Using Spectral Clustering to Find Successful Hero Combinations in DOTA 2

Roderick Cox

Faculty of Mathematical Studies, Southampton

November 2015

Community detection is a branch of mathematics that is becoming increasingly relevant as the processing powers of computing equipment grows and measurable data sets become larger. Communities within large data sets are often indiscernible without use of community detection algorithms. This report sets out to demonstrate and explain the use of spectral clustering with hierarchical clustering methods, upon data from an extensively played multiplayer online game, DOTA 2. The aim of using this technique is to partition the data in order to identify strong clusters of 5 playable heroes in the game who have a high overall success rate, due to the advantage gained from their complementary abilities.

1. Introduction

Community detection is a relatively new but increasingly relevant branch of mathematics. In a world where enormous data sets are able to be compiled, analyzing and responding to these datasets effectively is an essential task for scientists, businesses and politicians. The overused but nonetheless significant example of social media is a very relevant case where we can see large data sets being compiled and mathematically analysed to identify communities. Similarly in science, protein groups, hierarchical communities of species and interaction between cells are all poignant examples of how community detection algorithms can be used to gain vital knowledge of communities in the real world.

This report focuses community detection techniques upon the Valve's "DOTA 2", a multiplayer online game in which 2 teams of 5 choose from a pool of 110 unique playable characters (named in the game and in this report as "heroes" and compete against one another, often professionally where large sums of money are at stake. Choosing an effective team whose abilities complement each other is a strategy contributing strongly towards a team's chance of success; this is as much a community detection problem as the aforementioned examples. The communities that we wish to find are clusters of 5 heroes whose complementary abilities make them a successful team.

This report sets out to use the mathematical techniques of spectral clustering together with the k-means algorithm to determine clusters of heroes who should be played together. From these pools of heroes, through inspection of the results from hierarchical clustering, strong combinations of 5 heroes can be identified. This report will explain the intuition behind these methods, as well including a worked example with a subset of the sampled data. It will explain the method used to sample the data and how it corresponds to a community detection problem. It will assess the results of the method before improving upon the result through adjusting the data set accordingly to enhance the method. Finally, it will explain the results in context and discuss alternative strategies in solving this community detection problem

2. Preliminary Terminology and Equations

A **graph** *G= (V,E)* where *V* is the set of **vertices** (or **nodes)** {$v_1,\ldots,v_n$} and *E* represents the set of edges connecting the vertices $v_1 \ldots v_n$. The **weight** w_{ij} of an edge connecting vertices v_i, v_j represents the value of the similarity between those vertices.. If $w_{ij} = 0$ then the vertices v_i, v_j are not connected.

The **degree** d_i of a vertex $v_i \in V$ is the sum of the weights of all the edges connected to v_i. Therefore

$$d_i = \sum_{j=1}^{n} w_{ij} \quad [1]$$

The weighted **adjacency matrix** of a graph is the matrix

$$W = (w_{ij})_{i,j=1,\ldots n} \quad [2]$$

The **degree matrix** *D* is the diagonal matrix with the degrees $d_i,\ldots,dn$ in its diagonal entries.

$$D = \begin{pmatrix} d_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_n \end{pmatrix} \quad ` \quad [3]$$

A **cut** of a subgraph C of a graph G is defined as

$$cut(A) = \sum_{i \in A, j \notin \mathbb{A}} w_{ij} \quad [4]$$

The **conductance** Φ(C) of the subgraph C of a graph G is the connectivity of a group to the whole network relative to the group density, and is defined as

$$\Phi(C) = \frac{c(C, G \backslash C)}{\min(k_c, k_{G \backslash C})} \quad [5]$$

Where $c(C, G\backslash C)$ is the cut is size of C and $k_c, k_{G\backslash C}$ are the total degrees of C and the rest of the graph$G\backslash C$.

A subset *A C V* of a graph is **connected** if any two vertices in *A* can be joined in a path such that all intermediate points also lie in *A*

The nonempty sets $A_1 \ldots A_k$ form a **partition** of the graph if $A_i \cap A_j = \emptyset$ and $A_1 \cup \ldots \cup A_k = V$

For the size of a subset $A \subset V$

$$|A| = the\ number\ of\ vertices\ in\ A \quad [6]$$

$$vol(A) = \sum_{i \in \mathrm{A}} d_i \quad [7]$$

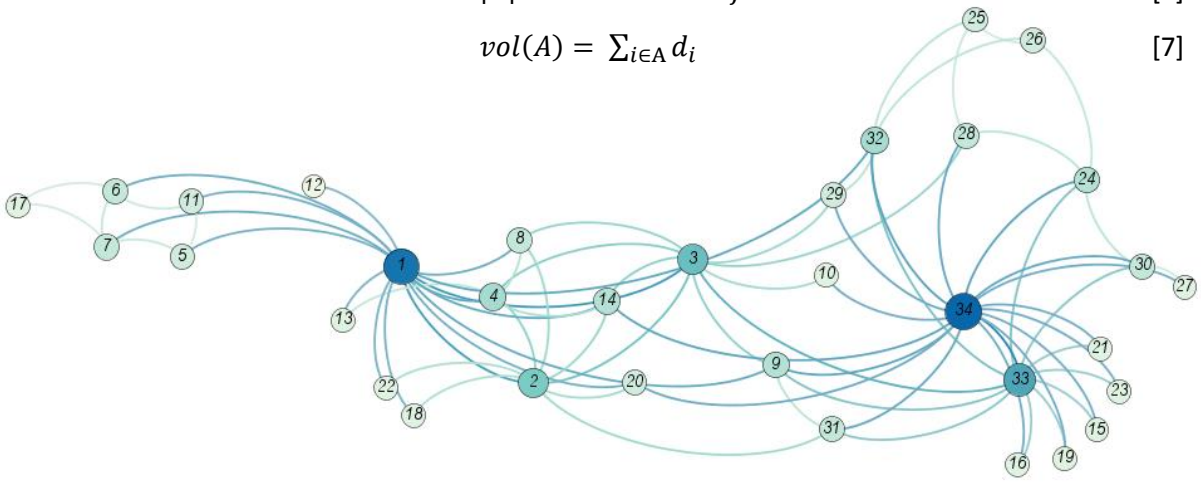

Figure 1: Unweighted graph to show social connections in Zachary Karate Club. This famous problem arose when disagreements between members caused the club to split. Community detection techniques would allow mathematicians to use the graph to predict how the members would partition Vertices with high degrees are larger and darker, representing members with more social links to other members. Edgelist source: Santa Fe Institute

3. Context: DOTA 2 and Community Detection

3.1: Brief overview of DOTA 2

DOTA 2 (Defence of the Ancients 2) is competitive multiplayer RPG (Role Playing Game) developed by Valve and sequel to a popular modification of Blizzard's "Warcraft III". The game is played casually by millions worldwide, but is also known for its professional scene. The prize pool as of today [19/11/15] stands at $18,429,613, where the 1st place team of 5 stands to win $6,634,661.

2 teams of 5 players are pitted against one another where each player takes control of 1 of 110 playable characters; named in the game and in this report as "heroes". The two teams begin the game on opposite sides of a large arena, inside their respective bases. The aim of the game is to destroy the opponents "Ancient"; a heavily defended structure within each team's base

Progress can be made in the game by gaining experience and gold (the game's currency) by killing the opponent's heroes, who respawn after being killed in the opponent's base after a period of time. Similar progress is made by destroying enemy structures protecting the opponent's ancient. The game follows a "rich get richer" dynamic; gaining experience causes individual heroes to "level up", resulting in stronger heroes with more abilities and versatility, who are in turn, harder to kill and more useful to their team.

The intricacies of the gameplay are unimportant. Essentially the game comprises of 5 unique heroes with differing abilities and strengths competing against another 5. Picking a strong hero combination is a factor which will contribute to a team's success.

3.2: Interpretations of strong hero combination

Simply choosing heroes which average a high ratio of kills to deaths per game is not an effective strategy. Many heroes, often named "carry heroes" rely on others, sometimes deemed "support heroes" to set up kills for them, or to push against enemy structures effectively. Choosing heroes with the highest individual success rate is similarly not ideal, as even their individual success could be improved upon or worsened by the heroes with which they co-operate. Some heroes are known to work well together, where their abilities complement each-others and yield greater success. Chaos_Knight and Wisp for instance, were deemed "best friends" by officials in an international match due to the frequency with which they are found together in the arena.

3.3: Correspondence to Community Detection

The aim of the project detailed by this report is to find a community of 5 heroes which would be deemed a strong, complementary team. We can define the notion of similarity in context as the success rate between two heroes ie. The rate of success where two heroes play on the same team. In this way the problem becomes a community detection problem. Vertices represent heroes, the weight w_{ij} of the edges represent the similarity between them.

We therefore have, for heroes h_i, h_j

$$s_{ij} = \frac{r}{n} \qquad [8]$$

Where r is the number of successful matches where h_i *plays with* h_j *,* and n is the number of matches where h_i and h_j play together

For 110 playable heroes, a 110x110 adjacency matrix W can be formed. Spectral clustering together with a suitable algorithm such as k-means can then be applied to partition the graph into clusters with strong similarities.

4. Sampling Process to Identify Optimal Groups

4.1: Source

Match information was sampled from the site Dotabuff.com where results from DOTA 2 matches are automatically uploaded and updated.

4.2: Data Sampled

Match information from the last 2,764,964 matches of players in the "high skill" category was analyzed. Only the "high skill" category was considered as more experienced players will better understand the need to utilize strong hero combinations, therefore the results are more indicative of the strength of the combinations, rather than solely the players' skill.

For all heroes $h_i, h_j, i, j \epsilon (1, \dots 110)$ the ratio of matches won to matches played between two heroes h_i, h_jon the same team was constructed.

The mean number of matches played between heroes $h_i, h_j, i, j \epsilon (1, \dots 110)$ was found to be 461.211676

4.3: Conversion to an adjacency matrix

The weight w_{ij} of the corresponding row i and column j of the weighted adjacency matrix W represents the success ratio between h_i, h_j

For hero combinations resulting in a similarity $s_{ij} < 0.45$ the connection is severed and treated as $w_{ij} = 0$ in the weighted adjacency matrix, as is standard practice when introducing algorithms designed for sparse matrices.

The corresponding undirected weighted graph $G = (V, E)$ is then formed from this matrix,

4.4: Exponentiation of adjacency matrix

As previously mentioned, clustering algorithms are generally more effective when applied to sparse graphs. This includes graphs with fewer connections, but in the case of weighted graphs, sparse can also refer to large differences between the weights. In order to further emphasise larger results, the values in the weighted adjacency matrix can be exponentiated such that a weight w_{ij} is substituted for $w^*{}_{ij}$ where

$$w^*{}_{ij} = \begin{cases} e^{\alpha w_{ij}} \; for \; w_{ij} \geq 0.45 \\ 0 \quad\quad for \; w_{ij} < 0.45 \end{cases} \qquad [9]$$

And α is a chosen constant such that the graph becomes sufficiently sparse.

At this point it is appropriate to note that in section 8.2, where the final results are re-evaluated, they are re-evaluated using an exponentiated version of the weighted adjacency matrix, choosing $\alpha = 3.5$, which makes the matrix sufficiently sparse. This is visualized by the sections of the heatmaps in figures 3 and 4.

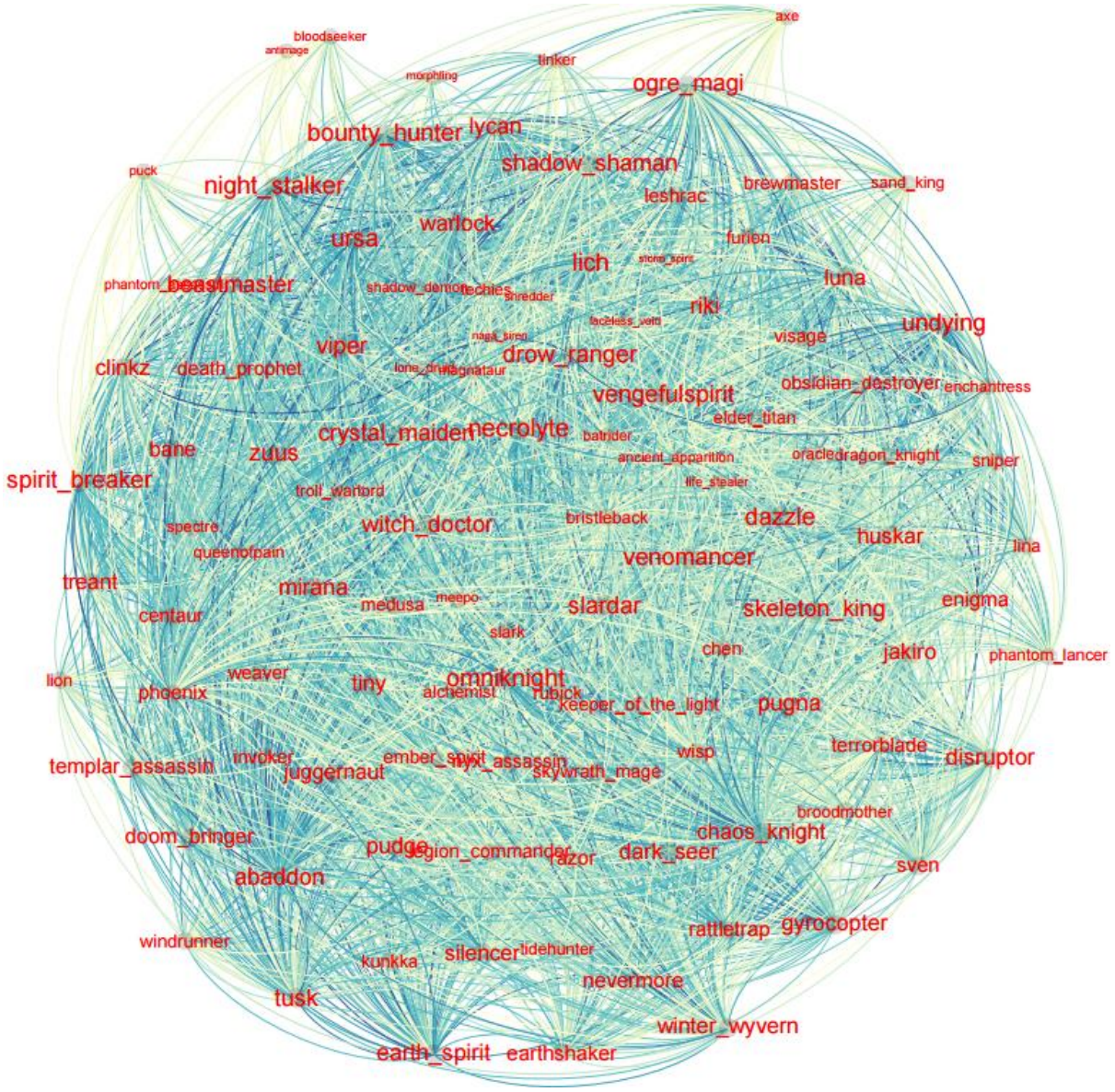

Figure 2: The corresponding similarity graph $G = (V, E)$ of the similarity matrix. Darker edges correspond to greater weights in connections. Larger nodes and node labels correspond to greater vertex degrees. Communities are clearly indiscernible, however vertices such as "omniknight", "bounty_hunter" or "night_stalker" might be more expected to appear in optimal clusters due to their strong connections to other vertices.

Figure 3: Sample of heatmap corresponding to original adjacency matrix. Darker cells indicate higher weights.

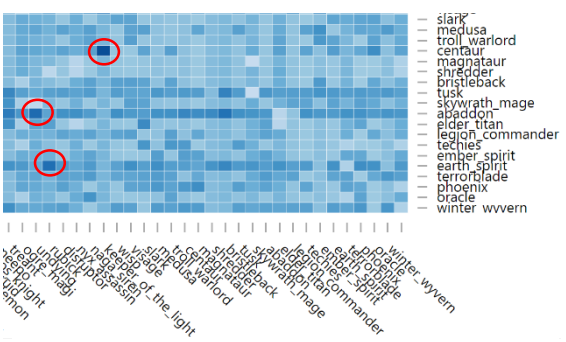

Figure 4: Sample of heatmap corresponding to exponentiatied matrix. We see that the larger weights represented in the original heatmap (figure 3) are now darker, indicating that they now contribute a higher relative weight.

5.1: Summary of Spectral Clustering

This summary will follow von Luxburg's "A tutorial on Spectral Clustering" and will be consistent with the definitions stated there. Whilst there is no universally agreed upon definition of a community, the approach of spectral clustering is to maximise the sum of the total in-cluster weights and minimise the sum of the total between cluster weights. The process begins with spectral embedding, which transforms the data from an adjacency matrix into a set of points in space, represented by elements of eigenvectors. This advantageous step prepares the data for a suitable algorithm such that clustering issues are bypassed. A great advantage of spectral clustering is the versatility with which it can be used, since the user can choose any appropriate method following spectral embedding. This report uses k-means and hierarchical clustering in conjunction, which proves to be appropriate and informative for the scale of this dataset.

The general steps taken to achieve this are as follows

- Construct W, the weighted adjacency matrix of a graph G
- Compute the Laplacian L of W, normalising if necessary
- Compute the first k eigenvectors of L
- Define a matrix $U \in \mathbb{R},^{nxk}$ where its columns comprise of the eigenvectors $u_1, \dots u_k$ of L
- Use a k-means or another suitable clustering algorithm to cluster the points $(y_i) i = 1, \dots, n$ into clusters $C_1, \dots C_k$

5.2: Explanation of the Method

5.2.1: A min-cut problem

This explanation will focus on giving intuitive reasoning behind spectral clustering methods from the perspective of a min-cut problem. For full proofs and propositions the reader should refer to von Luxburg's tutorial.

We define $W(A, B)$ as the sum of the weights between two subsets $A, B \subset V$

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij} \quad [10]$$

Where A, B are sets such that $A, B \subset V$ where V is the set of vertices $v_1, .., v_n$

Our aim is to partition the graph such that the sum of the weights between two subsets are minimised. This translates into choosing the partition $A_1, \dots, A_k$ such that the $cut(A_{1,\dots,} A_k)$ is minimised where

$$cut(A_{1,\dots,} A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \bar{A}_i) \quad [11]$$

Where $\bar{A}_i$ is the complement of A_i

In order to avoid the degenerate case, where simply one node would be partitioned, extensions of the concept of conductance [5] are introduced.

We define for the unnormalised case:

$$RatioCut(A_1, \dots, A_k) = \sum_{i=1}^{k} \frac{cut(A_1 \bar{A}_i),}{|\bar{A}_i|} \quad [12]$$

Where $|A_i|$ is the number of vertices in A_i [6]

And for the normalised case:

$$Ncut(A_1, \dots, A_k) = \sum_{i=1}^{k} \frac{cut(A_1 \bar{A}_i),}{vol(A_i)} \quad [13]$$

Where $vol(A_i)$ is the sum of the degrees of all vertices in A_i [7]

We see now that in order to minimise $RatioCut(A_1, \dots, A_k)$ the number of vertices partitioned must be large, similarly to minimise $Ncut(A_1, \dots, A_k)$ the sum of the degrees of the vertices partitioned must be high.

To see this in context, consider the following subset of vertices from the DOTA 2 graph

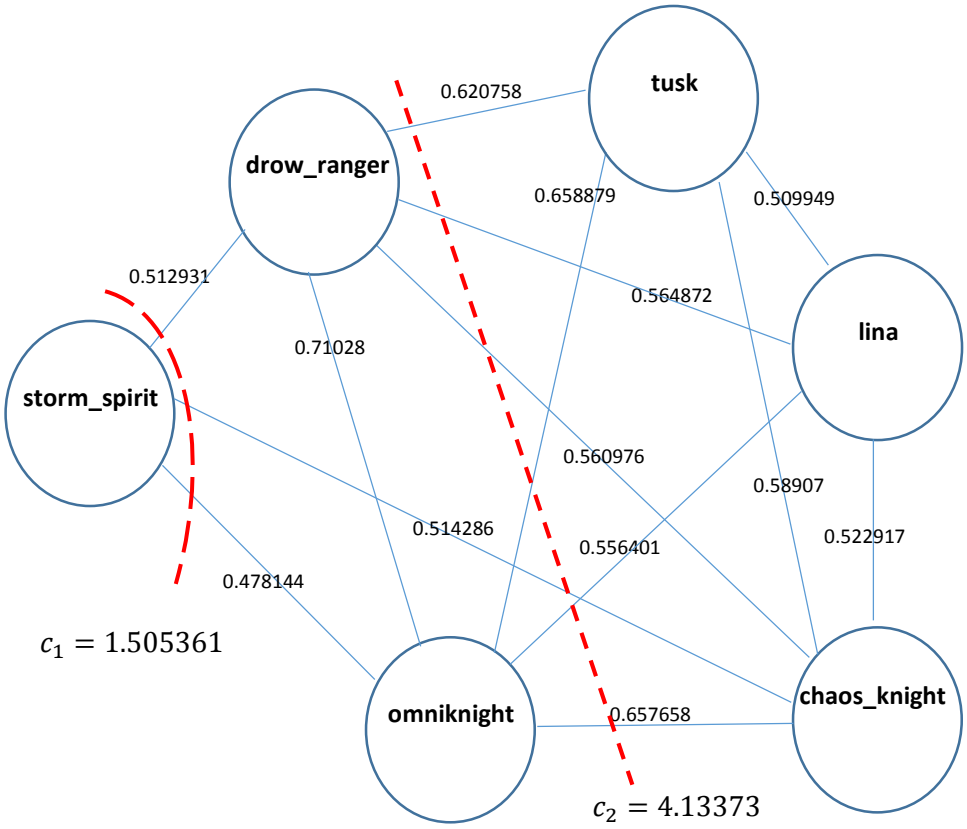

Figure 5: Sample vertex set showing 2 cuts: $c_1 = cut(storm\_spirit)$, the degenerate case, and $c_2 = cut(storm\_spirit, drow\_ranger, omniknight)$ which partitions the graph into 2 subsets

We note that $RatioCut(A_1, A_2) = 1.505361$

where $(storm\ spirit) \subset A_1, (drow\ ranger, tusk, lina, chaos\ knight, omniknight) \subset A_2$

While $RatioCut(A_1, A_2) = 1.37791$ where

$(storm\ spirit, drow\ ranger, omniknight) \subset A_1, (tusk, lina, chaos\ knight) \subset A_2$

Therefore when solving the minimum cut problem, the degenerate case is no longer a preferable cut which solves the partitioning problem.

Figure 2

5.2.2 The Graph Laplacian

The method of spectral clustering has at its core the graph laplacian L defined as

$$L = D - W \quad [14]$$

Where W is the weighted adjacency matrix [2] and D is the degree matrix [3]

The normalized graph laplacian L_{rw}, related to a random walk is defined as

$$L_{rw} = D^{-1}L \quad [15]$$

Where D is the degree matrix [3] and L is the graph laplacian [14]

The effect of the normalized graph laplacian is to convert the minimization problem from a minimization of RatioCut to Ncut.

Following von Luxburg's tutorial, we see that the normalized graph laplacian is preferable to the unnormalised version for two main reasons.

Firstly, the Ncut minimization method implements 2 clustering objectives, not only finding partitions which minimize the similarity between dissimilar clusters, but also in maximizing in-cluster similarities. This is particularly relevant in context as this project aims to identify an optimal community partitioned from a graph, which emphasizes the maximization of that community's in-cluster similarities. The alternative of using a RatioCut unnormalized method may be more balanced in favour of other clusters by only considering their dissimilarity. This may result in finding larger numbers of strong clusters rather than 1 or a small number of exceptionally strong clusters.

Secondly, unnormalised graph laplacian's are not preferable due to issues of convergence. Essentially, when adding more and more data points such that for n data points, $n \to \infty$, both the eigenvalues and eigenvectors of the normalized graph laplacian converge to some operator U. In contrast, the eigenvalues and eigenvectors for even small sample sizes can have unreliable results due to problems associated with spectral convergence. For further details, see von Luxburg's tutorial.

We will therefore use a normalized graph laplacian, of the form, $L_{rw} = D^{-1}L$. There is also the option of using the symmetric normalized graph laplacian $L_{sym} = D^{-1/2}LD^{-1/2}$. However as von-Luxburg points out, this has no computational advantages and may have undesired effects in the eigendecomposition by additionally multiplying its eigenvalues with $D^{1/2}$.

It can be shown that by defining the indicator vectors

$$h_{ij} = \begin{cases} \frac{1}{\sqrt{vol(A_j}} \; if \; v_j \in A_j \\ 0 \; otherwise \end{cases} (i = 1, \ldots . n; j = 1, \ldots k) \quad [16]$$

The solution of Ncut problem can be expressed as the first k generalized eigenvectors of $Lu = \lambda Du$

Therefore we can solve the Ncut spectral clustering problem by computing the unnormalized graph laplacian and degree matrix of a given adjacency matrix and the first *k* eigenvectors of the normailised graph laplacian $l_{rw} = D^{-1}L$

5.2.3: Eigen decomposition

Computing the eigenvalues $\lambda_1, \dots, \lambda_n$ and corresponding eigenvectors $u_1, \dots, u_n$ of the normalized graph laplacian is a problem that can be solved with mathematical software such as R or Matlab. Difficulty arises in choosing a k to determine the first k eigenvectors.

A general rule is to use the eigengap heuristic. The number k should be chosen such that $\lambda_1, \dots, \lambda_k$ are small relative to λ_{k+1}. A standard method would be to construct a histogram and look for a jump in the size of the eigenvalues. Intuitively, the eigenvalues and their multiplicity represent the connectedness of a graph. For graphs with k distinct, unconnected components, $\lambda_1 = \lambda_2 = \dots = \lambda_k = 0$ and $\lambda_{k+1} > 0$. Where k components have very few connections $\lambda_1 \approx \lambda_2 \approx \dots \approx \lambda_k$ with λ_{k+1} being noticeably larger than λ_k.

Identifying and choosing a suitable k then allows for the construction of a matrix $U \in \mathbb{R},^{nxk}$ of k eigenvectors $u_1, \dots u_k$ of L, upon which to perform a clustering algorithm

5.2.4: The k-means step

While k-means is the most common algorithm following spectral embedding, any suitable clustering algorithm is equally permissible to use at this stage. Very simply, k-means attempts to configure clusters of data through an iterative process of centralisation. The process follows these steps

1) Choose the number of clusters
2) Assign initial cluster centres (preferably far from each other to increase the efficiency of the process).
3) Attribute each data point to its closest cluster (where the distance is defined as the minimum value of the sum of the weights of the edges necessary to connect the data point to the cluster centre)
4) Re-centre the clusters by assigning the new cluster centre as the point which is the mean distance of all data points currently attributed to the cluster.
5) Is the cluster a final solution? If yes, give the solution, if no, repeat steps (3) and (4)

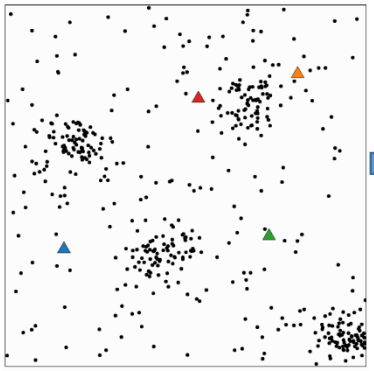

Figure 6: First Assignment of Cluster Centres

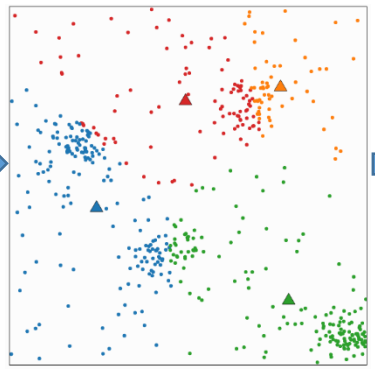

Figure 7: Connection of Nodes followed by Re-Centering of Clusters

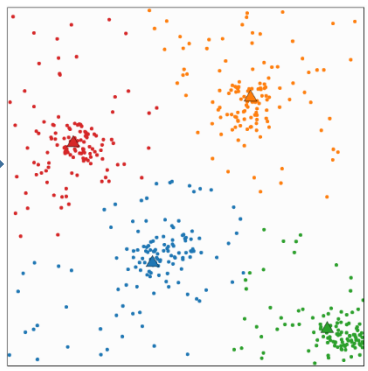

Figure 8: Final Solution: Steps 3 and 4 when repeated converge to this result

This simple algorithm will effectively partition data, provided that the eigenvalue decomposition of the normalised laplacian matrix yielded a pronounced eigengap, implying well defined clusters. As stated in Von Luxburg's tutorial "in ambiguous cases it also returns ambiguous results"

K-means gives no guarantee of splitting clusters into desired sizes. Where a certain sized cluster is required, further knowledge of the relationships between vertices may needed.

5.2.5: Additional Hierarchical Clustering Method

In order to gain further knowledge of the clustering of vertices and their relationships, hierarchical clustering may also be used.

Hierarchical clustering follows a similar process to k-means with regards to notion of minimising distances between vertices, with the key element of compiling vertices into a group once they are joined. The process follows these steps

1) Start with an initial set of data points, where each data point is assigned to be a group made up only of itself.
2) Connect each of the two closest groups, where the distance is the weight of the edge of minimum weight needed to connect the 2 groups. Once connected, these merge into a single group
3) Repeat (2) until all data is merged into a single group
4) Give solution

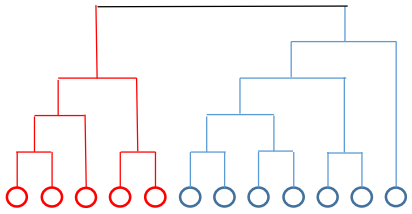

Figure 9: A simple representation of a cluster dendogram; the result of hierarchical clustering. Of interest are the relationships between nodes on several levels, which we see on several levels as a result of connecting groups (step 2). Loosely defined clusters (indicated by colour) and subclusters may also be obtained

Each level of the group size is a hierarchical segmentation, with the final resulting cluster representing the top of the hierarchy.

Due to the ease at which the vertex relationships and hierarchy can be visualised, this provides greater user insight into the relationships between vertices.

6: Worked Example with Small Vertex Set

Consider the vertex set of figure 5: The weighted adjacency matrix W for the subset of vertices $(v_1, \dots, v_6) \epsilon\ V$ is given by

	drow_ranger	storm_spririt	Lina	omniknight	chaos_knight	Tusk
drow_ranger	0	0.512931	0.564972	0.71028	0.560976	0.620758
storm_spirit	0.512931	0	0	0.478144	0.514286	0
lina	0.564972	0	0	0.556401	0.522917	0.509949
omniknight	0.71028	0.478144	0.556401	0	0.657658	0.658879
chaos_knight	0.560976	0.514286	0.522917	0.657658	0	0.57907
tusk	0.620758	0	0.509949	0.658879	0.57907	0

Table 1: adjacency matrix of the vertex set in Figure 5

With corresponding degree matrix D

	drow_ranger	storm_spririt	lina	omniknight	chaos_knight	Tusk
drow_ranger	2.969917	0	0	0	0	0
storm_spirit	0	1.505361	0	0	0	0
lina	0	0	2.154239	0	0	0
omniknight	0	0	0	3.061362	0	0
chaos_knight	0	0	0	0	2.834907	0
tusk	0	0	0	0	0	2.368656

Table 2: Degree matrix of the vertex set in Figure 5

The unnormalized graph laplacian given by $L = D - W$ is therefore

	drow_ranger	storm_spririt	lina	omniknight	chaos_knight	Tusk
drow_ranger	2.969917	-0.51293	-0.56487	-0.71028	-0.56098	-0.62076
storm_spririt	-0.51293	1.505361	0	-0.47814	-0.51429	0
lina	-0.56497	0	2.154239	-0.5564	-0.52292	-0.50995
omniknight	-0.71028	-0.47814	-0.5564	3.061362	-0.65766	-0.65888
chaos_knight	-0.56098	-0.51429	-0.52292	-0.65766	2.834907	-0.57907
Tusk	-0.62076	0	-0.50995	-0.65888	-0.57907	2.368656

Table 3: Unnormalized graph laplacian of the vertex set in Figure 5

The inverse D^{-1} of the degree matrix D is given by

	drow_ranger	storm_spririt	lina	omniknight	chaos_knight	Tusk
drow_ranger	0.33671	0	0	0	0	0
storm_spririt	0	0.664293	0	0	0	0
lina	0	0	0.464201	0	0	0
omniknight	0	0	0	0.326652	0	0
chaos_knight	0	0	0	0	0.352745	0
Tusk	0	0	0	0	0	0.42218

Table 4: Inverse degree matrix D^{-1} of the vertex set in Figure 5

Therefore the normalized graph laplacian given by $L_{rw} = D^{-1}L$

	drow_ranger	storm_spririt	lina	omniknight	chaos_knight	Tusk
drow_ranger	1	-0.17271	-0.1902	-0.23916	-0.18889	-0.20902
storm_spririt	-0.34074	1	0	-0.31763	-0.34164	0
lina	-0.26226	0	1	-0.25828	-0.24274	-0.23672
omniknight	-0.23201	-0.15619	-0.18175	1	-0.21483	-0.21522
chaos_knight	-0.19788	-0.18141	-0.18446	-0.23199	1	-0.20426
Tusk	-0.26207	0	-0.21529	-0.27817	-0.24447	1

Table 5: Normalised graph laplacian matrix L_{rw} of the vertex set in Figure 5

Eigenvalues	Eigenvectors
$\boldsymbol{\lambda_1 = 0.}$	$u_1 =$ [1,1,1,1,1,1]
$\boldsymbol{\lambda_2 = 0.9091618}$	$u_2 =$ [0.3323871 0.260010703 -0.08041519 -0.758746679 -0.10992703 -0.4082506]
$\boldsymbol{\lambda_3 = 1.192856}$	$u_3 =$ [0.2333486 -0.718276755 0.33028807 0.112715236 -0.03863039 -0.4082503]
$\boldsymbol{\lambda_4 = 1.220015}$	$u_4 =$ [-0.3179844 -0.312115312 -0.70772692 -0.053359188 0.42378112 -0.4082502]
$\boldsymbol{\lambda_5 = 1.247039}$	$u_5 =$ [-0.6946093 -0.002796964 -0.06226529 0.030250602 -0.80472628 -0.4082493]
$\boldsymbol{\lambda_6 = 1.430921}$	$u_6 =$ [0.3276682 0.410893471 -0.18705531 0.638558344 -0.06830723 -0.4082391]

Table 6: Table of the eigenvalues and their corresponding eigenvectors for the normalised graph laplacian L_{rw} of the vertex set in Figure 5

We would then construct a matrix of eigenvectors $U \in R^{nxk}$containing the vectors $u_1, \dots, u_k$ as columns and perform k-means (or another suitable clustering technique) on the rows.

7: Application of Techniques onto DOTA 2 Data Set

Here we apply the methods we have discussed to the full data set with the aim of finding strong groups from within pools of compatible heroes. Following spectral embedding we obtain the following scatterplot of eigenvalues.

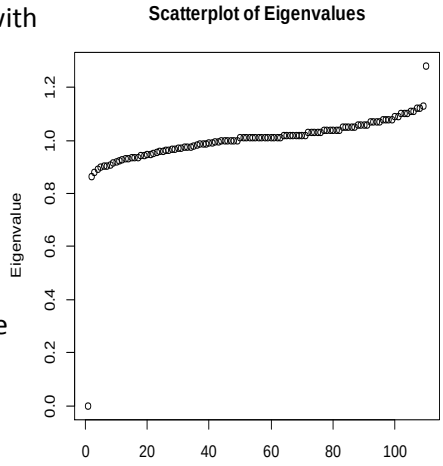

Figure 10 scatterplot of eigenvalues from the normalised graph laplacian where the x axis is labelled k

The scatterplot immediately implies choosing k=2, which would partition the graph into 2 large clusters. This information is not particularly informative when attempting to find groups of 5 from small pools. Aside from $k = 2$, and the trivial k=110, the lacking obvious eigengap and the graph's own unclear community structure imply that there is no obvious pick for k for the k first eigenvectors.

This leaves us with a degree of choice. A potential choice would be to assign k to be 22, with the hope to split the 110 vertices into clusters of around 5 (since 110/22 = 5). Although $\lambda_{22} = 0.953124$ and $\lambda_{23} = 0.9562849$, showing a very small eigengap between λ_k and λ_{k+1}, this is similar to all other feasible eigenvalues.

However, since we are looking to create pools and to allow some room for cluster sizes to vary and still be considered, we assign k to be a slighly smaller value. $k = 20$ was found to generate clusters large enough that the strongest heroes were in pools large enough to be considered, but small enough to easily extract clusters of 5 by referencing the cluster dendogram. After performing k-means on the eigenvector matrix of $u_1, \ldots, u_k$, the following clusters were obtained.

1	2	3	4	5	6	7
bane	rubick	axe	juggernaut	slardar	mirana	lion
zuus	nyx_assassin	bloodseeker	riki	lich	shadow_shaman	queenofpain
warlock	keeper_of_the_light	puck	luna	venomancer	skeleton_king	spectre
beastmaster	centaur	lone_druid	clinkz	dazzle	pugna	slark
viper	skywrath_mage	magnataur	huskar	bounty_hunter	dark_seer	troll_warlord
treant		shredder	doom_bringer	spirit_breaker	ogre_magi	bristleback
abaddon			lycan	undying	disruptor	
phoenix				tusk	winter_wyvern	

8	9	10	11	12	13	14
earthshaker	storm_spirit	windrunner	phantom_assassin	sand_king	Antimage	razor
nevermore	naga_siren	kunkka	life_stealer	tidehunter	Morphling	sven
rattletrap		alchemist	batrider	tinker	Faceless_void	death_prophet
gyrocopter		legion_commander	ancient_apparition	shadow_demon		
silencer		ember_spirit	meepo	visage		

15	16	17	18	19	20
Furion	Lina	enigma	drow_ranger	phantom_lancer	crystal_maiden
Enchantress	Sniper	dragon_knight	vengefulspirit	leshrac	pudge
Wisp	invoker	weaver	necrolyte	broodmother	tiny
Elder_Titan	obsidian_destroyer	chen	omniknight	jakiro	witch_doctor
Techies	medusa	brewmaster	night_stalker	chaos_knight	templar_assassin
oracle			ursa	terrorblade	earth_spirit

Tables 7,8,9: Pools generated by performing k-means

From these pools, strong clusters of 5 heroes can be extracted by inspecting the corresponding cluster dendogram, obtained by performing hierarchical clustering upon the data set. Some strong groups generated include:

crystal_maiden
pudge
witch_doctor
earth_spirit
tiny

*Table 10: Cluster 1 from group 20: Overall sum of weights **5.794858** Average 0.5794858 per connection*

bane
Beastmaster
Viper
Treant
abaddon

*Table 11: Cluster 2 from group 1. Overall sum of weights **5.907897** (Average 0.5907897 per connection)*

Ursa
Vengeful_spirit
necrolyte
Omniknight
Night_stalker

*Table 12: Cluster 3 from group 18. Overall sum of weights **6.6583247** (Average 0.6583247 per connection)]*

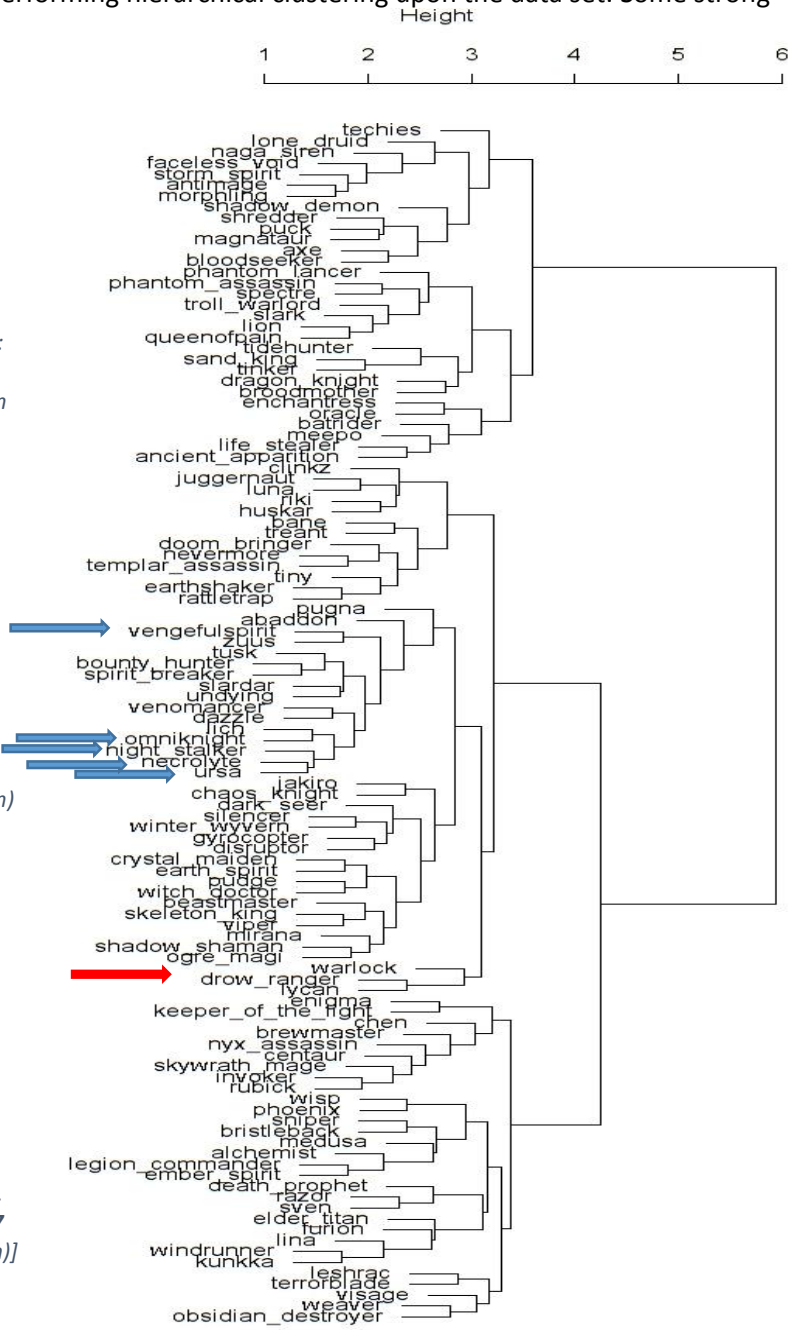

Figure 11: Corresponding cluster dendogram obtained from hierarchical clustering. The proximity of heroes to each other relates to the strength of their relationship. For cluster 3, group 18 is shown, blue arrows indicating that they appear in final combination, red indicating rejection due to low proximity to other heroes.

8: Analysis of Results

8.1: Assessment and Criticism of Results

The Current results present groups of respectable strength, notably cluster 3, with an average weight of 0.6623263 per connection. However the current results can be improved upon to find an even stronger group of 5. Clusters may be limited due to the spectral clustering algorithm focusing on finding clusters that are strong relative to the overall strength of the heroes they contain, creating more balanced pools. This is not ideal for the purposes of this report, where very strong groups are looking to be prioritised, regardless of the effect upon weaker groups.

8.2: Re-Evaluation using Exponentiated Adjacency Matrix

To address the problem of finding higher strength groups, an exponentiated version of original adjacency matrix will be used (see section 4.4) in order to prioritise higher weight connections. Applying the same spectral clustering method upon the exponentiated adjacency matrix, the following pools are generated.

1	2	3	4	5	6
sand_king	lina	nyx_assassin	axe	phantom_assassin	lion
tinker	leshrac	keeper_of_the_light	bloodseeker	life_stealer	queenofpain
obsidian_destroyer	weaver	wisp	puck	batrider	spectre
visage	invoker	medusa	shadow_demon	ancient_apparition	slark
	skywrath_mage	elder_titan	lone_druid	meepo	Troll_Warlord
			magnataur		Bristleback
			shredder		

7	8	9	10	11	12	13
earthshaker	juggernaut	vengefulspirit	windrunner	nevermore	crystal_maiden	tidehunter
pudge	riki	slardar	kunkka	zuus	mirana	enchantress
tiny	skeleton_king	lich	alchemist	pugna	shadow_shaman	broodmother
rattletrap	luna	necrolyte	rubick	dark_seer	warlock	oracle
	huskar	dazzle	centaur	gyrocopter	beastmaster	phoenix
	lycan	omniknight	legion_commander	silencer	viper	
		night_stalker	ember_spirit	disruptor	clinkz	
		bounty_hunter		abaddon	ogre_magi	
		ursa				
		spirit_breaker				
		tusk				

14	15	16	17	18	19	20
bane	razor	witch_doctor	antimage	drow_ranger	storm_spirit	enigma
templar_assassin	sven	venomancer	morphling	phantom_lancer	naga_siren	dragon_knight
doom_bringer	sniper	jakiro	faceless_void	chaos_knight		furion
treant	death_prophet	undying		techies		chen
terrorblade		earth_spirit				brewmaster
winter_wyvern						

Tables 13,14,15: Pools generated by performing k-means (exponentiated data)

From these pools the same cluster dendogram for the original adjacency matrix can be used to find some combinations of reasonable strength, such as cluster 4, whose average weight per connection is 0.5882743. However the main benefit of using the exponentiated data is represented by cluster 5, whose average weight per connection is 0.7007814. Spectral clustering now succeeds in finding a cluster of even higher strength using exponentiated data.

crystal_maiden
viper
clinkz
Ogre magi
Mirana

*Table 16: Cluster 4 from pool 12: Overall sum of weights= **5.882743.** Average weight per connection= 0.5882743*

witch_doctor
venomancer
jakiro
undying
earth_spirit

*Table 17: Cluster 5 from pool 16: Overall sum of weights= **7.007814** Average weight per connection= 0.7007814*

8.2 Interpreting the Results in Context

It is noteworthy that all the groups presented in this report have a roughly equal ratio of support heroes to carry heroes, justifying their necessity in the game, despite having generally a lower kill to death ratio. The exception to this rule is the final group; cluster 5 given by table 17. This group is almost entirely made up of support heroes, with all heroes aside from earth spirit being deemed supports. The success of this group may well be explained by the potential that every hero has for enormous area of effect damage, all 5/5 heroes are assigned as being "nukers" as well as supports, something which is not especially useful individually, but in teamfights is very effective. Witch doctor contributes the highest weight in the group. This is likely due to her ability to paralyse large numbers of opponents. When this ability is coupled with the other members' large area of effect abilities, which would be likely to damage the entirety of a paralysed opposing team, this particular group of 5 would be very strong in large scale team fights. An experienced team would use this to their advantage, which may explain why these heroes in combination have been shown to be so successful.

8.3 Possible Alternative Methods

Since this report aims to find rough communities, leaving potential for overlapping communities of heroes, an overlapping technique such as fuzzy partitioning may have also been a viable option. However pool and cluster sizes would be more difficult to specify, possibly resulting in pools too large to find hero combinations within by inspection, or with too much overlap to reach any informative conclusion about the data. In this case the combination of spectral embedding with k-means and hierarchical clustering was an effective method. For a much larger data set, where using a cluster dendogram to specify groups would no longer be a reasonable process to perform, fuzzy partitioning may be more appropriate.

References:

Ulrike von Luxburg: *A Tutorial on Spectral Clustering*, March 2007
Santo Fortunato: *Community detection in Graphs* Jan 2010
David M. Blei: *Hierarchical clustering* February 2008
MacQueen, J. B., in Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, edited by L. M. L. Cam and J. Neyman (University of California Press, Berkeley, USA), volume 1, pp. 281–297 1967
Bezdek, J. C., *Pattern Recognition with Fuzzy Objective Function Algorithms* (Kluwer Academic Publishers, Norwell, USA) 1981
Đỗ Ngọc Tuấn: YouTube series on Spectral Clustering
Golub, G. H., and C. F. V. Loan, *Matrix computations* (John Hopkins University Press, Baltimore, USA) 1989

Other

Prize pool information sourced on date 19/11/15 from
https://www.dota2.com/international/compendium/
Match Information sourced from Dotabuff.com
Cluster Visualisation created with: *Visualising K-Means Clustering* by *Karanveer Mohan. http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html*
Notable R libraries/functions used: "d3heatmap" "kernlab" "igraph" "stats" "hclust" "solve()" "plot()"

Figures 1,2 created with Gephi 0.8.2

Source of Zachare Karate Club Edgelist: Santa Fe Institute
http://tuvalu.santafe.edu/~aaronc/randomgraphs