INSTITUT FÜR INFORMATIK
LUDWIG-MAXIMILANS-UNIVERSITÄT MÜNCHEN

Zulassungsarbeit zur wissenschaftlichen Prüfung für das
Lehramt an Gymnasien in Bayern

Ranking on Multipartite Graphs

Eingereicht bei:	Prof. Dr. François Bry
Eingereicht von:	Niels Becker
	Zieblandstraße 3
	80799 München
Betreuer:	Christoph Wieser
Abgabetermin:	1.4.2013

Abstract

Graphs are commonly used to model complex and large scaled structures. Friendship relations on a social platform, recommendations in big online marketplaces like e-bay or Amazon, or the link structure of the web, are examples for such structures. It is generally accepted (and appreciated) that these huge web structures tend to grow. But with a growing amount of stored data and higher complexity of the networks, one has to find new scalable analysing tools that supply reliable information about the vertices e.g. actors in these networks. Ranking algorithms are such tools that map each node to a specific weight due to its specific behaviour in the system. Ranking on multipartite graphs is of special interest. Such graphs can be used to model systems with typed elements, for example a university with lecturers, courses and students, or buyers and sellers in an online marketplace. In this thesis a new eigenvector based ranking-algorithm is proposed: the $A_n - H_n$-Rank. It is designed to be applied on multipartite graphs with any number of types. First, the structure of multipartite graphs is examined, and the mathematical preliminaries are developed, on which the algorithm is based. The convergence of the algorithm is proven, and the algorithm is applied to gathered data.

Zusammenfassung

Graphen werden häufig benutzt, um komplexe und große Strukturen zu modellieren. Freundschaftsverhältnisse in einem sozialen Netzwerk, Empfehlungen in einem großen Online-Warenhaus wie e-bay oder Amazon oder die Verknüpfungsstruktur des Internets sind Beispiele solcher Strukturen. Es wird üblicherweise angenommen, dass diese riesigen Web-Strukturen stetig wachsen. Mit einer wachsenden Menge von gespeicherten Daten und immer größer werdender Komplexität der Netzwerke benötigt man neue skalierbare Analysemethoden, die verlässliche Daten über die Akteure bzw. Knoten in diesen Netzwerken liefern. Ranking-Algorithmen sind solche Analysewerkzeuge, die jedem Knoten ein bestimmtes Gewicht zuordnen, das von dem Verhalten des Knotens im System abhängt. Ranking auf multipartiten Graphen ist hierbei von besonderem Interesse. Solche Graphen können verwendet werden, um Systeme mit typisierten Elementen zu modellieren, beispielsweise eine Universität mit Dozenten, Lehrveranstaltungen und Studenten oder Käufer und Verkäufer in einem Online-Warenhaus. In dieser Arbeit wird ein neuer eigenvektorbasierter Ranking-Algorithmus beschrieben: der $A_n - H_n$-Rank. Dieser Algorithmus ist auf multipartite Graphen mit einer beliebigen Anzahl von Typen anwendbar. Zunächst wird hierfür die Struktur von multipartiten Graphen untersucht und die mathematischen Vorraussetzungen geschaffen, auf denen der Algorithmus basiert. Es wird bewiesen, dass der Algorithmus konvergiert, und der Algorithmus wird auf gesammelte Daten angewendet.

Acknowledgement

I wish to thank, first and foremost, my Professor Prof. Dr. François Bry for the support during the last few months. It is down to him that I could work on this great topic. I am thankful for his continuing engagement and his helpfulness.

It is a great pleasure to acknowledge my tutor Christoph Wieser whose support and useful feedback motivated me and kept me going. I am glad that he was always there to help and encourage me.

I also wish to thank Harald Zauner for giving me helpful advices for dealing with Pregel, and Mari Gorelashvili from the institute of physics, who helped me to deal with the properties of the English language.

Last but not least I want to thank Dörte Heller for her general support during my whole studies at the university and for helping me understand the properties of GNU Plot.

Contents

1 Introduction

Ranking The ranking of several comparable objects is a problem which we are facing every day. It is strongly connected to the decisions we have to make, and to our personal preferences. A buyer in a supermarket for example could choose to buy the more tasty but more expensive apple: Here, the quality has a higher influence on the buyers ranking than the price. In these marketplace situations different rankings of different people are very obvious, but ranking takes place in various everyday situations. One could choose take the bicycle rather than the car or public transportation to go to work. Maybe the reason for this decision is, that the bicycle would obtain the best ecological ranking. A student in a university could choose one course rather than another because the lecturer has a better reputation (a better reputation rank), or because the end term exam will probably be easier to pass (it has a lower difficulty rank).

Two basic approaches for ranking can be distinguished: The *global* or *query independent* ranking, and the *Query dependent* ranking. The *global* ranking tries to rank objects by a given variable that every ranked object holds. This can be as easy as sorting the apples in the supermarket by price or weight. A more complicated problem would be to rank the importance of different streets in a city, or the importance of different web pages on the Internet. In contrast, a *query dependent* ranking would always try to rank the degree to which an object in question matches a certain query. Of course a buy recommendation only works by query dependent ranking. Many web applications use a combination of these two ideas. *PageRank*, which is used by Google, ranks different web pages by its global importance. *PageRank* is then used to sort the web pages that match a specific search query.

There are many different approaches to rank comparable objects. But all of these approaches depend on additional information that has to be provided in some way by the environment. It is impossible to choose the cheaper product without looking at the price tag, or to choose the better course without gathering information about these courses.

This additional information is often given by the special relations between the objects. For example one may choose to buy a book because he appreciated another book by the same author. The relation between these two books *"written by the same author"* is here important for the personal ranking of the buyer. On the web a surfer will be more likely to visit a special web page, if it has a link to one which he visits often. To make a ranking calculable by a computer we will consider that this relational information is somehow stored in links between the rankable objects. These links may for example carry information about how often two products were bought together. In computer science ranking is mostly done by link analysis.

Ranking can be generally modelled by this scheme:

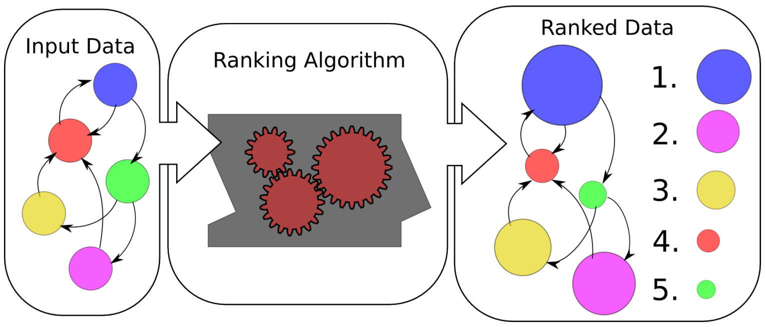


Figure 1: The ranking scheme. The radius of the vertices represents the obtained ranking

Mathematically a ranking is a function that maps objects to positive real numbers (represented by the size of circles) according to the input information. But regardless of the functional character of a ranking, it is common use to refer to *ranking algorithms*. This is certainly appropriate because some of the most important ranking functions, like *PageRank* and *HITS* etc., are computed by iterating certain ranking rules until convergence.

Rating By examining the Networks and Systems that surround us in our every day life, one may find that many of them consist of entities interacting with each other via a numeric rating. A rating from one entity in a system toward another can be regarded as a special form of a relation. As for instance in an online marketplace there are buyers and sellers rating each other. In a school or university you may find students and teachers rating each other with marks and feedback. And there are rating-agencies, which rate states or companies. Another example is facebook with its famous like-button, which can be considered as a binary rating that is summed up to point out appreciated people, events, or comments.

A ranking algorithm that is applied to a rating system tries to turn the existing ratings between the objects into a ranking.

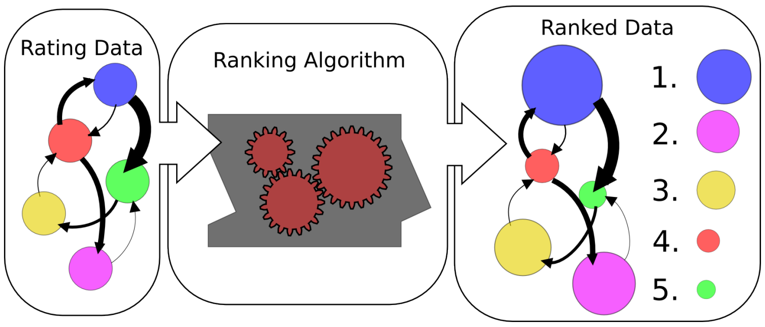


Figure 2: Ranking on a rating system: the width of the links signifies the given rating

One example of a ranking function that is applied to a rating system is the *mean function* which maps each node to its mean incoming rating. This ranking is applied for example by amazon.de. Here the ranking of a product is the mean of all given ratings for this product. This is a very natural way of proceeding. The assumption is that the better a product is rated, the better it must be. These rating systems are largely appreciated. One may rely on them to find a good product, a funny video or what ever the network in question may host.

Challenges and Problems Rating systems bring problems with them, when the gathered ratings in the network get by some reason unreliable for decision making. For example the best-rated product could be sold by an Internet fraudster, and a seller with good ratings could never the less sell a bad product. Here, it would be interesting to have ranking algorithm that takes the gathered ratings into account but also tries to reduce unpleasant effects.

In the case of a false rating by an Internet fraudster, a rating that is in some way false is observed. Therefore an unreliability of the ranking is caused. This problem is referred to as *"injected disturbance problem"*. Another example for injected disturbances are facebook *likes* that are bought from various providers, and not given by real users. On the web rankings that are "higher than deserved" can be caused by spam sites, and there are ranking algorithms as *TrustRank* [9] that already try to reduce these effects.

Another kind of problem could result from the lag of rating, rather then a biased surplus of injected ratings. For instance, if a product has only received one rating, this rating can hardly be considered as reliable. This is referred to problem as *"information lack problem"*. Another typical example for this problematic situation occurs when products do not receive a rating at all.

Obviously the mean function is very vulnerable to the *injected disturbance* problem, as the mean value shifts linearly to the amount and deviation of the injected ratings. In general it can be said that a good ranking function is reliable and somehow resistant to the mentioned problems.

To face these problems the rating system will be modelled in a graph with a set of vertices and a set of edges. The actors in the rating system will be represented by the vertices of the graph. The ratings given by these actors will be represented by directed edges that carry the numerical weight of the according rating. Such a graph is referred to as *rating graph*, as seen in [15].

Multipartite Rating By looking at rating Systems it can be observed that the actors in these systems often belong to different types. At a University the rating system would contain actors of the type "student", the type "lecturer" and the type "course". Each actor in the university belongs to one unique type. Normally two entities of the same type don't rate one another. Two lecturers for instance would not give marks to one another. And on e-bay two separate customers usually don't interact at all. This property gives rise to the idea of modelling these systems in multipartite graphs. Basically a graph is called multipartite if it contains several pairwise disjoint subsets of nodes, referred to as *parts*, where two nodes in the same part may not have a connecting edge. A proper definition of multipartite graphs is given in chapter 2.

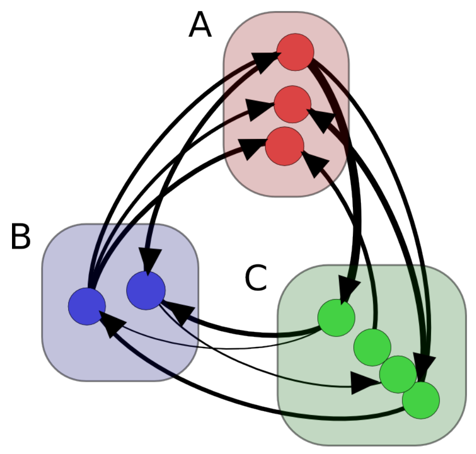


Figure 3: a rating graph with three parts

Different types of objects in the rating graph bring *typed* ratings with them. These could differ in meaning and quality. For example the ratings between parts A and B may be numbers between 1 and 10, while the ratings between parts C and A could be binary. This depends on what the rating system is used for. A student could get a binary rate from a lecturer for a passed (or missed) exam while he could give a feedback in form of a questionnaire with multiple ratings for different aspects of the course. We will focus on simple numerical ratings in this thesis, but even there the problem to compare different types of ratings remains. This problem is referred to as *typed rating* problem.

Contribution This thesis is arranged as follows:

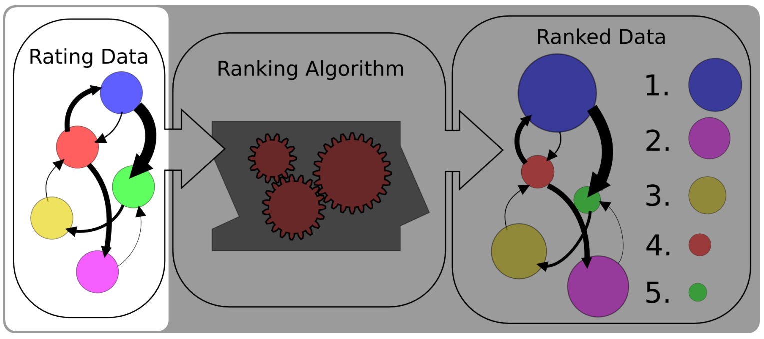
In section 2 the mathematical preliminaries are developed to examine different multipartite graphs. The *partition graph* of a multipartite graph is defined. This will be helpfull in the further analysis of multipartite graphs. We will show that some important properties of a multipartite graph can be contained within the partition graph. A main result of this thesis is given in lemma 2, and the following theorem 3. Here we will give a relation of a multipartite graph to its partition graph, with regard to eigenvector based ranking. With these results we will show briefly by what means eigenvector based ranking can be applied to multipartite graphs.

In section 3 we will look at different ranking algorithms which have already been applied to different kinds of multipartite graphs, and examine them by the means of mathematical groundwork. The focus here is on the different algorithmic approaches. The different kinds of multipartite graphs for which these algorithms were designed are examined.

In section 4 a new eigenvector based ranking algorithm, the $A_n - H_n$-Rank is developed. This new algorithm is applicable to different kinds of multipartite graphs. Its design is intended to be as general as possible, with regard to the three mentioned problems. We will see that the new ranking is well defined and an iterative algorithm for its computation will be given.

In section 5 the $A_n - H_n$-Rank is applied to the data gathered in the eval study at the Ludwig-Maximilians-Universität Munich described in [15]. The properties of the new ranking algorithm are evaluated.

2 Mathematical Preliminaries



The basic mathematical structure of this section will be a graph $G = (V, E)$ where $V = v_1, ..., v_n$ denotes the set of vertices and $E \subset V \times V$ denotes the set of directed edges in the graph. As weighted edges are considered, the weight of an edge is denoted by $e \in E$ as $\omega(e)$. It is assumed, that all edge weights are non-negative real numbers. If two vertices p, q have no connecting edge the weight $\omega(p, q)$ is set to be zero.

By this last definition the adjacency matrix M of G can be defined. It is a $n \times n$ Matrix with entries:

$$M_{i,j} = \omega(v_i, v_j) \tag{1}$$

2.1 Definition and Properties of Multipartite Graphs

We will start with a proper definition of a multipartite graph.

Definition 1. A Graph (V, E) with n vertices is called *multipartite* if we can find a partition $P_1 \dots P_p$ of the vertex set which provide:

$$\forall p \in P_i, q \in P_j \text{ with } \ (p, q) \in E \Rightarrow i \neq j$$

The sub-matrix of the adjacency matrix A, which represents the links from part P_i to P_j, vertices is called $A_{i,j}$.

A part is often referred to as "*independent* subset" of the vertex set. A single vertex is always independent.

Consider a multipartite graph $G = (V, E)$ with π partitions. Let the vertices from a part P_i with s_i vertices be given by $(p_{i1}, \dots, p_{is_i})$. By ordering the vertices of V partition wise $V = (p_{1_1}, \dots, p_{1_{s_1}}, \dots p_{\pi_1}, \dots, p_{\pi_{s_\pi}})$, the adjacency matrix of G is a block matrix, with blocks $A_{i,j}$.

Whether the adjacency matrix M of a multipartite graph is a block matrix or not, depends solely on the order of the vertices. Therefore it can be assumed that M is always given in block form.

Definition 2. A multipartite graph with p parts is called *cyclic-multipartite* if the vertices of each part P_i are only linked to vertices of part P_{i+1} for $i < p$, and vertices of P_p are only linked to vertices in P_1.

This leads to an adjacency matrix of the following form, which is referred to as the *cyclic-multipartite* form.

$$A = \begin{pmatrix} 0 & A_{(1,2)} & 0 & \dots & 0 \\ 0 & 0 & A_{(2,3)} & \dots & 0 \\ \vdots & \ddots & & \ddots & 0 \\ 0 & & & & A_{(p-1,p)} \\ A_{(p,1)} & 0 & \dots & & 0 \end{pmatrix}$$

Here a cyclic-multipartite graph with three parts is given, containing two, three and four nodes.

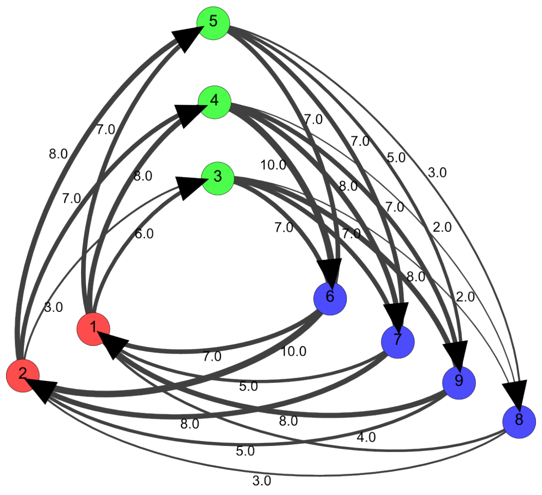


Figure 4: example of a cyclic multipartite graph

Its adjacency matrix is given by:

$$M = \begin{pmatrix} 0 & \begin{matrix} 6 & 8 & 7 \\ 3 & 7 & 8 \end{matrix} & 0 \\ 0 & 0 & \begin{matrix} 7 & 7 & 2 & 8 \\ 10 & 8 & 2 & 7 \\ 7 & 7 & 3 & 5 \end{matrix} \\ \begin{matrix} 7 & 10 \\ 5 & 8 \\ 4 & 3 \\ 8 & 5 \end{matrix} & 0 & 0 \end{pmatrix}$$

And it can be seen that M shows truly the form given below definition 2.

Definition 3. A multipartite graph is called complete-multipartite if for each pair of parts $P,\ Q\ \ (P \neq Q)$ there are two nodes $p \in P$ and $q \in Q$ with $(p, q) \in E$

Remark. Complete multipartite graphs are often denoted by $K_{s_1,\dots,s_p}$ where p is the number of parts and s_i the size of part i.

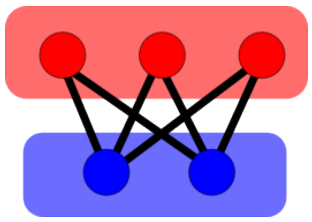


Figure 5: The graph $K_{3,2}$

Note that multipartite graphs cannot be complete in the common sense, which means that each node is connected with every other node. But complete graphs provide very useful properties for the computation of eigenvalue based ranking algorithms. Therefore a weaker property for a multipartite graph can be employed.

Definition 4. A multipartite graph is called *dense-multipartite* if for each pair of parts $P,\ Q\ (P \neq Q)$ satisfies the following implication:

$$\forall\ \tilde{p} \in P,\ \ \tilde{q} \in Q : (\tilde{p}, \tilde{q}) \in E \Rightarrow \forall p \in P, q \in Q\ (p, q) \in E$$

In this case the entries of any block $A_{i,j}$ are either all positive or all equal to zero.

This property of the adjacency matrix can be obtained with techniques described in the next subsection. We will see that this property is useful for eigenvalue based ranking algorithms on multipartite graphs.

It is obvious that there are many kinds of different multipartite graphs that are neither cyclic- nor multipartite complete but with a different constellation of links between parts. To examine these different constellations another definition is used.

Definition 5. Let $G = (V, E)$ be a multipartite graph with p parts $P_1 \dots P_p$. We define its *partition graph* $G_p = (V_p, E_p)$ as a graph with p vertices $\pi_1 \dots \pi_p$, and set: If $p \in P_i, q \in P_j$ and $(p, q) \in E$ then $(\pi_i, \pi_j) \in E_p$.
Which defines all the edges in G_p.

The partition graph of a multipartite graph G can be considered as the result of two simplification steps on G:

1. For all parts, the vertices in that part are merged to one vertex. The resulting vertices are the vertices of V_p
2. All edges with the same root and end in the graph are merged. These edges are the edges in E_p

Not surprisingly these steps can be applied to all multipartite graphs, therefore each multipartite graph has a partition graph. First, the edges of G_p are considered as directed if the edges of G were directed. If the multipartite graph has weighted edges, the weight of the edge (π_i, π_j) is set to be the sum of all edge weights of edges connecting vertices of P_i to P_j.

$$\omega((\pi_i, \pi_j)) := \sum_{p \in P_i, Q \in P_j} \omega((p, q)) \tag{2}$$

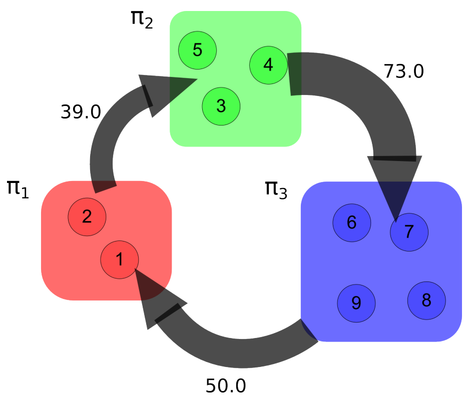


Figure 6: partition graph of example 1

This gives a fast tool to examine the basic structure of a multipartite graph.

The adjacency matrix M_p of this partition graph is given by

$$M_p = \begin{pmatrix} 0 & 39 & 0 \\ 0 & 0 & 73 \\ 50 & 0 & 0 \end{pmatrix}$$

The entries of M_p can be considered as absolute rating weights between the parts. Sometimes these values are not needed, therefore we define a simplification of M_p.

Definition 6. Let M be the adjacency matrix of a multipartite graph with p parts. Let M_p be the adjacency matrix of the partition graph. We define a $p \times p$ matrix $\overline{M_p}$ by:

$$(\overline{M_p})_{i,j} := \begin{cases} 1 & if \ \ (M_p)_{i,j} \neq 0 \\ 0 & else \end{cases}$$

The matrices M_p and $\overline{M_p}$ of multipartite graphs will be studied in this thesis. We will show that they have many properties in common with the original adjacency matrix M.

2.2 Mathematics of Eigenvector Centrality and Related Rankings

Eigenvector centrality is often associated with the *importance* of a certain vertex in a graph. Here *importance* is usually expressed by a positive real number. Possible meanings of the "importance of a vertex" can be seen by many different applications of eigenvector centrality in a variety of different domains. [16]

All eigenvector centrality measures define their ranking as an eigenvector v of a matrix A. Here the entry i of v holds the ranking of the vertex i of the graph. The matrix A is usually obtained by various manipulations of the adjacency matrix M of the graph. The most common definitions and theorems related to this topic are shown below.

Definition 7. An eigenvalue λ of a matrix M is called *dominant* if $|\lambda| >= |\mu|$ for all eigenvalues μ if M. If $|\lambda| > |\mu|$ holds, λ is called *strictly dominant.*

Definition 8. A real, non negative $n \times m$ Matrix $M = m_{i,j}$ is called *stochastic* or *column stochastic* if the sum over each column is one:

$$\sum_{i=1}^{n} m_{i,j} = 1 \quad \forall j \in \{1, \dots, m\}$$

It can be shown that 1 is a dominant eigenvalue of each column stochastic square matrix. The product of two column stochastic matrices is again column stochastic. A non-negative vector whose sum over all entries is equal to one, is normed according to the sum norm $||.||_1$. Because only non negative vectors are considered in this thesis, we may call a vector *normed* if the sum over all its entries is equal to one. The product of such a normed vector and a column stochastic matrix is again normed.

Definition 9. A non negative matrix M is called *primitive*, if A^m is positive for some $m \in \mathbb{N}$

A dense multipartite graph is primitive if and only if its partition graph is primitive. So primitiveness is maintained in the partition graph.

Another property important for eigenvector centrality is the *irreducibility* of a graph.

Definition 10. A graph is called *irreducible*, if for each pair of vertices p, q there is an directed path from p to q. Otherwise a graph is called *reducible.*

An irreducible graph is also called *strongly connected.* Obviously, irreducibility is a weaker property than primitivity. A dense multipartite graph is irreducible if and only if its partition graph is irreducible. Therefore also the irreducibility of graph is also maintained in the partition graph.

Definition 11. An eigenvalue λ is called *simple*, if its algebraic and geometric multiplicity are equal to 1.

Theorem 1. (Perron-Frobenius) If M is a primitive square matrix, then there is a real eigenvalue λ of M such that:

1. λ is positive and simple.

2. λ has an positive eigenvector v. Any other positive eigenvector of M is a multiple of v.

3. λ is strictly dominant.

There are many formulations of this theorem, and some of them are easier to prove than others. Another important formulation states that if a Matrix A is irreducible and non-negative, then it has a positive real dominant eigenvalue λ with an positive real eigenvector. This formulation is a weaker statement because the weaker condition *irreducible* is rewarded with a weaker conclusion of the existence of an *dominant* (but not *strictly dominant*) eigenvalue.

If such a dominant and positive eigenvector of a matrix M exists, it is referred to as "Perron vector", following the notation of [4]. Many eigenvector centrality measures are based on the computation of the Perron vector.([5], [4], [2]). The Perron vector can be found by applying an iterative algorithm known as *power iteration* or *Von Mises iteration*.

Theorem 2. (Von Mises Iteration) Let M be a square matrix with a strictly dominant and simple eigenvalue λ and an eigenvector v associated with λ. If v_0 is not orthogonal to v then the following recursively defined series v_k converges to a vector v_∞.

$$v_{k+1} = \frac{M \cdot v_k}{|M \cdot v_k|}, \quad k > 0$$

And

$$v_\infty = \tfrac{v}{|v|}$$

Remark.
1. It can be shown, that if λ is only *dominant* (and not *strictly dominant*) then the series does not necessarily converge, but exhibits a converging subsequence which converges to v_∞.
2. If M is chosen to column stochastic, and v_0 is normed, then $M \cdot v_0$ is also normed, and the series converges without normalisation at each step.

The properties *column stochastic* and *positive* are valuable for the computation of eigenvectors. But usually an adjacency matrix of an arbitrary graph will not provide these properties. Often the following or a similar matrix manipulation is proposed to achieve the desired result.

Lemma 1. Let M be a non-negative $n \times m$ Matrix and $\alpha \in]0,1[$. Consider the matrix $\widetilde{M}$ given by:

$$\widetilde{M_{i,j}} = M_{i,j} \frac{\alpha}{\sum\limits_{k=1}^{n} M_{k,j}} + \frac{1-\alpha}{n}$$

Then $\widetilde{M}$ is positive and column stochastic.

A proof of this lemma is given in [15]. The factor α is often referred to as *damping factor*. We will refer to the function $f : M \to \widetilde{M}$ as *damping function*.

Note, that the damping procedure is not applicable, if M has a column that has only zero entries. This case occurs if a vertex has no incoming edge. There are different measures discussed to deal with this problem, one of these is described in [15]. It is assumed that our use case does not exhibit this problem, or that it has been solved in advance.

Example. Here a 4×4 matrix M and $\widetilde{M}$ computed with a damping factor of 0.85 are given.

$$M = \begin{pmatrix} 0 & 4 & 0 & 8 \\ 1 & 0 & 4 & 0 \\ 0 & 8 & 1 & 0 \\ 7 & 0 & 7 & 9 \end{pmatrix} \to \widetilde{M} = \begin{pmatrix} 0.0375 & 0.32083 & 0.0375 & 0.4375 \\ 0.14375 & 0.0375 & 0.32083 & 0.0375 \\ 0.0375 & 0.60417 & 0.10833 & 0.0375 \\ 0.78125 & 0.0375 & 0.53333 & 0.4875 \end{pmatrix}$$

A small damping factor is considered to have a small impact on the ranking. The influence of a damping factor is often examined in the literature. The factor $(1-\alpha)$ is sometimes called random leap factor.

2.2.1 Eigenvector Centrality on Multipartite Graphs

In this section it is be examined how the eigenvector centrality can be applied to multipartite graphs, and how the necessary properties of the adjacency matrix can be provided.

The damping function is of course also applicable to the adjacency matrix of any multipartite graph, but it would destroy its multipartite structure: If the adjacency matrix of a graph is positive, then the according graph is complete. And therefore there were edges between the vertices of the same part.

To preserve the multipartite structure of the graph, a *block wise* damping method is proposed, which can be applied to an adjacency matrix of a multipartite graph.

$$M = \begin{pmatrix} 0 & A_{1,2} & \dots & A_{1,p} \\ A_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & A_{p,p-1} \\ A_{p,1} & \dots & A_{p-1,p} & 0 \end{pmatrix} \xrightarrow[\textit{damping}]{\textit{blockwise}} M_d := \begin{pmatrix} 0 & \widetilde{A_{1,2}} & \dots & \widetilde{A_{1,p}} \\ \widetilde{A_{2,1}} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \widetilde{A_{p,p-1}} \\ \widetilde{A_{p,1}} & \dots & \widetilde{A_{p-1,p}} & 0 \end{pmatrix}$$

As seen in the last subsection, it is assumed that each of the columns of the blocks $A_{i,j}$ are positive.

If this method is applied to the adjacency matrix of the graph in example 1, we obtain:

$$M = \begin{pmatrix} 0 & \begin{matrix} 0.64 & 0.53 & 0.47 \\ 0.36 & 0.47 & 0.53 \end{matrix} & 0 \\ 0 & 0 & \begin{matrix} 0.3 & 0.32 & 0.29 & 0.39 \\ 0.4 & 0.36 & 0.29 & 0.35 \\ 0.3 & 0.32 & 0.41 & 0.26 \end{matrix} \\ \begin{matrix} 0.29 & 0.36 \\ 0.21 & 0.3 \\ 0.18 & 0.14 \\ 0.32 & 0.2 \end{matrix} & 0 & 0 \end{pmatrix}$$

By this method a dense multipartite graph is obtained. It is not column stochastic in general, but each column of the matrix sums to the number of overlying blocks in that column. A column stochastic matrix can be easily obtained by dividing each column by that number.

Using blockwise damping or similar methods has already been discussed in other approaches [15] [8].

This damping method can assure the existence of a Perron vector in certain cases: If M_p is primitive, then M_d is primitive and the Perron-Frobenius theorem holds for M_d.

The property *"dense multipartite"* is not always sufficient for the existence of a strictly dominant eigenvalue. For instance the simple 2 cycle has two parts and is *dense multipartite*. Its adjacency matrix is given by $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. It is column stochastic and has two eigenvalues 1 and -1 which are both dominant.

What can be stated about the eigenvalues and eigenvectors of the matrix M_d? Within this subsection a necessary condition for the existence of an Perron vector of M_d is given. To accomplish this, the eigenvectors of $\overline{M_p}$ are examined.

If v is a ranking vector of a multipartite graph, the *ranking weight of a part P_i* is set to be $w(P_i) = \sum\limits_{j \in P_i} v_j$. The ranking weight of a part is the sum off all rankings that vertices from this part obtained. For brevity we will the vector that contains all partition weights is denoted by $\overline{v}$.

Definition 12. For a ranking vector v of a multipartite graph we set:

$$\overline{v} = \begin{pmatrix} w(P_1) \\ w(P_2) \\ \vdots \\ w(P_p) \end{pmatrix}$$

By definition, $\overline{\lambda v} = \lambda \overline{v}$ holds. Therefore the function $v \to \overline{v}$ is linear. This function is referred to as "weight contraction".

Example. Let v be an arbitrary ranking vector of the block wise damped adjacency matrix from the last example. $\overline{v}$ can easily be found.

$$v = \begin{pmatrix} 0.52161 \\ 0.43073 \\ 0.31176 \\ 0.34276 \\ 0.29782 \\ 0.30584 \\ 0.24073 \\ 0.15185 \\ 0.25391 \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \end{matrix}\right\} w(P_1) = 0.95234 \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} w(P_2) = 0.95234 \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} w(P_3) = 0.95234 \end{matrix} \qquad \overline{v} = \begin{pmatrix} 0.95234 \\ 0.95234 \\ 0.95234 \end{pmatrix}$$

The weight contraction $\overline{v}$ is equally distributed.

The following Lemma shows another important property of the weight contraction. This lemma leads directly to the eigenvalues of M_d.

Lemma 2. Let M be the adjacency matrix of a multipartite graph and M_d be defined as above. If v is a column vector of length n then

$$\overline{M_d \cdot v} = \overline{M_p} \cdot \overline{v}$$

Proof. The product $M_d \cdot v$ can be expressed with the block matrices $\widetilde{A_{(i,j)}}$. For simplification we set $\widetilde{A_{(i,:)}} := (\widetilde{A_{(i,1)}}, \dots, \widetilde{A_{(i,p)}})$ which is a $s_i \times n$ matrix.

$$\begin{aligned} M_d \cdot v &= \begin{pmatrix} \widetilde{A_{(1,1)}} & \dots & \widetilde{A_{(1,p)}} \\ \vdots & & \vdots \\ \widetilde{A_{p,1}} & \dots & \widetilde{A_{p-1,p}} \end{pmatrix} \cdot v \\ &= \begin{pmatrix} \widetilde{A_{(1,:)}} \cdot v \\ \vdots \\ \widetilde{A_{(p,:)}} \cdot v \end{pmatrix} \end{aligned}$$

$\widetilde{A_{(i,:)}} \cdot v$ is a $s_i \times 1$ vector. Now, let us consider the first entry of $\overline{M_d \cdot v}$.

$$\begin{aligned} (\overline{M_d \cdot v})_1 &\overset{def}{=} \sum_{i=1}^{s_1} \left(\widetilde{A_{(1,:)}} \cdot v \right)_i \\ &= \sum_{i=1}^{s_1} \left(\widetilde{A_{(1,:)}} \right)_i \cdot v \end{aligned}$$

This is possible by considering a *column wise* sum over the entries of $\widetilde{A_{(1,:)}}$. Then the equality follows with the distributivity of the dot product.

$$\begin{aligned} ... &= \sum_{i=1}^{s_1} \left(\widetilde{A_{(1,1)}}_i, \quad \dots \quad , \widetilde{A_{(1,p)}}_i \right) \cdot v \\ &= \left(\sum_{i=1}^{s_1} \widetilde{A_{(1,1)}}_i, \quad \dots \quad , \sum_{i=1}^{s_1} \widetilde{A_{(1,p)}}_i \right) \cdot v \end{aligned}$$

Now consider the column wise sum $\sum_{i=1}^{s_1} \widetilde{A_{(1,l)}}_i$. By definition each column of $\widetilde{A_{(1,l)}}$ sums up to 1 if $(\overline{M_p})_{1,l} = 1$, and to 0 if $(\overline{M_p})_{1,l} = 0$. If 1_m is a row vector $\overbrace{(1, \dots, 1)}^{m}$ of length m, the last expression can be simplified.

$$... = \left(1_{s_1} \cdot (\overline{M_p})_{1,1}, \quad 1_{s_2} \cdot (\overline{M_p})_{1,2}, \quad \dots \quad , 1_{s_p} \cdot (\overline{M_p})_{1,p} \right) \cdot v$$

Which is a partial sum over the entries of v, and is equal to the first entry of $\overline{M_p} \cdot \overline{v}$ □

This technical lemma allows the following theorem.

Theorem 3. If v is an eigenvector of M_d with eigenvalue λ, then $\overline{v}$ is an eigenvector of $\overline{M_p}$ to the same eigenvalue.

Proof. By condition, we have

$$M_d \cdot v = \lambda v$$

Now the weight contraction is applied to both sides of the equation.

$$\begin{aligned} \overline{M_d \cdot v} &= \overline{\lambda v} \\ \overset{Lemma}{\Rightarrow} \overline{M_p} \cdot \overline{v} &= \lambda \overline{M_p} \end{aligned}$$

□

Remark. 1. If M_d has an positive eigenvector v, then $\overline{v}$ is also positive. Therefore a necessary condition for existence of an Perron vector of M_d is discovered.

2. An entry of $\overline{v}$ can be regarded as the centrality of a certain part. In several cases (for instance complete or cyclic multipartite) the centralities of the different parts are equal.

3. With this theorem a measure is found to know the average ranking of the vertices in a certain part, before the ranking is done. Consider for instance a normed ranking vector v with uniquely distributed weight contraction $\overline{v}$. By the theorem the vertices in a part of size s will obtain a mean ranking of $\frac{1}{p \cdot s}$. This can be used to compare the rankings of vertices in different parts.

4. Note that M_d may have much more eigenvalues as $\overline{M_p}$. The theorem holds never the less, because $\overline{v}$ can be equal to 0. This causes a degenerate case. We can observe this in the next example.

Example. Consider the matrix M_d from the last example. Its eigenvalues and eigenvectors are given by:

values	-0.5+0.87i	-0.5-0.87i	1	-0.05+0.09i	-0.05-0.09i	0.1
vectors	$\begin{pmatrix} -0.52 \\ -0.43 \\ 0.16-0.27i \\ 0.17-0.3i \\ 0.15-0.26i \\ 0.15+0.26i \\ 0.12+0.21i \\ 0.08+0.13i \\ 0.13+0.22i \end{pmatrix}$	$\begin{pmatrix} -0.52 \\ -0.43 \\ 0.16+0.27i \\ 0.17+0.3i \\ 0.15+0.26i \\ 0.15-0.26i \\ 0.12-0.21i \\ 0.08-0.13i \\ 0.13-0.22i \end{pmatrix}$	$\begin{pmatrix} 0.52 \\ 0.43 \\ 0.31 \\ 0.34 \\ 0.3 \\ 0.31 \\ 0.24 \\ 0.15 \\ 0.25 \end{pmatrix}$	$\begin{pmatrix} -0.2+0.35i \\ 0.2-0.35i \\ -0.17-0.3i \\ 0.15+0.26i \\ 0.02+0.04i \\ -0.31 \\ -0.33-0i \\ 0.17-0i \\ 0.47 \end{pmatrix}$	$\begin{pmatrix} -0.2-0.35i \\ 0.2+0.35i \\ -0.17+0.3i \\ 0.15-0.26i \\ 0.02-0.04i \\ -0.31-0i \\ -0.33 \\ 0.17 \\ 0.47-0i \end{pmatrix}$	$\begin{pmatrix} 0.4 \\ -0.4 \\ 0.34 \\ -0.3 \\ -0.04 \\ -0.31 \\ -0.33 \\ 0.17 \\ 0.47 \end{pmatrix}$

Table 1: Eigenvalues of M_d

If the weight contraction is applied to the eigenvectors we obtain:

values	-0.5+0.87i	-0.5-0.87i	1	-0.05+0.09i	-0.05-0.09i	0.1
vectors	$\begin{pmatrix} -0.95-0i \\ 0.48+0.83i \\ 0.48-0.82i \end{pmatrix}$	$\begin{pmatrix} -0.95-0i \\ 0.48-0.83i \\ 0.48+0.82i \end{pmatrix}$	$\begin{pmatrix} 0.95 \\ 0.95 \\ 0.95 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

Table 2: Weight contractions of the eigenvectors of M_d

And the eigenvalues and eigenvectors of $\overline{M_p}$ are given by:

-0.5+0.87i	-0.5-0.87i	1
$\begin{pmatrix} 0.58 \\ -0.29+0.5i \\ -0.29-0.5i \end{pmatrix}$	$\begin{pmatrix} 0.58-0i \\ -0.29-0.5i \\ -0.29+0.5i \end{pmatrix}$	$\begin{pmatrix} -0.58 \\ -0.58 \\ -0.58 \end{pmatrix}$

Table 3: Eigenvectors of $\overline{M_p}$

which are all multiples of the according contracted eigenvectors of M_d.

Here, a necessary condition for the existence of valuable ranking vectors on multipartite graphs is found, if the mentioned damping method is applied. Other necessary conditions are be the primitivity irreducibility of the graph.

Subsequently the question arises if there are multipartite graphs that are reducible but where the new condition holds. This question is not trivial, considering the multitude of partition graphs and matrices $\overline{M_p}$ that need to be examined. But the calculation for graphs with three parts can be done easily.

First all possible partition graphs need to be found. Any matrix $\overline{M_p}$ can be given by

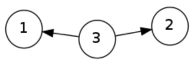
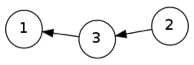
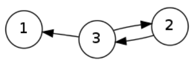
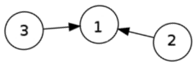
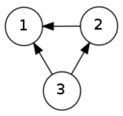
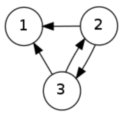
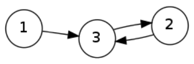
$$\overline{M_p} = \begin{pmatrix} 0 & a & b \\ c & 0 & d \\ e & f & 0 \end{pmatrix}$$

where a, b, c, d, e, f are binary digits. There are $2^6 = 64$ possible matrices. These matrices can be enumerated by writing all entries in one line and eliminating all zeros that where found on the diagonal. The obtained 6 binary digits form a binary number. The following scheme illustrates this procedure:

$$\begin{pmatrix} \boxed{0} & 1 & 0 \\ 0 & \boxed{0} & 1 \\ 1 & 0 & \boxed{0} \end{pmatrix} \rightarrow \begin{pmatrix} \boxed{0} & 1 & 0 & 0 & \boxed{0} & 1 & 1 & 0 & \boxed{0} \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}_2 \rightarrow 38_{10}$$

A tripartite cyclic graph would obtain the number 38 in our numeration. The number of a complete tripartite graph is 63. By this means, all possible matrices have found their unique number.

Many of the 64 partition graphs under consideration are isomorphic to each other. Two graphs are isomorphic if they are equal except for the numeration of the vertexes. For instance a 3-cycle (No.38) and its transpose in circle in opposite direction(No.25) are isomorphic. From each equivalence class of isomorphic graphs only one representative has to be considered, which reduces the number of matrices that have to be examined. Further we can eliminate all graphs where one vertex is separated from the others because it has no outgoing or incoming edge. This leaves us with a manageable number of different graphs.

No.	Graph
3	
6	
7	
10	
11	
15	
21	

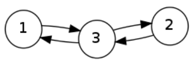
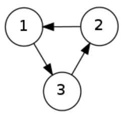
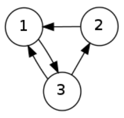
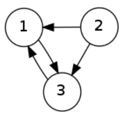
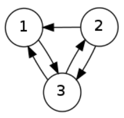
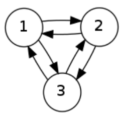
No.	Graph
23	
25	
27	
30	
31	
63	

Table 4: Partition graphs with undirected edges and the according ranking

These graphs may be called *regular* because all graphs with three vertices are either isomorphic to one of these graphs or degenerate.

The next figure shows a Venn diagram containing all regular graphs denoted by their numeration.

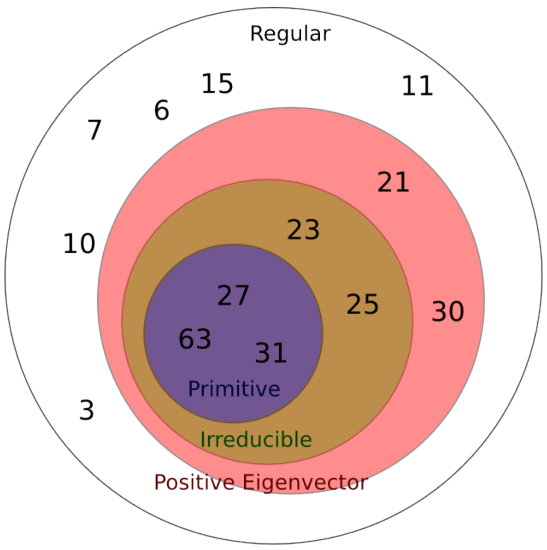


Figure 7: The set of regular, irreducible and primitive graphs, and the set of graphs whose adjacency matrices have positive eigenvectors.

There are two reducible graphs that have a positive eigenvector. Therefore there are two reducible tripartite graphs where eigenvalue based ranking is could be applicable.

If this remains true for multipartite graphs with 4 or more parts is unknown.

2.2.2 HITS

Jon Kleinbergs ranking algorithm HITS [13] can be adapted to suit the problem of ranking a bipartite graphs in 3.3.4. But we will take a closer look at the original HITS algorithm first. Kleinberg proposes a query-dependent ranking algorithm for web pages of the world wide web. As usual in this domain, web pages are considered as nodes and hyperlinks are modelled as directed and not weighted edges.

The query dependency is realised in the first step of the algorithm by computing a sub graph of the world wide web that contains only pages that are considered as relevant to the query. The further computation is then done on this sub graph. This first focusing step reduces the computational cost of the algorithm immensely, but is not crucial for its convergence. As a global idea is examined, this first step can be ignored. Every node in the graph is considered as relevant. The basic intuition of the following ranking is that each node of the graph has two rank able qualities: the hub and the authority score. A node is a good hub (with big hub score) if it links to many good authorities. And vice versa a node is a good authority, if it receives links from many good hubs. In terms of web sites a good hub would be a link collection, and a web page with a high authority score would be mentioned on many of these good link collections.

Consider the HITS equations which define the hub score h_i and the authority score a_i of vertex i in the graph:

$$h_i = \delta \sum_{j=1}^{n} A_{ij} a_j$$
$$a_i = \lambda \sum_{k=1}^{n} A^T_{ik} h_k$$

Here, A is the adjacency matrix and λ and δ are appropriate numbers.

By defining h as the collection vector of all hub scores $h = (v_1, ...v_n)^T$, and a as the collection vector of all authority scores $a = (a_1, ...a_n)^T$ these equations can be simplified to:

$$h = \lambda A \cdot a$$
$$a = \delta A^T \cdot h$$

And finally the eigenvector equations of the matrices $M := AA^t$ and $N := A^t A$ are obtained.

$$h = \delta\lambda A A^T \cdot h$$
$$a = \lambda\delta A^T A \cdot a$$

By setting $\epsilon := \frac{1}{\lambda\delta}$ we can simplify:

$$\begin{aligned} hM &= \epsilon M \\ aM &= \epsilon N \end{aligned}$$

M and N are symmetric:

$$\begin{aligned} M^t &= (AA^t)^t \\ &= A^{t^t}A^t \\ &= AA^t \\ &= M \end{aligned}$$

and:

$$\begin{aligned} N^t &= (A^tA)^t \\ &= A^tA^{t^t} \\ &= A^tA \\ &= N \end{aligned}$$

In linear algebra it can be shown that real, symmetric, and non negative matrices have only real and orthogonal eigenvalues, [3]. So the eigenvalue mentioned in the HITS-equation is a real, positive number. ϵ is set to be the biggest eigenvalue of M, and therefore h as the eigenvector to the biggest eigenvalue of M. It can be shown that the principal eigenvector h of M has only non negative entries [13]. h is therefore a good ranking vector without negative entries that would hinder a sense full interpretation of ranking.

But is a also a principal eigenvector? Note that a has only non negative entries. This follows directly from the definition of a.

All eigenvectors of N are orthogonal, so if ϵ is not the biggest eigenvalue of N then a has a negative entry. But this is impossible.

2.3 About Minimal Partitioning and Colouring

In graph theory multipartite graphs with p parts are also called p-chromatic or p-colourable. A good introduction to this subject is given in [17]. The *colouring problem* of graphs is an often treated subject in graph theory. In its simplest from graph colouring consists in colouring the vertices of a graph with different colours, such that no two adjacent vertices share the same colour. All vertices of a specific colour form a part according to our definition 1, therefore *colouring* and *partitioning* can be considered as equal. In the common literature a part is often referred to as *independent subset* vertex set of the graph [17].

Usually there is more than one colouring of a graph. Each vertex could be coloured uniquely, which would lead to a partition graph that is equal to the original graph. Therefore each graph is multipartite, but in most use cases the number of parts is considered to be much lower than the number of vertexes.

The actual graph colouring problem consists in finding a minimal colouring of the graph. A p-colouring of a graph is minimal if there is no q-colouring with $q < p$. The

minimal amount of colours that is needed for a graph colouring is called *chromatic number* and is often denoted by $\chi(G)$ where G is the graph.
The computation of a minimal colouring is NP-hard, such as the computation of the chromatic number. There are algorithms as the *greedy colouring algorithm* that do not always compute a minimal colouring, but one where generally few colours are used.

Finding a minimal colouring of a graph is not treated in this thesis, but it can be interesting in real life applications. For instance a network could be searched for different parts that were not evident or planned.

In this thesis *directed* multipartite graphs are often considered. The colouring of directed graphs is not often discussed, but it can differ from an usual colouring. In [10] for instance, two vertices A and B have to receive different colours if there is a directed path from A to B and from B to A. It can be shown that this definition leads to a valuable generalisation of the colouring problem on directed graphs.

However, we set that two vertices A and B will have to be coloured differently if there is an edge from A to B *or* from B to A. Then each colouring of a directed graph can be reduced to undirected colouring, where the edge directions are simply ignored.

Note that the partitioning of our input graphs can not be considered as minimal. For instance, a cyclic multipartite graph has a chromatic number of 2 if it has an even number of parts, and 3 if its number of parts is odd and bigger than one. So cyclic multipartite graphs with more than 3 parts are not minimally partitioned.

This can be easily shown. For simple cyclic graph the proposition can be shown by two inductions over the odd and the even numbers: A two cycle has a chromatic number of 2. And by adding two vertices to an even cycle no further colour is needed than the two that exist by induction.
A three cycle has a chromatic number of 3. By adding two vertices to an odd cycle no more than two of the three existing colours are needed.
The same holds for cyclic multipartite graphs and the proof can be done similarly.

A complete- and dense multipartite graph $G = K_{\pi_1 \ldots \pi_p}$ has already a minimal colouring. Each part can be coloured uniquely which shows that p is bigger or equal to the chromatic number of G. Now it has to be shown that the graph is at least p-colourable. Let us consider a complete sub graph, where a vertex is taken from each part. This graph is complete and has p vertexes, therefore its chromatic number is p. Therefore the chromatic number of G is at least p. A similar proof is found in [19].

Theorem 4. If a multipartite graph is minimally partitioned, then its partition graph is complete.

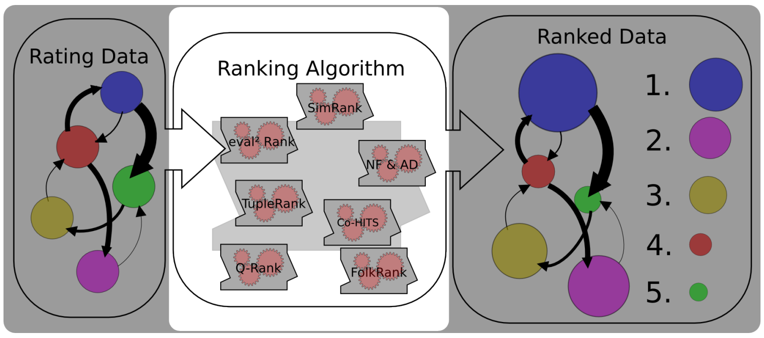
Proof. If the partition graph is not complete, then it has two vertices π_i, π_j that are not connected. Then the vertices of P_i and P_j are pairwise unconnected. Hence $P_i \cup P_j$ is independent. $\square$

A last proposition can be stated:

Theorem 5. Let G be a graph with (not necessarily minimal) parts $P_1 \ldots P_p$. If G_p is the partition graph of G with vertices $\pi_1 \ldots \pi_p$, then $\chi(G) \leq \chi(G_p)$.

Proof. Let us consider G_p as minimally coloured. Each vertex $v \in P_i$ of G can be coloured equal to the according partition vertex π_i in G_p. This gives a valid colouring of G due to the independence of the parts. Therefore $\chi(G) \leq \chi(G_p)$ holds. □

3 Related Ranking Approaches



In this section different ranking algorithms that are applied to multipartite graphs are examined. First, general ranking approaches are presented. Then the different graphs that serve as input for the algorithms are classified. Further, these algorithms are examined in more detail.

3.1 General Approaches

In section 1 the difference between *query depend* an *global* ranking approaches has been mentioned. Another classification of ranking algorithms is done by examining their *defining approach*, that is used to define a ranking.

The earlier discussed ranking approach of using *eigenvectors* of a somehow processed adjacency matrix as a ranking is one of the general ideas. This is described by Bonacich [1] et al. In this section, several eigenvector based ranking algorithms will be examined, that are applied to multipartite graphs.

Another very popular approach is the *Random Surfer* approach. Here, ranking is seen as a result of a stochastic process. One popular application of this idea is proposed by Brin and Page in [2]: The *PageRank* algorithm. The PageRank of a web page represents the probability that a person would get to this page by surfing randomly surfing from page to page. The *PageRank* can also be defined as an eigenvector centrality: The stochastic process of surfing randomly through the web can be modelled as a *Markov chain*. The *PageRank* is therefore the steady state distribution of this Markov chain. The transition matrix of this process is often called Google Matrix. To find the principal eigenvector of the Google Matrix the *Van Mises Iteration* can be used. And it can be shown that this eigenvector is exactly the *PageRank*. This shows, that the *eigenvector centrality* approach and the *Random Surfer* approach can sometimes lead to the same ranking. Certain ranking algorithms will be examined that are based on the *Random Surfer* approach (see 3.3.7, 3.3.3, 3.3.5).

The last approach is referred to as the *update rule* approach. Here, every rankable object carries its rank, which gets continuously updated by a specific rule. Obviously every ranking algorithm can be presented in this form, trivially by assigning the definition of an objects ranking to the object. For example, this general approach is used to define the HITS algorithm [13].

As seen, some of the algorithms that follow one approach can also be defined by another. It is likely that this also holds for some of the algorithms examined in this section. The following scheme shows a division of different ranking algorithms by their defining approach.

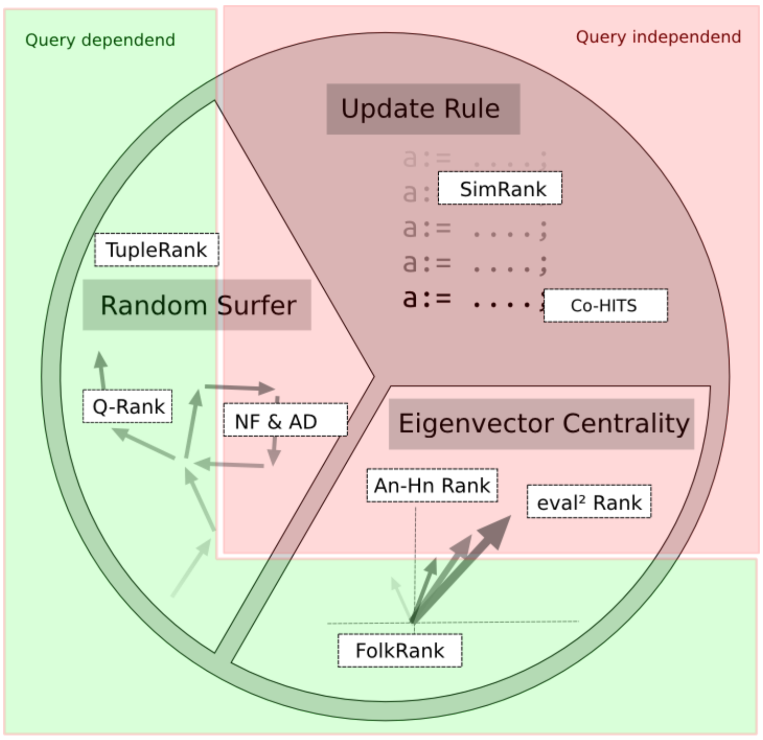


Figure 8: Examined ranking approaches with their defining approach

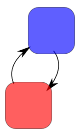
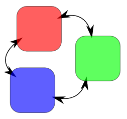
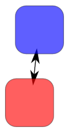
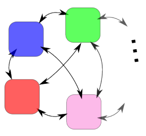
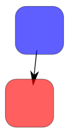
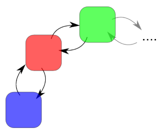
3.2 Characterisation of the Input Graphs

By looking at different ranking algorithms that are applied to multipartite graphs, the variety of different input graphs comes apparent. The input graphs can be classified by the properties of their edges.

Edge Direction	Edge Weight	Section	Algorithm Name
bidirectional	numerical	3.3.6	FolkRank
	binary	3.3.3	Tuple Rank
		3.3.7	NF & AD
unidirectional	numerical	3.3.1	*eval*^{2} Rank
		3.3.4	Co-HITS
		3.3.5	QRank
	binary	3.3.2	SimRank

Table 5: The examined ranking algorithms, classified by the edge properties of the according graphs

Now the different partition graphs are examined. Most of them can be categorised by the special cases *multipartite cyclic* and *multipartite complete* that have been defined in chapter 2.

Partition Profile	Algorithm	Number of Parts	Partition Graph
cyclic & complete multipartite	*eval*^{2}Rank	2	
	Co-Hits		
	FolkRank	3	
	NF & AD	2	
complete multipartite	TupleRank	n, applied for n = 3	
asymmetric	SimRank	2	
	QRank	n	

directed edges: $\longrightarrow$	undirected edges: $\longleftrightarrow$

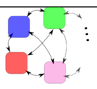
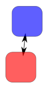
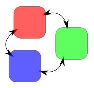
Table 6: Different partition graphs of the according ranking algorithms

Obviously, the most work has been done on ranking multipartite graphs with two or three parts. With these small part numbers many of our special cases collapse. For example, on a tripartite graph with undirected edges, there is no difference between cyclic and complete. The same holds for all bipartite graphs. For small part numbers our approach is valuable as it covers most of the here examined graphs. For higher part numbers the variety of different kinds of multipartite graphs diverges with a growing number of possible partition graphs.Only two of the examined algorithms treat this case of high part numbers. TupleRank described in [6] is only applied to a graph with three parts but is expendable to more parts. This graph in [6] matches the *multipartite complete* condition according to our definition, but it is not mentioned if this property is required for the algorithm to converge.

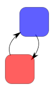
Another remarkable partition graph is used by QRank: It shows a chain like structure. In the tripartite case, it has been mentioned as No.23 from the last section.

Furthermore, it gets evident that most of the examined algorithms treat the two special cases *cyclic-* and *complete multipartite*, which both can be considered as symmetric. Ranking algorithms on graphs with asymmetric partition graphs have only been studied in two examples.

In the next tables it can be seen, that even for the special cases there are many untreated situations where no ranking algorithm is presented in this thesis. First we have a look at the graphs with undirected edges:

undirected edges	2 parts	3 parts	arbitrary number of parts
complete multipartite	NF & AD	FolkRank	TupleRank 
cyclic multipartite			?

Now the same for graphs with directed edges:

directed edges	2 parts	3 parts	arbitrary number of parts
complete multipartite	*eval*^{2} Rank,	?	?
cyclic multipartite	Co-Hits 	?	?

3.3 Related Ranking Algorithms in Detail

3.3.1 *eval*^{2}Rank

The *eval*^{2}Rank, proposed by Oster in *A Social Platform for Bipartite Student-Lecturer Ranking* [15]. It is strongly related to PageRank by its definition and iterative calculation method. In contrast to PageRank it is designed for bipartite graphs with weighted edges. It is used as a ranking on a bipartite rating graph. The basic assumption here is that the opinion of a generally good rated person (Lecturer or Student) should have a higher influence than the opinion of a bad rated person. The evaluation of the algorithm shows that, in a student-lecturer rating environment, this assumption leads to an useful ranking.

Of special interest for this thesis is the proposed normalisation of edge weights. By this normalisation the sum of the outgoing edge weights is normalised to 1, ensuring that a rating can always be regarded as a relative, not an absolute value. This can be considered as a solution for the *typed rating* problem mentioned in the introduction. To assure the convergence of the algorithm, a *block wise* damping method similar to one that was mentioned earlier is used.

Further, the special properties of the adjacency matrix of the bipartite ranking graph are used to reduce the calculation time of the ranking.

3.3.2 SimRank

Proposed by Jeh and Widom [12], this algorithm computes a measure of similarity for each pair of nodes in a graph. This similarity based on the structural embedding of this two nodes. The basic idea is that *"two objects are similar if they are related to similar objects"* [12], which leads finally to the definition of similarity between nodes as the average meeting distance of two random surfers starting at the two nodes in question, and following the edges of the graph. In its basic form SimRank is not particularly designed for bipartite graphs, but has been successfully adapted for bipartite situations. A mentioned example of such a bipartite situation are buyers that are more or less similar to each other if they buy similar items. Another example are students that are more or less similar if they attend similar lectures. Edge weights, that could model for instance buyer preferences, are mentioned but not included in the algorithm.

3.3.3 TupleRank

TupleRank is proposed in [6]. Here, the idea of random walks on a undirected and (initially) bipartite graph leads to the definition of *returning relations* that indicate whether two vertices of the same part are connected via a vertex in the other part. This additional information is stored in an extended directed graph with far more nodes. This graph is then used to compute a *relevance score* describing the relevance of one node toward the other nodes in the graph. This is accomplished by modelling a random surfer starting from the specific node in question, and restarting from it randomly with a certain restart probability.

By applying this idea to a tripartite graph, it is shown that different walking directions of the random surfer between the parts have an impact on the output of the algorithm. The direction to be preferred remains uncertain and multipartite graphs with more than 3 parts are not examined.

3.3.4 The Generalised Co-HITS Algorithm

This Algorithm developed by Deng, Lyu and King follows a similar idea as TupleRank. Relations between vertices of the same part of a bipartite graph are developed. These intra-part relations are found by solving two equations that can be considered as generalisations of the HITS equations seen in [13]. The mentioned generalisation consists in introducing two personalised parameters whose variation and optimisation is largely discussed [8]. Indeed, the HITS equations and other ranking algorithms emerge as special cases of this algorithm. Different variations are successfully applied to bipartite graphs with weighted edges. The ranking is not query dependent and is therefore imposing a global measure of centrality for all nodes in a certain part. Also the idea of weight normalisation and block wise damping is implemented.

3.3.5 QRank

Designed as a query dependent recommendation algorithm, QRank [7] is based once again on the random walk approach. It is applicable to a multipartite graph with any number of parts but is tested on a tripartite graph. The idea of restarting the random walk at a specific node with a certain restart probability is extended. Instead of investigating the centrality of one node, a query may content several nodes with different given weights signifying different importance for the query. This motivates the definition of a query vector that is used to compute a returning state for the random walk. In [7] a great effort is made to improve the efficiency of the algorithm. This is realised by clustering the multipartite graph depending on the query. It is worth mentioning that the example of a tripartite graph described by Cheng et al. exhibits a special partition structure that is neither multipartite-cyclic or multipartite-complete, while the algorithm is not necessarily restricted to this structure.

3.3.6 FolkRank

The idea of modelling social resource sharing tools as tripartite graphs, referred to as folksonomies, is presented in [11]. A folksonomy is here defined a graph containing nodes that represent the different users, tags and resources (e.g images) stored in a social platform. The undirected edges of such a graph are called tag assignments, following the intuition that a user would use a specific tag to label a specific resource. Edge weights are imposed on this graph by assigning higher weights to links that are proposed more often on the social platform.

A basic notion of *importance* is introduced: *"...a resource that is tagged with important tags by important users becomes important itself"*. Much like in (3.3.5) a personalised query vector is used to implement a topic-specific ranking which is strongly related to PageRank. The FolkRank is finally computed as the difference of a general and a topic-specific ranking.

The basic idea is to use edge weights, that are present in the system as tag multiplicities, to calculate node weights.

3.3.7 Neighbourhood Formation and Anomaly Detection

The Neighbourhood Formation algorithm proposed by Sun, Qu et al. in [18] tries to detect nodes in a bipartite graph that are *relevant* to a query node. Nodes with a high relevance score in the same part of the query node would then form the soft neighbourhood around this node. The proposed method for computing this relevance score is to model several random walks starting from the query node, comparable to 2.3.3, 2.2.5. If a node in the same part as the query node is reached by many random walks, it obtains a higher relevance score. An entire set of relevance scores of nodes in one part can then be used to propose a measure of anomaly detection in the other part, namely the normality score. While an anomaly is defined as a node with a very low normality score, a typical example of an anomaly would be a node that is linked to two disjoint neighbourhoods in the other part.

The anomaly detection algorithm is then evaluated by adding random edges to a given data set taken from different web applications. These injected random edges are shown to have a lower average normality score then the genuine edges.

4 The $A_n - H_n$ Ranking on Cyclic Multipartite Graphs

4.1 Motivation

Let us consider the ranking system of a university as described at the beginning, with lecturers, students and different courses. This system can be modelled to satisfy the Definition 2 of a cyclic-multipartite graph: There are lecturers giving grades to the students, students giving feedback about the courses that they attended, and lecturers receiving feedback from these courses. This situation is modelled in a graph, where each of these entities, namely the students, the lecturers and the courses will be represented by a node in the graph. Let us assume that the three different sets of entities are disjoint. This may be a good assumption as students are not likely to be courses or lecturers and vice versa. So these sets will form three parts of our graph.

For practical reason it will be assumed that any rating in this system is integer number between one and ten, with ten being regarded as the most and one as the least desirable rating. Each rating has a unique source, a numerical value and a unique receiver in a different part. Therefore the ratings in the system can be considered as weighted and directed edges of the tripartite graph. An example for an edge between the *student* part and the *course* parts may be a student who rates a course that he attended. Non existent edges in this graph are considered to have an edge weight of zero.

We will try to model a ranking algorithm for this graph, which basically means to use the existing edges and edge weights to calculate a ranking for the nodes of the graph. Other than many of the considered ranking algorithms in section 3 an approach that is not query dependent is proposed. This ranking should signify a general, in our example academic, quality of an entity represented by a positive numeric value. A high rating value could identify largely appreciated lecturers or courses as well as good achieving students.

Which criteria should a ranking on such a graph satisfy?

At first, it should be fair, and the basic concept should be understandable. This is very important, as social interactions are considered in our example.

It should take the special cyclic structure of the graph into account: For example difficult courses tends to be rated worse than an easier one, and the students usually succeed less, creating low weighted cycles in the graph. But a lecturer who gives mainly difficult courses should not be rated worse because of his choice of courses. Similar to the idea of low weighted cycles in the graph, one could also imagine that certain cliques of actors would try to push each others ranking by giving very high rankings to one another. While this scenario is rather unlikely to occur in the example of a university, it becomes more important by considering buyer recommender systems, where users and user ratings could be generated automatically at a large scale.

In our example it can be assumed, that the parts differ largely in size. (Regrettably there are much less lecturers than students). But in order to be fair, the ranking of a vertex should be independent from the size of the part in which the vertex is situated. The problem ranking-weight draining to larger parts has already been observed [15], and will be discussed later.

4.2 The Basic Approach

We will try to motivate an approach that is based on the HITS algorithm.

First we consider that the HITS algorithm was used to rank the nodes of a bipartite graph. We may call the two parts S and L. Each vertex in S and L receives its hub- and its authority score.

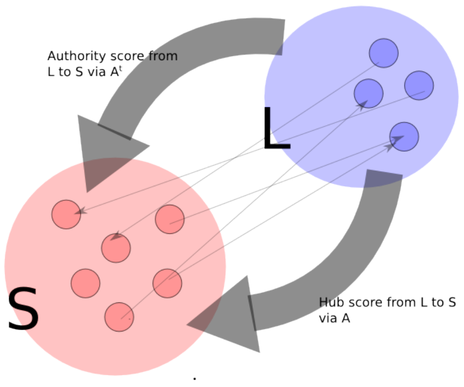


Figure 9: ranking directions for the nodes in S

The HITS algorithm is basically computed by iterating the defining equations, where a kind of rank flow emerges along the edges of the graph (by applying the hub score rule) and in opposite direction along the transposed edges of the graph (by applying the authority score rule).

In an bipartite graph the quality of *hub* and *authority* gets more specific: a good hub in S will always link to good authorities in L. And a good authority in L will receive many links from good hubs in part S. Therefore, it is now possible to give a specific direction of each hub and authority score.

To illustrate this new quality of directed ranking one could impose new names for the rankings: For example the hub score of a node i in L could be reasonably called $hubS_i$ due to the intended direction of this ranking. This leads obviously to an inter part ranking where each node can be examined for its importance towards the nodes of another part.

In previous approaches for ranking algorithms on multipartite graphs, on the other hand, it has been a reoccurring pattern to define inner part relations as *returning relations* or *hidden, inner part links* to examine the graph. We will try to unite the idea of inner part ranking and inter part ranking for cyclic multipartite graphs.

Let G be a cyclic multipartite graph with p parts as seen in definition 2 and A its adjacency matrix. General hub score towards part k in the direction of the cycle, and reciprocally a general authority score from part $p - k$ in opposite direction of the cycle is defined by:

$$h_k = \lambda A^k \cdot a_{p-k}$$
$$a_{p-k} = \mu A^{tp-k} \cdot h_k$$

For any $k \leq p, k \in \mathbb{N}$.

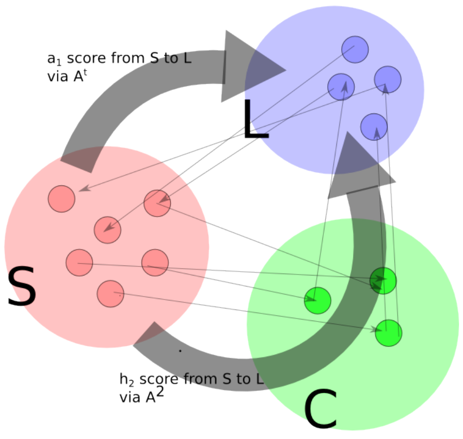


Figure 10: ranking directions for the nodes in S

So the cyclic structure of the graph can be used to define scores that are related to any other part in the cycle. For example it might be interesting if good courses are generally attended by good students. Again eigenvalue equations can be obtained by simplification:

$$h_k = \lambda\mu A^k (A^t)^{p-k} \cdot h_k$$
$$a_{p-k} = \lambda\mu (A^t)^{p-k} A^k \cdot a_{p-k}$$

Note that for bipartite graphs with $p = 2$ and $k = 1$ the HITS equation emerges. Furthermore for $p = 2$ and $k = 2$ the h_2 score resembles the definition of the $eval^2$Rank seen in [15].

For a multipartite graph with p partitions, the h_p rank is equal to the a_0 rank. Hence the two *extreme* ranking pair (a_0, h_p) can be considered as one ranking, and the same holds for the (h_p, a_0) ranking pair. The parameter k that defines which ranking pair is regarded at the moment will be referred to as *ranking parameter*. The influence of the ranking parameter on the obtained ranking is discussed in section 5.

But note that the eigenvalue equations are unsophisticatedly defined. To proof the existence of maximal eigenvalues and principal eigenvectors of the matrices M

and N in the HITS equations the symmetric property of these matrices was used. While it is not impossible that these vectors would exist for the matrices $A^k(A^t)^{p-k}$ and $(A^t)^{p-k}A^k$, the ideas from section 2.2.1 are used to ensure the existence of such vectors, and that they can be found by applying the Von Mises Iteration.

4.3 Proof of Convergence

In order to apply the mathematics discussed in 2.2.1 block wise column stochastic matrices are needed. The block wise damped matrices A_d and A^t_d of A and A^t can be computed easily. Note that A^t_d is not the transpose of A_d, for clarity one could write $(A^t)_d$.

It can be assumed that the graph does not contain any dangling nodes or that this problem is has been treated with an appropriate damping method.

Now the eigenvectors and eigenvalues of a Matrix $Q = A^k_d \cdot (A^t_d)^{p-k}$ are examined. In the following the $h_k$rank is defined to be an eigenvector of Q. A prove that the given definition is well-defined will be given. It may be assumed, that all following calculations are also applicable to a matrix of the form $(A^t_d)^{p-k} \cdot A^k_d$ to find the $a_k$rank.

Both matrices A_d and A^t_d are column stochastic. Therefore any Q is also column stochastic and also in block form.

Because Q is column stochastic it is known that 1 is a dominant eigenvalue of Q. If Q is irreducible then there is a positive real eigenvector the eigenvalue 1, but that is not necessarily true.

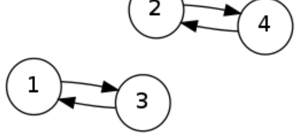
Let us consider a simple four cycle with adjacency matrix given by

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

There are four parts, each of size 1, and C is block wise damped and the h_3 vector cab be computed. This is a positive eigenvector of the matrix

$$Q = C^3 \cdot C^t = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Which is associated with the following graph that is obviously reducible:



Wee see that this graph contains two disjoint cycles. The next lemma will show that this can be generalised for non trivial part sizes. In this example Q is seen as an adjacency matrix of a new graph. This approach might be useful, as it may be hard to define "disjoint cycles" of an arbitrary matrix.

Q is a block matrix and each block is either positive or has exclusively zero entries. But Q is not necessarily the adjacency matrix of a multipartite graph because it could have entries in the main diagonal. Furthermore the dimensions of the blocks are the same as in A_d and the contraction method can be applied on Q to compute $\overline{Q_p}$. Using $\overline{Q_p}$ an important lemma can be stated.

Lemma 3. The graph associated with Q is either cyclic multipartite or it consists of several disjoint cyclic multipartite components.

Proof. It is sufficient to show that the graph associated with $\overline{Q_p}$ is either a cycle or a composition of several disjoint cycles. We examine $\overline{Q_p}$

$$\begin{aligned}\overline{Q_p} &= \overline{A_d^k \cdot (A_d^t)^{p-k}} \\ &= \overline{A_p^k (A_p^t)^{p-k}} \\ &= \overline{A_p}^k \, \overline{(A_p^t)}^{p-k}\end{aligned}$$

Note that $\overline{A_p}$ are both simple cycles $\overline{A_p^t}$ in opposite direction, and therefore $\overline{A_p}^{-1} = \overline{A_p^t}$. This can be used to simplify the last expression.

$$\begin{aligned}\ldots &= \overline{A_p}^k \, \overline{(A_p)}^{-(p-k)} \\ &= \overline{A_p}^{p-2k}\end{aligned}$$

$\overline{Q_p}$ is a power of $\overline{A_p}$ which is a cycle of length p. Therefore $\overline{Q_p}$ is a simple cycle, if $p-2k$ and p are coprime, and it is a composition of several cycles if $p-2k$ and p have a greatest common divisor. □

Corollary. It follows directly, that there is an integer number κ with $\overline{Q_p} = I_p$ where I_p is the identity matrix of size p. (For instance if κ is a common multiple of the lengths of the cycles in $\overline{Q_p}$). Therefore Q^κ is a block matrix where all blocks are on the main diagonal.

κ can be used later to assure the convergence of the Von Mises Iteration. But first we need to prove that Q has an eigenvector that satisfies the ideas of the basic approach.

Theorem 6. Q has a positive real eigenvector v with eigenvalue 1.

Proof. By the lemma it is known that Q is either multipartite cyclic or it contains several disjoint multipartite cyclic components. If Q is cyclic multipartite then it is irreducible and there is a real positive eigenvector. If Q contains l disjoint cycles then each of these cycles is an irreducible subset of the vertex set. Therefore there is a transposition matrix P so that P^tQP has the following form:

$$P^tQP = \begin{pmatrix} C_1 & 0 & \dots & 0 \\ 0 & C_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & C_l \end{pmatrix}$$

Where each of the matrices C_i is the adjacency matrix of the cycle i. P^tQP is still column stochastic and therefore every matrix C_i is column stochastic. C_i is also irreducible and therefore there is a positive eigenvector v_i of C_i with eigenvalue 1. Now let us consider the concatenation vector $v = (v_1, v_2, \dots, v_l)^t$ of all eigenvectors v_i. v is a positive real eigenvector of P^tQP as:

$$\begin{aligned} P^tQP \cdot v &= \begin{pmatrix} C_1 & 0 & \dots & 0 \\ 0 & C_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & C_l \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_l \end{pmatrix} \\ &= \begin{pmatrix} C_1 \cdot v_1 \\ C_2 \cdot v_2 \\ \vdots \\ C_l \cdot v_l \end{pmatrix} \\ &= \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_l \end{pmatrix} \end{aligned}$$

□

We assured that Q has a positive real eigenvector, and the final definition of our ranking vectors can be given by:

$$\begin{aligned} h_k &= A_d^k (A_d^t)^{p-k} \cdot h_k \\ a_{p-k} &= (A_d^t)^{p-k} A_d^k \cdot a_{p-k} \end{aligned}$$

The real factors λ and μ have become obsolete.

Now an iterative algorithm that converges to the defined eigenvector v of Q will be given.

Consider the sequence of vectors given by the Von Mises Iteration over Q. We set

$$v_i = Q^i \cdot v_0 (i \in \mathbb{N})$$

Where v_0 is an arbitrary positive starting vector. The sequence (v_i) will not necessarily converge, but there are sub series that converge to v. First we will find an converging sub sequence, and then we will show that it v can be computed also by the regular Von Mises iteration by choosing an appropriate starting vector v_0.

Lemma 4. The sub series of (v_i) given by $w_i := v_{\kappa \cdot i} = Q^{\kappa \cdot i} \cdot v_0$ converges to a positive vector.

Proof. Q^κ is in diagonal block form with square blocks $B_1, \dots B_p$. We obtain for w_i

$$\begin{aligned} w_i &= (Q^\kappa)^i \cdot v_0 \\ &= \begin{pmatrix} B_1 & 0 & \dots & 0 \\ 0 & B_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & B_p \end{pmatrix}^i \cdot v_0 \\ &= \begin{pmatrix} B_1^i & 0 & \dots & 0 \\ 0 & B_2^i & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & B_p^i \end{pmatrix} \cdot v_0 \\ &= \begin{pmatrix} B_1^i \cdot v_{0_1} \\ B_2^i \cdot v_{0_2} \\ \vdots \\ B_p^i \cdot v_{0_p} \end{pmatrix} \end{aligned}$$

Where $v_{0_1}, \dots v_{0_p}$ are appropriate subvectors of v_0. All of these blocks are positive and column stochastic. Therefore there exists an positive real eigenvector b_i to the eigenvalue 1 for each block matrix B_i. Each eigenvector b_i can again be found by an Von Mises iteration over B_i with the starting vector v_{0_i}. Hence, the series (w_i) converges to the concatenation vector $b = (b_1, \dots, b_p)^t$ which is positive. □

Note that it is still unproven whether b is an eigenvector of Q, which will be proven know. b is depended on the choice of the starting vector v_0. More precisely, it depends on the sub vectors v_{0_i}. If we choose a starting vector $(\lambda v_{0_1}, \dots v_{0_p})^t$ instead of $(v_{0_1}, \dots v_{0_p})^t$ a vector $b = (\lambda b_1, \dots, b_p)^t$ will be obtained instead of $(b_1, \dots, b_p)^t$. This motivates the use of a "blockwise normed" starting vector v_0 where each the sum over each sub vector v_{0_i} is equal to one. For instance, such a vector can be found by

$$v^* := \begin{pmatrix} \left.\begin{matrix} \frac{1}{s_1} \\ \vdots \\ \frac{1}{s_1} \end{matrix}\right\} = v_1^* \\ \left.\begin{matrix} \vdots \end{matrix}\right\} v_2^*, \dots, v_{p-1}^* \\ \left.\begin{matrix} \frac{1}{s_p} \\ \vdots \\ \frac{1}{s_p} \end{matrix}\right\} = v_p^* \end{pmatrix}$$

Note that the weight contraction of v^* is given by $\overline{v^*} = \overbrace{(1, \dots, 1)}^{p}$ and the same holds for b.

Theorem 7. The Von Mises iteration over Q will converge when v^* is chosen as starting vector.

Proof. Q is either in cyclic block form or it has several disjoint cyclic components, but is sufficient to prove the theorem for the cyclic case. For brevity a graph with three parts is considered, the proof with an arbitrary number of parts will follow the same idea. Now Q may be given by

$$Q = \begin{pmatrix} 0 & A & 0 \\ 0 & 0 & B \\ C & 0 & 0 \end{pmatrix}$$

The first vectors of the Von Mises iteration are given by:

i	0	1	2	3	4	5
v_i	$\begin{pmatrix} v_1^* \\ v_2^* \\ v_3^* \end{pmatrix}$	$\begin{pmatrix} A\ v_2^* \\ B\ v_3^* \\ C\ v_1^* \end{pmatrix}$	$\begin{pmatrix} AB\ v_3^* \\ BC\ v_1^* \\ CA\ v_2^* \end{pmatrix}$	$\begin{pmatrix} ABC\ v_1^* \\ BCA\ v_2^* \\ CAB\ v_3^* \end{pmatrix}$	$\begin{pmatrix} ABCA\ v_2^* \\ BCAB\ v_3^* \\ CABC\ v_1^* \end{pmatrix}$	$\begin{pmatrix} ABCAB\ v_3^* \\ BCABC\ v_1^* \\ CABCA\ v_2^* \end{pmatrix}$

The products of the block matrices A, B and C repeat in the same order, and i can set to be arbitrarily large. If s_1 is the length of v_1^*, the first s_1 entries of v_i can be expressed in one of the following forms.

$$v_{i_1} = \begin{cases} (ABC)^n \cdot v_1^* & \text{if } i \textit{ modulo } 3 = 0 \\ (ABC)^n \cdot A\ v_2^* & \text{if } i \textit{ modulo } 3 = 1 \\ (ABC)^n \cdot AB\ v_3^* & \text{if } i \textit{ modulo } 3 = 2 \end{cases}$$

The matrix (ABC) is positive and column stochastic, and therefore the Von Mises Iteration over (ABC) will converge for appropriate starting vectors. Note, that the three vectors r, s and t given by

$$r := v_1^*, \quad s := A\ v_2^*, \quad t := AB\ v_3^*$$

are appropriate starting vectors. Therefore the 3 series

$$((ABC)^n \cdot r), ((ABC)^n \cdot s), ((ABC)^n \cdot t)$$

will converge to multiples of the same vector ν when n grows large. But r, s and t are normed, and therefore they all converge to the same multiple of ν. This also applicable to the lower entries of v_i, and therefore (v_i) converges. □

Remark.

1. Only the fact that the weight contraction of v^* is equally distributed was used in the proof. Therefore the Von Mises Iteration over Q will converge on any starting vector with a equally distributed weight contraction that satisfies the conditions of theorem 2. But v^* will be used as a starting vector from now on.

2. As the sub series $(v_{\kappa i})$ from the last lemma is a sub series of (v_i) it converges to the same vector. If κ is known, this property might be useful to shorten the runtime of the algorithm. Hereby the $eval^2$-Rank proposed in [15] may also be found by performing a Von Mises iteration with starting vector v_0 over the bipartite matrix M_d.

Example. All a_k, h_{p-k} ranking pairs of the tripartite cyclic example 1 can be computed.

h_1, a_3	h_2, a_2	h_3, a_1
$\begin{pmatrix}0.509\\0.444\\0.313\\0.345\\0.295\\0.311\\0.246\\0.149\\0.246\end{pmatrix}, \begin{pmatrix}0.46\\0.485\\0.234\\0.356\\0.356\\0.299\\0.278\\0.113\\0.255\end{pmatrix}$	$\begin{pmatrix}0.522\\0.43\\0.311\\0.343\\0.298\\0.307\\0.242\\0.151\\0.252\end{pmatrix}, \begin{pmatrix}0.455\\0.49\\0.227\\0.356\\0.361\\0.302\\0.279\\0.116\\0.249\end{pmatrix}$	$\begin{pmatrix}0.522\\0.431\\0.312\\0.343\\0.298\\0.306\\0.241\\0.152\\0.254\end{pmatrix}, \begin{pmatrix}0.46\\0.485\\0.234\\0.356\\0.356\\0.299\\0.278\\0.113\\0.255\end{pmatrix}$

Table 7: Complete $A_n - H_n$Rank on the graph of example 1.

4.4 Implementation

In the last subsection the foundation for an iterative algorithm that is based on the Von Mises iteration was laid. First a simple implementation that takes the adjacency matrix A as basis of calculation is given. This approach is based on numerical operations on matrices. It does not require objects or a "call by reference" and therefore it can be adapted to fit the specification of GNU Octave [1]

First a block wise damping function `dampBlockWise(A, alpha)` has to be implemented. This function computes A_d from a multipartite matrix A with the damping factor `alpha`. We assume that a damping function `damp(M, alpha)` which computes $\widetilde{M}$ of an arbitrary matrix M is given. Further it can be assumed that a list of all blocks $A_{i,j}$ is obtained by a call of a function `blocks(A)`. Then a function `dampBlockWise(A, alpha)` can be easily implemented as a mapping of the `damp()` function over `blocks(A)`, to obtain the block wise damped matrix A_d and A_d^t respectively.

As the Von Mises iteration is a method of approximation, a stop criterion is needed. This criterion has to hold when a good approximation of a ranking vector is given. It is known that the Von Mises iteration will converge on our matrix, and it can be assumed, that this stop criterion is reached after $c \in \mathbb{N}$ iterations, if c is chosen to be sufficiently large. Certainly it would be interesting to know the minimal number of iterations that are needed for the approximation to satisfy a certain aberration criterion. Finding a minimal number c could be done by examining the speed of convergence of the Von Mises iteration which is known in certain cases. But for c is considered to be well chosen.

It may be assumed that the number of parts p is always given during the computation. An appropriate starting vector v_0 for this iteration as defined in the last subsection may be computed by a function `startvec()`.

With these preliminaries given the final functions `akRank(A, alpha,k, n)` and `hkRank(A,alpha,k,n)`, that implement a Von Mises

[1]GNU Octave is documented and presented at `http://www.gnu.org/software/octave/`

Iteration over the according matrices can be given. As mentioned earlier the Von Mises Iteration may be computed without an extra normalisation step.

```
% a method that computes the a_k rank and h_(p-k)
% A the multipartite Matrix
% alpha the damping Factor
% k the ranking variable
% n the number of iterations
function ak = akRank(A, alpha, k, delta)

      %damp the matrices Ad and Adt block wise
   Ad = dampBlockWise(A, alpha);
   Adt = dampBlockWise(transpose(A), alpha);

   %Define Q
   Q = Adt^k * Ad^(p-k);

   %define the first approximation of ak
   ak = startvec();

   %the Von Mises Iteration
   for i = 1 : c
      ak = Q * ak;
   endfor;
end
```

Implementation 1: A function that computes the $a_k$rank

```
% a method that computes the h_k rank
% A the multipartite Matrix
% alpha the damping Factor
% k the ranking variable
% n the number of iterations
function hk = hkRank(A, alpha, k, delta)

      %damp the matrices Ad and Adt block wise

      Ad = dampBlockWise(A, alpha);
   Adt = dampBlockWise(transpose(A), alpha);

   %Define Q
   Q1 = Ad^k * Atd^(p-k);


   %define the first approximation of hk
   hk = startvec();

   %the Von Mises iteration
   for i = 1 : c
      hk = Q * hk;
   endfor;
end
```

Implementation 2: A function that computes the $h_k$rank

Note that the iteration over `Q` is not very efficient. The two functions can be combined to compute the rankings a_k and h_{p-k} simultaneously. By this means a ranking pair as it was defined in the basic approach is obtained. This function is called `AnHnRank` as all upcoming variations of the algorithm will be based on this implementation.

```
 1 % A function that computes the ranking pair (a_k, h_(p-k))
 2 % A the multipartite matrix
 3 % alpha the damping factor
 4 % k the ranking variable
 5 % c the number of iterations
 6 function [hub, auth] = AnHnRank(A, alpha, c, delta)
 7
 8        %damp the matrices Ad and Adt block wise
 9    Ad = dampBlockWise(A, alpha);
10    Adt = dampBlockWise(transpose(A), alpha);
11
12    %Define Q
13    Q = Ad^k;
14
15    %Define P
16    P = Adt^(p-k);
17
18    %define the initial values for hub and auth
19    hub = initvec();
20    auth = initvec();
21
22    %iteration
23    for i = 1 : c
24       hub = Ad^k * auth
25       auth = Adt^(p-k) * hub
26    endfor;
27 end;
```

Implementation 3: A function that computes the (a_k, h_{p-k}) ranking pair.

Using this last algorithm may be more efficient than applying the first two algorithms consecutively to obtain a ranking pair, but it can still be optimised. Note that matrix multiplications can be parallelized, and this could be applied to the lines 24 and 25 of the algorithm.

Furthermore the matrices `Ad` and `Adt` could be very large, depending on the use case. Storing every entry of these matrices is not very efficient. Only the blocks of `Ad` and `Adt` could be stored. This would reduce the matrix multiplications of line 24 and 25 to operations on these blocks. But since the blocks of A may be sparse it will be even more efficient to store the data in an adjacency list or a similar data structure without producing data in the damping process.

Fortunately it is possible to solve the problems of runtime and storage efficiency with the Google pregel framework. The main idea of this framework is to implement the nodes of a graph as objects, that can store information send messages along the edges of the graph. See [14] for further information about Google pregel.

Unfortunately this framework is not made available by Google, but there are several available implementations that satisfy the main characteristics of Google pregel [2]. We give a pregel based implementation in pseudo code that can be adapted

[2]Apache Giraph: `http://incubator.apache.org/giraph/`, Phoebus: `https://github.com/xslogic/phoebus`, GoldenOrb `http://goldenorbos.org/`

to fit the GoldenOrb specification.

Each vertex sends its initial authority value along the edges of the graph, where the next vertex processes it with the other authority values he obtained. This forwarding of authority values is done exactly k times. The processed authority values which each vertex computes in superstep k is the new hub value of this vertex. By this means have reached line 24 of the numerical AnHnRank function. Then the `hub` values are computed in opposite direction of the edges of the graph. This happens $(p - k)$ times, and line 25 of the code of the AnHnRank function is reached. We repeat the hub and authority forwarding c times, and the final values are stored in the vertexes.

A `AnHnVertex` class is implemented as a subclass of `Vertex`.

```
AnHnVertex extends Vertex<Double[], Double, AnHnMessage>{
...
}
```

Each node will need to store its current `hub` and `auth` value. These values may be stored in an array of doubles which is the vertex type. The edge type is double, because every edge will need to provide its edge weight.

The message type `AnHnMessage` contains more information. Each message will have to carry either a fraction of the new `hub` or `auth` value. Therefore each message must carry a double value which is stored in the variable `AnHnMessage.messageValue`. It is important that each message is either a hub *or* an authority message, and that this difference is indicated by the message, so it has to carry an identifier which is stored in an `enum` value `message.messageType`. A third variable is needed which implies whether the message has reached its destination, or it has to be forwarded along the graph. The final class definition of the message type is given by:

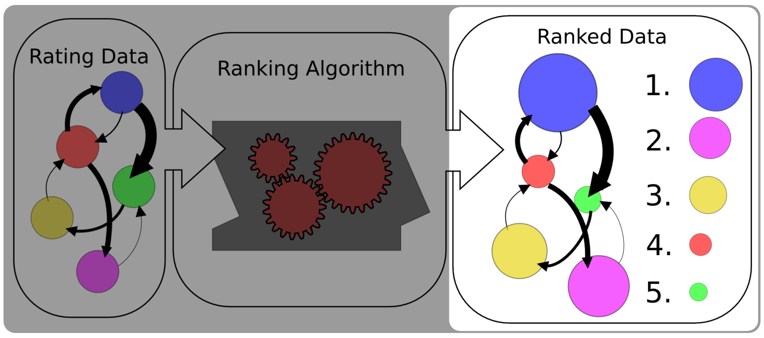
```
AnHnMessage extends Message{
   double messageValue;
   int remainingDistance;
   Type messageType;
}

enum Type {
   HUB, AUTHORITY
}
```

For brevity it may be assumed that the outgoing edges of a vertex are stored in array of edges `AnHnVertex.out`. Each vertex will also have to know its incoming edges to send messages in opposite direction along the graph. Let these edges be stored in `AnHnVertex.in`. To avoid complication we may say that the destination of an incoming edge is its root vertex. There are other global values that each vertex has to now, as the size of the part it is in, or the damping value. These values may be stored in the distributed framework, and we will just assume that each vertex has a reference to them.

Now all the information we need to give an implementation of the AnHnVertex class is gathered.

```
AnHnVertex extends Vertex<Double[], Double, AnHnMessage>{

  //the outgoing edges
  Edge[] out;

  //the incoming edges
  Edge[] in;

  //number of parts in the graph
  int p;

  //the size of part this vertex is in
  int partsize;

  //the sum of all weights of the outgoing edges
  double inWeight;

  //the sum of all weights of the incoming edges
  double outWeight;

  //defined ranking parameter
  int k;

  //damping value
  double alpha;

  //the computed values.
  double auth;
  double hub;

  //number of iterations before abort
  int c

  public void compute(Collection<AnHnMessage> messages) {

    if(superStep() == 1){

      //The inital auth value
      auth = 1 / partsize;

      //The first auth message with destination distance k
      for(Edge e : out){
        sendMessage(e.destination, Type.AUTHORITY, k, auth);
      }

    } else { //superstep > 1

      //We need to know in which state the system is in
      int remDis = messages[0].remainingDistance;
      Type messageType = messages[0].messageType;

      //Sum up the incoming values
      double sum = 0;
      for(AnHnMessage m : messages){
        sum += m.messageValue;
      }


      if (remDis == 0) {
      //The messages have reached its destination

        if(messageType == Type.AUTHORITY){
        // We have received authority messages
          //store the damped hub value;
          hub =  (1 - alpha)/partsize + alpha * sum / inWeight;

          for(Edge e : in){
          //Send hub messages with the new auth value
          //In opposite direction
            sendMessage(e.destination, Type.HUB, p-k, hub);
          }

        } else {
        //We have received a hub message
          //Store the damped auth value;
          auth = (1 - alpha)/partsize + alpha * sum / outWeigth;

          if (superStep() <= c * p){
            //start another cycle ...
            for(Edge e : out){
              sendMessage(e.destination, Type.AUTHORITY, auth);
            }
          }
        }

      } else {
      //The message is still on its way

        remaining distance --;

        if (messageType == Type.AUTHORITY){

          for (Edge e : out){
            //Forward auth message with damped value val
            double val  = (1 - alpha)/partsize
                          + alpha * sum / outWeigth;
            sendMessage(e.destination, Type.AUTHORITY, remDis, val);
          }

        } else {

          for (Edge e : in){
            //Forward hub message with damped value val
            double val  = (1 - alpha)/partsize
                          + alpha * sum / inWeigth;
            sendMessage(e.destination, Type.HUB, remDis, sum);
          }
        }
      }
    }
    voteToHalt();
} }
```

5 Evaluation



In this section the ranking is evaluated according to the problems we mentioned at the introduction. For instance, it would be desirable, that a rating caused by an *injected disturbance* would have a lower influence on the *An-HnRank* of a vertex than it would have on other rankings.

In order to evaluate our new algorithm rating data has to be gathered first. One possible way of doing this is described in the next sub section. Afterwards different rankings will be compared. Then the new ranking is applied to a graph with *injected disturbances*.

5.1 Gathering Rating Data

To get a reliable result of our evaluation, an authentic multipartite rating graph is needed. It would make no sense to use rating data that is unlikely to occur in real life. Authentic data can be obtained by gathering data in a real life rating system.

But in this rating graph the occurring problems as *"injected disturbance"* or *"lack of information"* need to be identified in advance. So the gathered data would have to be examined *by hand* to provide this identification. This may be hard to do considering large scale systems with many ratings and vertexes. Furthermore we could never know if the examination by hand would provide an valid identification.

One way of attacking this problem is described in [18] where new edges of an somehow *untypical* weight are added to the rating graph. By this means we bias the rating given by certain vertexes. Therefore, the *injected disturbance* problem can be simulated. But this approach has its flaws. The rating graph may already exhibit such disturbing vertexes, without further manipulation. If these non manipulated vertices are identified by the algorithm, an examination by hand would be needed to assure the validity of the results. If these vertices are not identified, their existence is unknown unless the graph is again examined by hand. Hence we would not know if the computed ranking exhibits the desired property.

We propose to simulate a rating graph. If this approach is followed, different simulation patterns can be used for different vertices, and we get an exact mapping of a vertex to its rating behaviour. But which simulation patterns are authentic?

We model a *single rating* as a single value of a random variable. If an authentic simulation pattern has to be found, an authentic random variable is needed. Afterwards we can produce synthetic data with merely any number of vertices and ratings.

We examine the data gathered at the eval studies at the Ludwig Maximilian Universität Munich. Here ratings from lecturers and students are gathered. This bipartite rating system will be used to compute two random variables $\mathcal{L}$ and $\mathcal{S}$ which may then be considered as authentic.

Let us start with a random variable $\mathcal{L}$ which will model the rating behaviour of the lecturers. The ratings given by the lecturers are rounded to obtain integer numbers from 1 to 10. We can give the relative frequency of these rounded ratings.

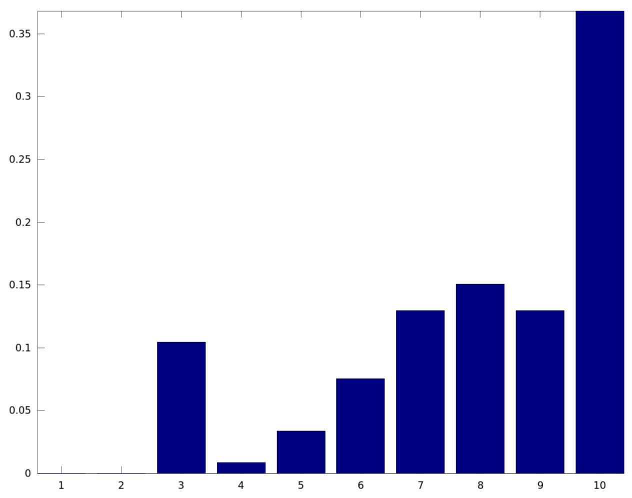


Figure 11: relative frequencies of the lecturer ratings

We use these relative frequencies as values of the discrete probability density function $f_{\mathcal{L}}$. Standard measures from statistics may be used to obtain the random variable $\mathcal{L}$ with $f_{\mathcal{L}}$ as probability density function.

The random variable $\mathcal{S}$ which is used to model the student rating can be obtained in the same way. Here our measurements exhibited a different probability density function:

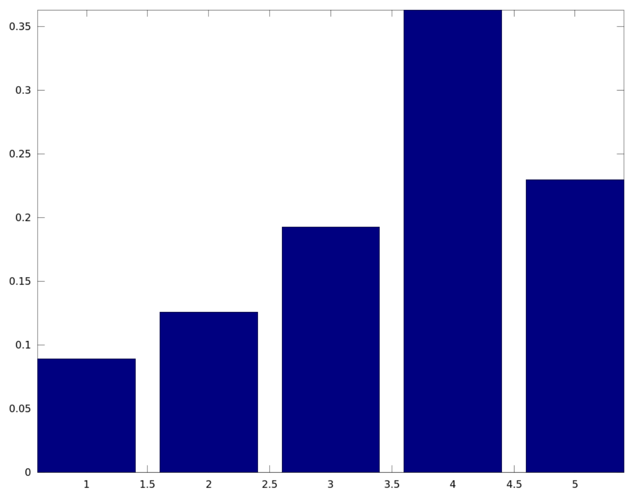


Figure 12: relative frequencies of the student ratings

Note that here the rankings given vary between values of one and five. This can be considered as an occurrence of the *typed rating* problem because vertices from different parts use different scales for their numeric rating. As mentioned earlier, this problem is solved by the damping procedure included in the $A_n - H_n$ Ranking.

By this means, a rating graph with three parts can be simulated. In our simulation the *course*-part will simply gather the ratings from the *student*-part, and pass on the accumulated values on to the *lecturer*-part. By creating this simulated rating graph, we have to bear in mind that a student would only rate courses he attended. All evaluations given in the next subsection are based on such a simulated graph. It is denoted by G_{sim}.

5.2 Comparison With Other Rankings

We will compare the $A_n - H_n$Ranking to other rankings, that may be applicable to G_{sim}. Of course the *mean input* ranking mentioned in the introduction will be used as a matter of comparison.

Further we will use two other rankings that are not necessarily designed for multipartite graphs. In [15] the *weighted PageRank* PR_w is described, that is applicable to any graph with weighted edges. Further we will naively apply the HITS algorithm to our graph. We know from section 2.2.2 that it will also converge graphs with weighted edges.

Correlation coefficients are used for the comparison. By this means linear dependencies between the rankings can be discovered.

Correlation	h_1	h_2	h_3	a_1	a_2	a_3
PR_w	.8	.73	.73	.9	.99	.99
authority score	.81	.79	.76	.79	.7	.7
hub score	-.83	-.77	-.78	-.79	-.72	-.7
mean incoming value	-.7	-.65	-.66	-.61	-.49	-.48

Table 8: Correlation coefficients of the new ranking with other rankings.

As known, the $A_n - H_n$-Rank gives partition wise normed ranking vectors. Therefore, the ranking of vertices from different parts are examined separately.

Rankings in the *lecturer* part:

correlation	h_1	h_2	h_3	a_1	a_2	a_3
PR_w	0.8	0.73	0.73	0.9	1	1
authority score	0.81	0.79	0.76	0.79	0.7	0.7
hub score	-0.83	-0.77	-0.78	-0.79	-0.72	-0.71
mean incoming value	0.29	0.1	0.1	0.77	0.96	0.96

Table 9: Correlation coefficients of the new ranking with other rankings in the *lecturer*part.

Rankings in the *student* part:

correlation	h_1	h_2	h_3	a_1	a_2	a_3
PR_w	0.8	0.73	0.73	0.9	1	1
authority score	0.81	0.79	0.76	0.79	0.7	0.7
hub score	-0.83	-0.77	-0.78	-0.79	-0.72	-0.71
mean incoming value	0.03	-0.02	-0.02	1	0.88	0.85

Table 10: Correlation coefficients of the new ranking with other rankings in the *student* part.

Rankings in the *course* part:

correlation	h_1	h_2	h_3	a_1	a_2	a_3
PR_w	0.8	0.73	0.73	0.9	1	1
authority score	0.81	0.79	0.76	0.79	0.7	0.7
hub score	-0.83	-0.77	-0.78	-0.79	-0.72	-0.71
mean incoming value	0.6	0.17	0.17	0.8	0.95	0.95

It is striking, that the correlation of the partwise rankings are equal to the global rankings except the *mean input* ranking. This might be due to the block structure of the adjacency matrix.

All correlations are relatively high except for the *mean input* ranking. The high correlation between a_3 and PR_w could be expected, due to their definition.

5.3 Influence of Injected Disturbances

The algorithm is know evaluated according to the *injected disturbance* problem, which was mentioned at the introduction. Such a disturbance can be simulated by varying the outgoing edges of certain nodes in one part. Nodes from the *lecturer*-part are used for the manipulation, the results obtained by manipulating nodes from the other parts are similar to the ones given now. With set up data, the impact of this variation on the rankings can be measured.

For this experiment there are two main parameters that can be varied. The first parameter is the percentage p of manipulated vertices in the *lecturer* part. Values between one and twenty percent are chosen for p.

The other parameter that may be varied is the strength of the manipulation, meaning the difference that a manipulated edge weight has to a "normal" edge weight. Consider the mean value of the lecturer rating $m = 7.92$, and the standard deviation $s = 2.26$. Let w be the average weight of the manipulated edges. We don't consider values of w within the interval $[m - s, m + s]$ as real disturbances, and chose values w_1, w_2 and w_3 for the new edge weights that differ from m of a multiple of s.

$$w_i := m - s \cdot i$$

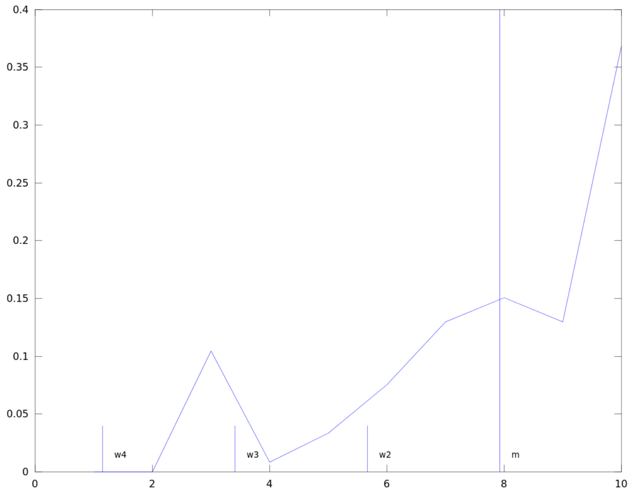


Figure 13: Proposed values of the manipulated edges.

With these two parameters in hand a set of manipulated rating graphs according to w and p can be computed. These graphs are denoted by $G_{sim}(w, p)$. With variations of w and p, the rankings of the vertices in $G_{sim}(w, p)$ vary. From the vertices in the *student*-part one is chosen, that has an incoming edge from a manipulated vertex. The first ranking we want to observe is the *mean input* ranking. We plot the relative variation of the mean input value on the x_3 axis and the variations of p and w on x_1 and x_2. It is not surprising, that there is a nearly linear dependency between the *mean input* ranking and the manipulations.

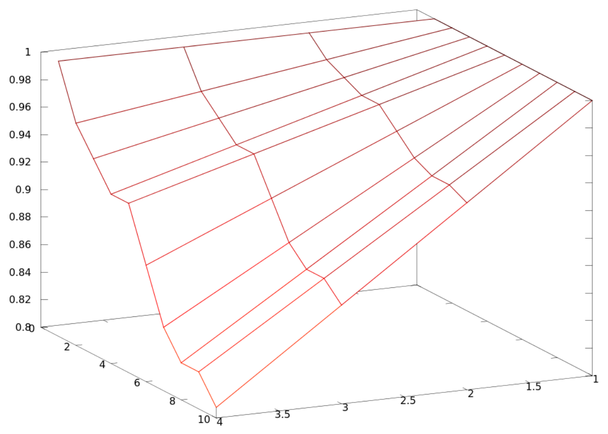


Figure 14: variation of the *mean input* ranking

Next we consider the variations of the a_1 ranking, that is known to correlate strongly with the *mean input* ranking. The correlation still holds under our manipulation, but that the a_1 ranking damps the manipulation effect.

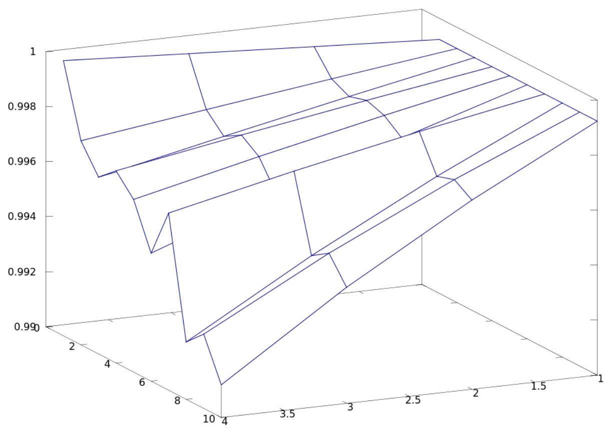


Figure 15: variation of the a_1 ranking

We plot the other ranking of the a_k, h_{p-k} ranking pair h_2, and observe that this ranking is barely influenced by the Ranking.

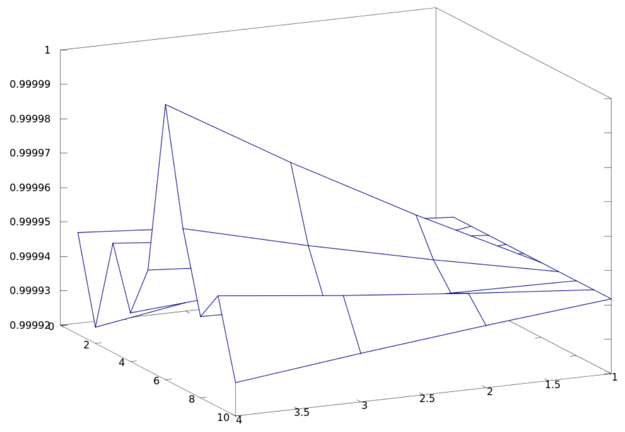


Figure 16: variation of the h_2 ranking

Note that this is the character of the rankings of one *single* vertex that we accepted as *typical*. Further investigation, especially about the correlation of the different rankings, is necessary to get more reliable results.

6 Conclusion and Future Work

The goal of this thesis was to examine how ranking approaches related to eigenvector centrality can be applied to multipartite graphs, and which approaches have already been applied to those graphs. We further wanted to develop a ranking algorithm that is adaptable to different multipartite graphs with weighted and directed edges.

In this thesis a mathematical structure was developed to compare different multipartite graphs. We saw that the *partition graph* is a fast and easy tool to make such a comparison. With the *partition graph*, we pointed out for which kinds of multipartite graphs ranking algorithms are already developed.

Possible eigenvector centrality measures on multipartite graphs were examined. A matrix manipulation, the *blockwise damping*, that was already used on bipartite matrices [15], was adapted to multipartite matrices. Further it was shown that with such manipulations the *primitiveness* and the *irreducibility* of a graph is equivalent to the *primitiveness* and *irreducibility* of its *partition graph.*

As main result of this thesis we showed in theorem 3, that we can use the *partition graph* to define a necessary condition for the existence of an positive eigenvector of a *blockwise dampled* multipartite matrix. As a corollary of this main theorem we found, that an eigenvector ranking on a multipartite graph can be consistent with an eigenvector ranking on its partition graph.

A new ranking that is applicable to any graph that is *cyclic multipartite* was defined. We proved that it is well defined, and that it can be computed with an iterative algorithm by using the mathematical propositions that we developed earlier. It was shown that the new ranking algorithm is a generalisation of the algorithm discussed in [15] if applied on bipartite graphs. A simple and a parallelized implementation of this algorithm was given.

A rating simulation with different rating behaviours was used to build a tripartite graph with weighted edges. We then showed that the new ranking has expedient properties with regard to problems that might occur in such a rating system.

Presumption on the eigenvalues of M_d and $\overline{M_p}$ In section 2.2 it was proven, that if v is an eigenvector of M_d and if its weight contraction of $\overline{v}$ was not zero then $\overline{v}$ was an eigenvector of $\overline{M_p}$. This means that there could be eigenvectors of $\overline{M_p}$ that are no weight contraction of an eigenvector of M_d. But for all multipartite matrices that were studied during the work on this thesis, this situation did not occur. All examined eigenvectors of matrices $\overline{M_d}$ were indeed weight contractions of eigenvectors of the according matrix M_d. A (unfortunately still unproven) presumption of this observation would state:

$$\forall w \in Eig_\lambda(\overline{M_p}) \;\; \exists v \in Eig_\lambda(M_d) \quad : w = \overline{v}.$$

Where $Eig_\lambda(A)$ denotes the set of eigenvectors of a matrix A with eigenvalue λ. If this presumption holds, then eigenvector based ranking could be performed on multipartite graphs with a reducible partition graph. As stated in section 2.2.1 all graphs with partition graphs No.23 or No.30 could be ranked with eigenvector centrality.

Application to other graphs. The $A_n - H_n$-Rank was motivated on a *cyclic multipartite* graph. However the new ranking could be applied to other types of multipartite graphs. If these rankings remain appropriate has to be examined. But all types of regular tripartite graphs where an $A_n - H_n$-Rank can be applied can be given easily. Often it is not possible to get complete ranking pairs (a_k, h_{p-k}). The following rankings can be computed on tripartite graphs:

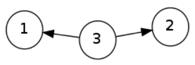
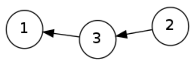
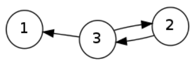
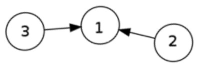
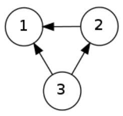
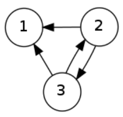
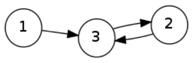
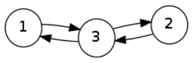
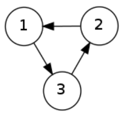
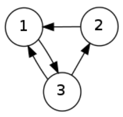
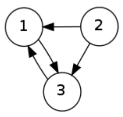
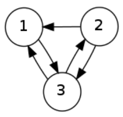
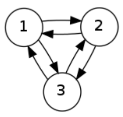
	h1	h2	h3	a1	a2	a3
No.3 	0	0	0	0	0	0
No.6 	0	0	0	0	0	0
No.7 	0	0	0	1	1	?
No.10 	0	0	0	0	0	0
No.11 	0	0	0	0	0	0
No.15 	0	0	0	1	1	?
No.21 	1	1	?	0	0	0
No.23 	1	1	1	1	1	1
No.25 	1	1	0	1	1	0
No.27 	1	1	1	1	1	1
No.30 	1	1	?	0	0	0
No.31 	1	1	1	1	1	1
No.63 	1	1	1	1	1	1

Table 11: Partition graphs and the according computable ranks. "1": sure computability, "?": computable if the presumption holds, "0": not computable

7 Bibliography

References

[1] Phillip Bonacich. Power and Centrality: A Family of Measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107 – 117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[3] Françcois Bry, Norbert Eisinger, Christoph Wieser, and Harald Zauner. Mathematics of PageRank, October 2011. URL: `http://www.pms.ifi.lmu.de/lehre/webinfosys/12ws13/unterlagen-public/math/math-of-pagerank.pdf`.

[4] François Bry. Web-Informationssysteme. Lecture in the winter term 2012/2013 at Ludwig-Maximilian-Universität München. URL: `http://www.pms.ifi.lmu.de/lehre/webinfosys/12ws13/`.

[5] Kurt Bryan and Tanya Leise. The $25,000,000,000 Eigenvector: The Linear Algebra behind Google. *SIAM Review*, 48(3):569–581, 2006.

[6] Jiyang Chen, Osmar R Zaıane, Randy Goebel, and Philip S Yu. TupleRank: Ranking Relational Databases using Random Walks on Extended K-partite Graphs. 2010.

[7] Haibin Cheng, Pang-Ning Tan, J. Sticklen, and W.F. Punch. Recommendation via query centered random walk on k-partite graph. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on.*

[8] Hongbo Deng, Michael R. Lyu, and Irwin King. A generalized Co-HITS algorithm and its application to bipartite graphs. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 239–248, New York, NY, USA, 2009. ACM.

[9] Zoltan Gyongyi, Hector Garcia-Molina, and Jan Pedersen. Combating Web Spam with TrustRank. Technical Report 2004-17, Stanford InfoLab, March 2004.

[10] Ararat Harutyunyan. *Brooks-Type Results For Coloring of Digraphs.* PhD thesis, Simon Fraser University, 2011.

[11] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. FolkRank: A ranking algorithm for folksonomies. In *University of Hildesheim, Institure of Computer Science*, pages 111–114, 2006.

[12] Glen Jeh and Jennifer Widom. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM.

[13] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[14] Malewicz, Grzegorz and Austern, Matthew H and Bik, Aart JC and Dehnert, James C and Horn, Ilan and Leiser, Naty and Czajkowski, Grzegorz. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

[15] Julien Oster. A Social Platform for Bipartite Student-Lecturer Ranking. Diplomarbeit/diploma thesis, Institute of Computer Science, LMU, Munich, 2011.

[16] Leo Spizzirri. Justification and Application of Eigenvector Centrality.

[17] Stefan Felsner. Graphentheorie, October 2008. URL: `http://page.math.tu-berlin.de/~felsner/Lehre/GrTh05/Graphentheorie.pdf`.

[18] Jimeng Sun, Huiming Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Data Mining, Fifth IEEE International Conference on.*

[19] Thomas Kamps. Grundlagen der Graphentheorie, October 2008. URL: `http://www.mathematik.uni-regensburg.de/loeh/teaching/discgeosem_ws0809/kampsGraphentheorie.pdf`.

Eidesstattliche Erklärung

Erklärung zur Hausarbeit gemäß § 29 (Abs. 6) LPO I

Hiermit erkläre ich, dass die vorliegende Hausarbeit von mir selbstständig verfasst wurde und dass keine anderen als die angegebenen Hilfsmittel benutzt wurden. Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen sind, sind in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht.

Diese Erklärung erstreckt sich auch auf etwa in der Arbeit enthaltene Graphiken, Zeichnungen, Kartenskizzen und bildliche Darstellungen.

Ort, Datum: ______________________	Unterschrift: __________________
	(Niels Becker)

Das Kultusministerium teilt mit, dass es sich bei dieser Arbeit um eine schriftliche Arbeit im Rahmen des Studiums für das Lehramt an Gymnasien handelt. Eine Bewertung der Arbeit ist nicht veröffentlicht. (3.6.2013)