected state and all others in a susceptible state. To make testing more comprehensive, results from two variants of the epidemic model are presented. In the single-contact SIR model, an infected node has the chance to infect a single randomly chosen neighbour at each timestep, with no guarantee made on the neighbour's state. On the other hand, in the all-contact SIR model, an infected node has the chance to infect all of its neighbours during each timestep. We remark that the mechanics of the spreading model used are another aspect of influence maximisation that could be specialised in real-world use.

The influence of a set of seeds returned by an algorithm A is R_{A}, defined as the number of nodes in a recovered state when the SIR simulation terminates. The same measure, Δ, used by Bao et al. (2017) is employed to compare the relative quality of algorithms with DC as a baseline. Using this metric, given in Equation 7, the correctness of the centrality, colouring and HC implementations is assessed by a direct comparison to previous results. DBGC and EKM are new so we evaluate their performance with respect to these existing approaches.

$$\Delta = \frac{R_{\text{A}} - R_{\text{DC}}}{R_{\text{DC}}} \tag{7}$$

IV RESULTS

In this section, the results of the influence maximisation algorithms running on the networks from section III are presented and discussed. For each network, each algorithm was tested for values of k from ranging from 20 to 200 in intervals of 20, with each data point averaged over 100 executions of the SIR model.

To give a meaningful assessment of the algorithms, transmission and recovery probabilities used for the simulated epidemics were required that avoided instances where infected nodes recovered so quickly as to stifle any spread of influence regardless of the initial spreaders or, conversely, where nodes were so susceptible to infection that any arbitrary set of seeds could influence the majority of a network. We chose the same values previously used to test HC to fairly evaluate our new algorithms against these results. Therefore, the all-contact and single-contact SIR simulations had μ set to 1 and 0.1 respectively and both used a transmission rate of $\beta = 0.2$. Just as with the spreading model, the choice of evaluation parameters such as these is a decision largely dependent on the domain in which influence maximisation is applied.

We first analyse how the relative influence of each of the maximisation algorithms varies with the number of spreaders when evaluated with the all-contact SIR model, shown in Figure 2. On the three smallest real networks (SMAGRI, EMAIL and BLOGS), we see that the spreaders returned by the EKM algorithm are significantly more influential than those returned by other algorithms for all values of k. Here, EKM successfully produces a good distribution of spreaders whose network positions allow them to infect a large number of nodes before enough recover to slow down the spread of influence. These seeds may be located within highly connected areas of the graph which they influence from within or positioned in between multiple communities such that they may influence a number of nodes that are not similar amongst themselves.

In contrast, the HC algorithm achieves the greatest influence in the HEP and PGP networks. While these networks are larger than those on which EKM has the best performance, we observe that EKM and DBGC *are* competitive with HC in the largest network, SEX, which is a bipartite graph connecting buyers to sellers. Therefore, network size does not explain this trend. Instead, we see that these two networks have the greatest diameters and average path lengths of all the

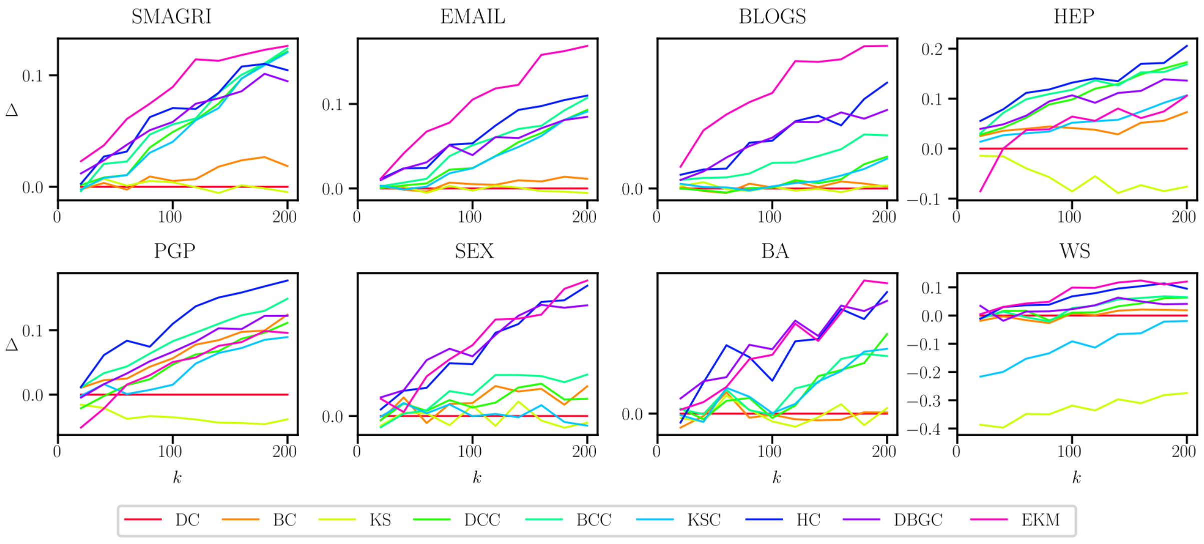

Figure 2: Relative influence against number of seeds for the all-contact SIR model

datasets tested, resulting in low average node degrees that make it less likely for infected nodes to propagate the spread of influence before they quickly recover in the next timestep. We learn from the superior performance of HC in this scenario that knowledge of the exact network structure, which is lost in an embedding, becomes more powerful in maximising influence when an epidemic is less contagious (i.e. when nodes are less likely to adopt their neighbours' behaviour).

This highlights the importance of effective seed extraction from clusters. In HC and EKM, the seed selected from each cluster is the most central node according to the metric already used to define it. In less densely connected networks, HC chooses better initial spreaders as the use of LP similarity guarantees that centroids minimise the number of short paths to other cluster nodes. However, in denser networks, where there are likely to be more inherently influential nodes in each cluster, the EKM algorithm is generally superior to HC due to learned node representations capturing a much wider network information than LP similarity, giving a higher quality network clustering that yields better activator node coverage. A hybrid approach whereby more fine-grained network information is incorporated into the seed selection step of the EKM algorithm in sparse networks is certainly a direction for future work. In the two synthetic networks, BA and WS, EKM and HC again find more influential seed sets than any of the algorithms based on graph colouring and node centrality measures.

Another point of note in this experiment is the variable performance of the DBGC algorithm. On some networks it produces results comparable with HC, whilst in others it is beaten by the colouring algorithms using k-shell and betweenness centrality. We observe that DBGC also surpasses EKM on the HEP and PGP networks, providing more evidence supporting our reasoning that computing directly on a network is advantageous when it is inherently harder for influence to spread. We attribute the inconsistent performance of DBGC to the difficulty of tuning its core node and similarity threshold parameters based on the ψ metric to work well in general across a range of datasets, remarking that similar issues exist with the DBSCAN algorithm upon which it is based. Indeed, DBSCAN was one of the clustering algorithms tested as an alternative to K-means in EKM and was discarded for this reason. While there remains potential in the use of density-based graph clustering methods in certain application domains where they can be calibrated for a specific network type, we conclude that centroid-based methods such as HC and EKM offer a more reliable general-purpose solution.

We next consider the relative influence against k for the single-contact SIR model, shown in Figure 3. The Δ values are larger now that degree centrality is unimportant. As with the all-contact model, EKM outperforms all other algorithms on the smallest three networks and is at least as good on the synthetic graphs. Furthermore, the influence spreading ability of the seeds it obtains from the sparser PGP and HEP networks is now comparable to that of the HC and DBGC graph clustering methods. We reason that this is because the exact seed node selected from a clustered region of a network is less important now that the chance of fast node recovery is lower, despite infected nodes only contacting a single neighbour each round. For example, take an initial spreader of degree d that is assumed not to be adjacent to another seed. Under the all-contact model, the probability of this seed not influencing a neighbour is $(1-\beta)^d$ since it will recover after a single round. But in the single-contact model, if the seed experiences n rounds of infection, the probability of it not influencing a neighbour is $(1-\beta)^n$. With $\mu = 0.1$, each seed in the single-contact model will remain infected for an expected 10 rounds. The average degree of initial spreaders returned by the EKM algorithm in the HEP and PGP networks is less than this value (see Figure 4), meaning $n > d$ for most seeds. This implies that the epidemic is now

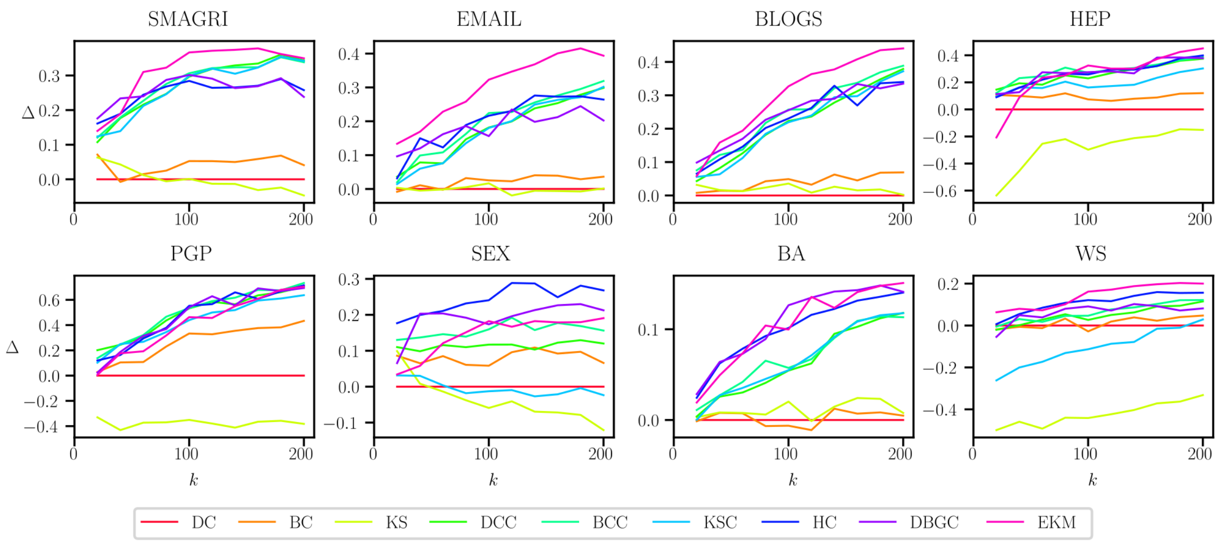

Figure 3: Relative influence against number of seeds for the single-contact SIR model

more contagious in these networks, corroborating our interpretation that clustering embeddings is preferable in such conditions.

In fact, the only network in which EKM is consistently worse than other clustering methods under the single-contact model is SEX. This is possibly due to the node2vec neighbourhood sampling strategy not capturing node information effectively because of the network's bipartite structure, leading to lower quality feature representations. The lack of random walk transitions to nodes adjacent to the previously visited one means random walks have a higher chance of leaving the region of the graph containing the starting node.

Comparing our results to those published by Bao et al. (2017), we see that our HC implementation successfully reproduces the Δ values from previous tests to a satisfactory degree of accuracy, with minor discrepancies due to the nondeterministic elements of the algorithm and evaluation method. The most apparent difference to these previous results is actually in the DCC, BCC and KSC algorithms, which achieve a greater influence spread in our tests with both SIR models. Given that the results of the other algorithms conform and that this is the case for all three colouring algorithms, we suspect that this is due to the implementation of the colouring algorithm used by the authors to gather these previous results.

We conclude our results by examining the notion that the distribution of seeds is more important than their inherent influence in the multiple spreader case. This view is supported by Figures 4 and 5, which show how the median degree of and mean pairwise distance between spreaders vary with k. Unsurprisingly, we see that the basic centrality algorithms generally return spreaders with high median degree as a result of multiple adjacent nodes from highly connected network regions being selected. Median degree for all algorithms naturally decreases with higher k as more low degree spreaders are necessarily chosen. We remark that the seeds chosen by the three clustering algorithms typically have the lowest median degree, despite collectively being the most influential. The only network exhibiting a moderately different trend is the synthetic WS graph, as the model yields a lower range of node degree values and has no mechanism by which high degree nodes are likely to be adjacent.

Similarly, we observe that EKM is the only algorithm that locates seeds with mean pairwise distance greater than the network average path length on all networks for all values of k, further evidencing its excellent clustering ability. Considering that many of these seeds have low degree, this reinforces the conclusion that obtaining an even distribution of multiple initial spreaders with respect to network structure is key to maximising their influence.

Overall, the results underline the importance of the broader network information present in the node embeddings during clustering. While the LP measure used in both DBGC and HC treats all nodes greater than distance three away from a node as having zero similarity in the interest of limiting computational cost, the length of the efficient random walks performed by node2vec is much greater than the diameter of any of the test networks. As a result, a meaningful distance (treated as the inverse of similarity) between *any* pair of nodes may be computed when forming EKM clusters. Discordantly, the semi-local nature of LP similarity means that while it may good for intra-cluster positioning of centroids, it is restricted in its capacity to position these clusters with regard to the global network structure.

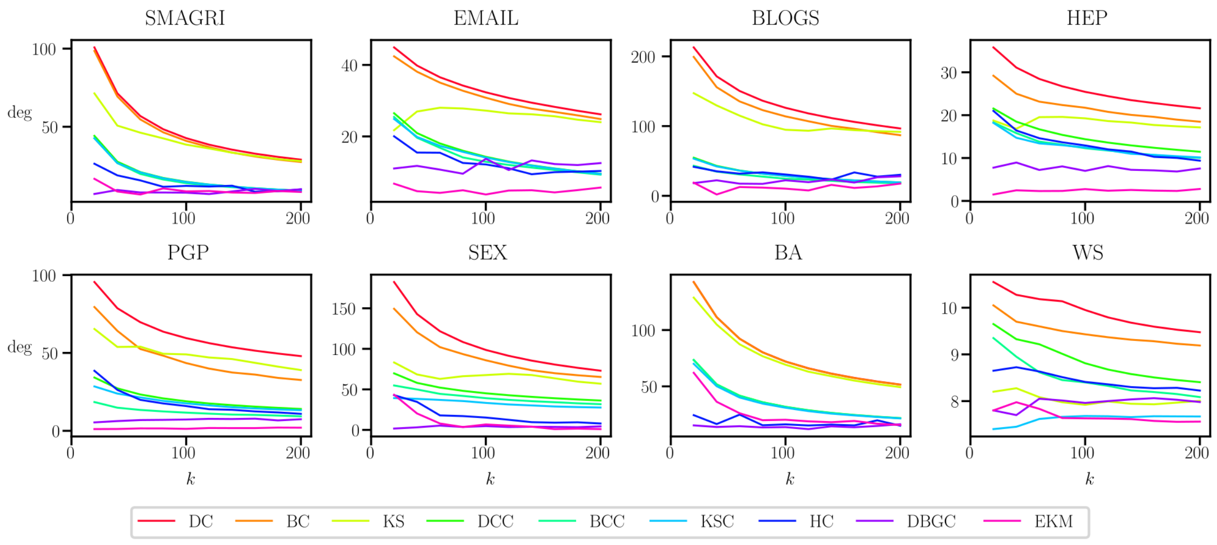

Figure 4: Median seed degree against number of seeds

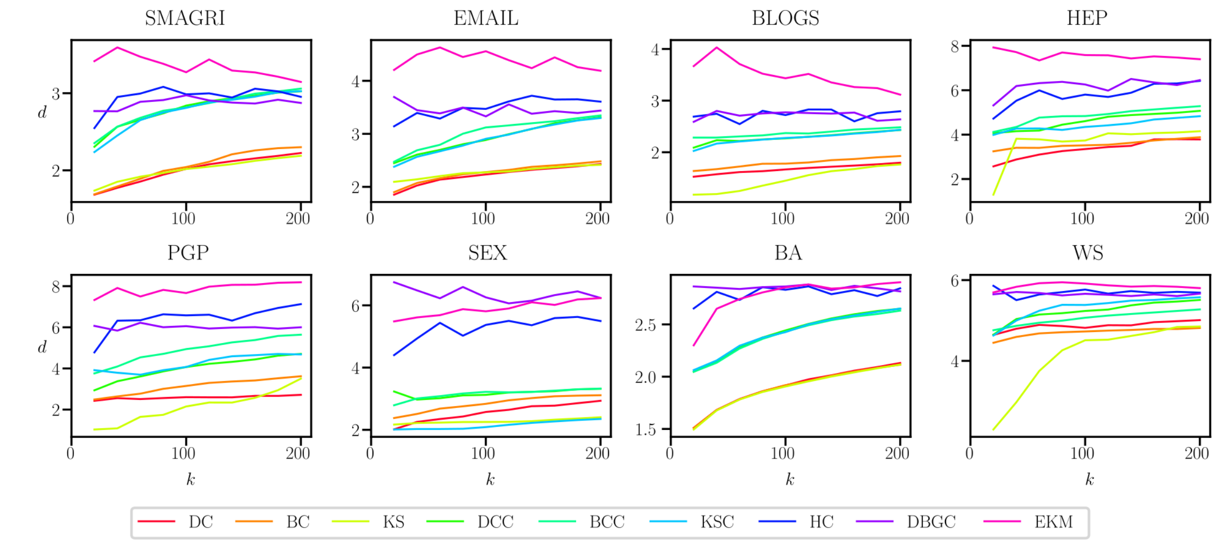

Figure 5: Mean distance between seeds against number of seeds

V CONCLUSIONS

In this project, we have successfully answered the research question of how cluster analysis can be applied to the problem of influence maximisation, extending previous work by demonstrating that density-based and graph embedding clustering approaches are effective on a range of real and synthetic networks. From the evaluation of a suite of implemented algorithms, we have learnt that a good distribution of seeds in the multiple spreader case is of greater importance than their individual influence, and that an informative node similarity metric is instrumental in finding a clustering of nodes from which this can be obtained. One very promising choice is the distance between node feature representations in an embedding of a network. Still, knowledge of precise node adjacency remains valuable, particularly in sparser networks where it is inherently more difficult for influence to spread.

One key contribution is DBGC, which applies concepts from DBSCAN to the network domain and illustrates that the adaptation of spatial clustering algorithms to graphs is a viable strategy for influence maximisation. Another is the EKM algorithm, which combines the node2vec graph embedding procedure with traditional K-means clustering and surpasses graph clustering approaches on several networks. Initial test results are consistent with the assertion that the learned node representations offer good predictive power in downstream tasks (Leskovec & Sosič 2016) and indicate that there is significant potential in the application of embedded clustering to the problem of multiple spreader influence maximisation.

In addition to the inclusion of connectivity information into the seed selection stage of EKM discussed in section IV, possible directions for future work include the development of a graph embedding algorithm that is specialised for influence maximisation and research into new node similarity measures that consider broader network information.

References

Arthur, D. & Vassilvitskii, S. (2007), K-means++: The advantages of careful seeding, *in* 'Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms', SODA '07, Society for Industrial and Applied Mathematics, USA, p. 1027–1035.

Bae, J. & Kim, S. (2014), 'Identifying and ranking influential spreaders in complex networks by neighborhood coreness', *Physica A* **395**(1), 549–559.

Bao, Z.-K. et al. (2017), 'Identifying multiple influential spreaders by a heuristic clustering algorithm', *Physics Letters A* **381**(11), 976–983.

Barabási, A.-L. & Albert, R. (1999), 'Emergence of scaling in random networks', *Science* **286**(5439), 509–512.

Brin, S. & Page, L. (1998), 'The anatomy of a large-scale hypertextual web search engine', *Computer Networks and ISDN Systems* **30**(1-7), 107–117.

Castillo, C., Chen, W. & Lakshmanan, L. V. (2012), 'Tutorial: Information and influence spread in social networks'. [online] Available at: https://www.microsoft.com/en-us/research/event/kdd2012-tutorial-information-influence-spread-social-networks/ (Accessed: 3 January 2020).

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

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise, *in* 'KDD'96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining', pp. 226–231.

Goyal, A. & Lakshmanan, L. (2011), Celf++: Optimizing the greedy algorithm for influence maximization in social networks, *in* 'Proceedings of the 20th International Conference Companion on World Wide Web', pp. 47–48.

Goyal, A., Lu, W. & Lakshmanan, L. (2011), Simpath: An efficient algorithm for influence maximization under the linear threshold model, *in* '2011 11th IEEE International Conference on Data Mining', pp. 211–220.

Grover, A. & Leskovec, J. (2016), node2vec: Scalable feature learning for networks, *in* 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'.

Hartuv, E. & Shamir, R. (2000), 'A clustering algorithm based on graph connectivity', *Information Processing Letters* **76**(4), 175–181.

Hu, Z.-L., Ren, Z.-M., Yang, G.-Y. & Liu, J.-G. (2014), 'Effects of multiple spreaders in community networks', *International Journal of Modern Physics C* **25**(5), 1440013.

Jian, Q., Song, G., Cong, G., Wang, Y., Si, W. & Xie, K. (2011), Simulated annealing based influence maximization in social networks, *in* 'Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence', pp. 127–132.

Kempe, D., Kleinberg, J. & Tardos, É. (2003), Maximizing the spread of influence through a social network, *in* 'Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 137–146.

Kendall, M. (1938), 'A new measure of rank correlation', *Biometrika* pp. 81–93.

Kermack, W. & McKendrick, A. (1927), A contribution to the mathematical theory of epidemics, *in* 'Proceedings Mathematical Physical and Engineering Sciences', pp. 700–721.

Kitsak, M., Gallos, L., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H. & Makse, H. (2010), 'Identification of influential spreaders in complex networks', *Nature Physics* **6**(11), 888–893.

Lancichinetti, A., Fortunato, S. & Radicchi, F. (2008), 'Benchmark graphs for testing community detection algorithms', *Physical Review E* **78**(4 pt.2), 046110.

Lawyer, G. (2014), Understanding the spreading power of all nodes in a network: a continuous-time perspective, *in* 'Proceedings of the National Academy of Sciences of the United States of America', Vol. 9, pp. 1–10.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J. & Glance, N. (2007), Cost-effective outbreak detection in networks, *in* 'Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 420–429.

Leskovec, J. & Sosič, R. (2016), 'Snap: A general-purpose network analysis and graph-mining library', *ACM Transactions on Intelligent Systems and Technology (TIST)* **8**(1), 1.

Liu, J.-G., Ren, Z.-M. & Guo, Q. (2013), 'Ranking the spreading influence in complex networks', *Physica A* **392**(18), 4154–4159.

Lü, L., Jin, C.-H. & Zhou, T. (2009), 'Similarity index based on local paths for link prediction of complex networks.', *Physical review. E, Statistical, nonlinear, and soft matter physics* **80 4 Pt 2**, 046122.

Ma, L.-L., Ma, C., Zhang, H.-F. & Wang, B.-H. (2016), 'Identifying influential spreaders in complex networks based on gravity formula', *Physica A* **451**(1), 205–212.

Mathioudakis, M., Bonchi, F., Castillo, C., Gionis, A. & Ukkonen, A. (2011), Sparsification of influence networks, *in* 'Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 529–537.

McInnes, L., Healy, J. & Melville, J. (2018), 'UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction', *ArXiv e-prints* .

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013), 'Distributed representations of words and phrases and their compositionality', *Advances in Neural Information Processing Systems* **26**.

Morone, F. & Makse, H. (2015), 'Influence maximization in complex networks through optimal percolation', *Nature* **524**(1), 65–68.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Peixoto, T. P. (2014), 'The graph-tool python library', *figshare* . [online] Available at: http://figshare.com/articles/graph_tool/1164194 (Accessed 8 July 2019).

Radicchi, F. & Castellano, C. (2016), 'Leveraging percolation theory to single out influential spreaders in networks', *Physical Review E* **93**(6).

Richardson, M. & Domingos, P. (2001), Mining the network value of customers, *in* 'Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 57–66.

Richardson, M. & Domingos, P. (2002), Mining knowledge-sharing sites for viral marketing, *in* 'Proceedings of the eigth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 61–70.

Schlitter, N., Falkowski, T. & Lässig, J. (2014), 'Dengraph-ho: a density-based hierarchical graph clustering algorithm', *Expert Systems* **31**(5), 469–479.

Seidman, S. (1983), 'Network structure and minimum degree', *Social Networks* **5**(3), 269–287.

Travençolo, B. & Costa, L. F. (2008), 'Accessibility in complex networks', *Physics Letters A* **373**(1), 89–95.

Wang, Y., Cong, G., Song, G. & Xie, K. (2010), Community-based greedy algorithm for mining top-k influential nodes in mobile social networks, *in* 'Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 1039–1048.

Watts, D. & Strogatz, S. (1998), 'Collective dynamics of 'small-world' networks', *Nature* **393**(6684), 440–442.

Xiang-Yu Zhao, Bin Huang, M. T. H.-F. Z. D.-B. C. (2015), 'Identifying effective multiple spreaders by coloring complex networks', *Europhysics Letters* **108**(6), 68005.

Zeng, A. & Zhang, C.-J. (2013), 'Ranking spreaders by decomposing complex networks', *Physics Letters A* **377**(14), 1031–1035.

Zhou, T., Lü, L. & Zhang, Y.-C. (2009), 'Predicting missing links via local information', *The European Physical Journal B* **71**(4), 623–630.