Image Ranking using Group Ranking methods

Johan Sejr Brinch Nielsen

July 19, 2008

Contents

Preface

This bachelor thesis was written at the Computer Science Department of the University of Copenhagen Denmark, during the spring of 2008. The subject for this paper was inspired by a guest lecture of Dorit Hochbaum which was held during the course "Introduction to Optimization" in 2006, taught by professor David Pisinger.

I would like to thank my supervisor David Pisinger for taking this project under his supervision and for his inspiration throughout the process. Also a great thanks to Dorit Hochbaum for taking her time to give the guest lecture on the Close Rankings method which was the inspiration for this project.

I would also like to thank my fellow student Jacob Bølling Hansen for revising the paper from an academic point of view, and likewise my mother Lisbeth Brinch for revising the paper from a linguistic point of view.
I would like to thank Anja Westh-Liljenbøl for lending me her book on Scientific Computing, which contained a fine description of both the Conjugate Gradient method and the Golden Section Search.

My graditute to DIKU for putting the commercial solver CPLEX at my disposal and to its IT-department for quickly correcting the CPLEX problems that occurred doing system updates.

Finally, I want to thank every student and professor at DIKU who participated in the survey which was used in the testing section.

1 Introduction

In the past couple of years websites have been increasingly focusing on user feedback. This has led to a growth in the number of websites asking for user opinions, including sites that allow users to express these opinions through voting.
As a consequence, the number of websites offering users the ability to share and vote on content have experienced massive growth. Everything is now shared and rated on the Internet, including articles, blog posts, videos and images.
In this paper I will focus on the websites offering image sharing and ranking. This specific category within sharing has grown massively too [1]. Even though some of these image sharing services are quite advanced in other areas, using popular technologies such as AJAX[2], most of them are still using a simple method for ranking images. The consequence

[1]Today a search for "“Photo Sharing”" returns 99.5 million hits on www.google.com and 31.8 million hits on www.live.com
[2]Asynchronic Javascript And XML

is an unfaithful ranking that is easily manipulated and might not even support the users' beliefs.

In this project, I will take a look at the most popular method used on websites today and compare it to modern Group Ranking methods. In this process I will enlighten the flaws of the simple method and show that more advanced methods can be used to provide a more faithful ranking and at the same time show that it is possible to base the ranking on more than just the weights provided by the users. I will show the possibility of ranking by more variables, such as user rankings and the relevance of each vote.

Even though the motivation for this project urged from image ranking, the methods used throughout the project is general Group Ranking methods. In fact, substituting the word "image" with "music" might yield a fine paper on ranking music. As a consequence, the results in this paper apply to general Group Ranking and the methods discussed can be used as such.

The source code produced and used throughout this project can be found at:
http://opix.dk/media/bachproj.tar.gz

1.1 Relevant research

Saaty (1997) proposed the Analytic Hierarchy Process (AHP), which was to become the most used method for multicriteria decision-making. The core of this method was to find an eigenvector that provided the vector of weights.

Keener (1993) discussed the pros and cons of intensity rankings and preference rankings, and the importance of the Perron-Frobenius theorem concerning the criteria needed in order to guarentee a positive solution to the eigenvector problem $Ar = \lambda r$.

Page et al. (1999) published their famous Google PageRank in an attempt to change the way websites were ranked. The motivation of the paper was to describe a method that could rank websites objectively by measuring a surfer's interest in them. The idea of the PageRank method is to structure the websites as a matrix, representing a flow graph, and then compute an eigenvector that provides the vector of weights.

Saaty and Vargas (1984) presented a model for solving the group ranking problem in a way that minimizes the deviation between the solution and the preferred solution of each user. They used the least-squares method to approximate the solution to the proposed objective function.

Ali et al. (1986) presented a model based on Operations Research by minimizing the deviation between the solution and the preferred solution among users. The problem is defined using integer variables and was solved using a linear programming routine.

Chandran et al. (2005) proposed a linear programming approach to the model of Saaty and Vargas. In the new formulation the objective function had been modified in a way that allowed for optimizations.

Levin and Hochbaum (2006) described a new model for solving weight-only and intensity-only group ranking problems based on Operations Research. The model is a generalization of previously proposed models, (Ali et al. 1986) and (Chandran et al., 2005), and allows a more flexible ranking in that it introduces belief factors. The paper introduces an efficient method to solve the problem to optimality.

1.2 Overview

In section 2, I give a short informal description of the Image Ranking Problem followed by the needed notation and terminology and then the formal description in section 2.3. I describe the problems of Rank Reversal, Normalization and introduce the problem of Newcomer's Rush; I give a generic workaround for both normalization and Newcomer's Rush and describe why it is impossible to derive such a method with respect to Rank Reversal.

In section 3, I describe the Average-Point method, the problems arising from using it and finally how some of these problems can be relaxed by using simple modifications.

In section 4, I show how the PageRank method can be used as a group ranking method and later how it can solve the Image Ranking Problem. I describe the computations needed to use the method and give a short but strict proof of why the computations terminate and why the normalization described in (Page, 1999) can be skipped.

In section 5, I describe the Close Rankings method and show how this method can be used to solve the Image Ranking Problem. I describe a problem in the way this method compares rankings and propose a simple modification that relaxes this problem. I then go through the steps needed to transform the Close Rankings model into an unconstrained convex minimization problem and show how this minimization problem can be solved using the Conjugate Gradient method.

1.3 Contributions

I give a detailed description of how the PageRank method and the Close Rankings method can be used to rank images in a much more dynamic way than the currently used Average-Point method. I show how the Close Rankings method can be used to rank images using both user rankings and vote relevance.

I introduce the Newcomer's Rush scenario and show how all three methods are affected by this problem. Furthermore I give a workaround that relaxes the effects of Newcomer's Rush.

When researching the PageRank method, I discovered a simple proof that the recursive computations used to solve the method converge, something that was not in (Page, 1999). I give a short description of the proof and show that the normalization described in the pseudocode in (Page, 1999) is unnecessary when dealing with the Random-Surfer model.

I show that the Close Rankings method has a problem with the way it compares votes, which implies that it might go in the complete opposite direction of what was intented without being punished. I propose a simple change in the objective function of the optimization problem which fixes this problem.

I show how the problem produced by the Close Rankings method can be solved using the Subgradient method combined with Golden Section Search.

2 The Image Ranking Problem

The Image Ranking Problem consists of ranking a number of images in a way that respects the opinions of the users. The users can express these opinions by placing a vote on a subset of the images, and hence express a preferred order of these images. The goal is now to order the total set of images in a way that corresponds to the preferred order of each user. Since it will be impossible to please every single user in any non-trivial instance, the method that defines this ordering has to figure out a way to please the users as much as possible.

2.1 Notation

In this section I will introduce the notation used for accessing and working with set elements.

$S^{i,j,\dots,k}$ are the elements of the set S, that is associated with the objects $i, j, \dots, k$. E.g. $V^{u,i}, u \in U, i \in I$ are the votes $v \in V$ that is associated with user u and image i.

s_i is the ith element of $S = \{s_1, s_2, \dots, s_n\}$.

$\min(S)$ is an element $m \in S$ such that $\forall s \in S : m \leq s$.

$\max(S)$ is an element $m \in S$ such that $\forall s \in S : m \geq s$.

$\sum(S)$ is the sum of all elements in the set S, $\sum_{s \in S} s$.

$\mid S \mid$ is the number of elements in the set S.

$\mid\mid V \mid\mid_1$ is the norm of the vector V, $\sum_{i=1}^{n} \mid V_i \mid$.

$avg(S)$ is shorthand for the average $\frac{\sum(S)}{|S|}$.

2.2 Terminology

In this section I will define some terms that are used throughout the paper.

a "ranking relation" $\preceq$ is an order relation on S. An object s_i has a rank greater than or equal to s_j iff $s_j \preceq s_i$. An object s_i is said to have a strictly greater rank than s_j iff $s_j \preceq s_i \wedge s_i \npreceq s_j$. The relation satisfies completeness, such that if $s_i, s_j \in S$ then $s_i \preceq s_j \vee s_j \preceq s_i$. Furthermore the relation is transitive, hence if $s_i \preceq s_j \wedge s_j \preceq s_k \Rightarrow s_i \preceq s_k$. I will say that two objects share rank, $s_i =_{\preceq} s_j$, iff $s_i \preceq s_j \wedge s_j \preceq s_i$. I will use the shorthand notation $(x_0, x_1, \dots, x_n)_{\preceq}$ instead of $x_0 \preceq x_1 \preceq \dots \preceq x_n$.

to "rank" a set S is to define a ranking relation, $\preceq$, on S.

a "score relation" on a set S is a relation $Sc : S \rightarrow \mathbb{R}$. A ranking relation can be defined from a score relation by $s_i \preceq s_j \Leftrightarrow Sc(s_i) \leq Sc(s_j)$.

a "user" is a reviewer from classic group ranking theory. A ranking relation can be based upon the users' opinions. I will let the set of all users be determined as U.

to "vote" is a way for a user to express how high a rank she believes a particular object should have. Each user can only vote once on each object. A singe vote is a triple (u_i, s_j, w) where u_i is the voting user, s_j is the object being rated and w is the weight. The set of all votes will be determined as V. The phrase "to vote α on ..." is short for "to place a vote with weight α on ..." and hence the two phrases can be used interchangeably.

a "weight" is a real number, in this paper in the range $[0..1]$ [3] that is given by a user on an image trough a vote. The higher the weight is, the higher the user thinks that the particular image should be ranked. The set of all weights will be determined as W and the weight of a particular vote vr will be determined as $w(v)$.

2.3 Formalization of the Image Ranking Problem

The Image Ranking Problem (IRP) is the problem of producing a ranking relation $\preceq$ on a set of images I, given the following information:

1. The set of images, I
2. The set of users, U
3. A score relation from users into their scores: $ScU : U \rightarrow [0..1]$
4. A set of votes, V: $(U_i, I_a, w \in W) \in V$
5. A score relation from votes into their relevance $ScV : V \rightarrow [0..1]$

I will call any method that produces a ranking relation $\preceq$ from the above information an "IRP-method". Such a method is very easy to derive, e.g. a method returning a random ranking relation. However this method does not seem to actually solve the problem in any useful way. The expectations to a solution is that it should respect the opinions of the users. In particular, the ranking relation should rank the images in such a way that it satisfies the users as much as possible.

[3]An alternative range $[1..10]$ is used on a variety of websites, e.g. www.ratemypicture.com.

2.4 A simple example

In this section I will try to illustrate what an IRP method will have to take into account. I have constructed an example of a simple IRP instance and discussed how this instance can be ranked.

Example 2.4-1:

U/I	0	1	2	3
0	-	0.5	0.3	0.4
1	0.6	-	-	0.8
2	0.5	0.8	-	0.6
3	0.6	0.7	0.7	-

An example of a simple IRP instance

The instance has 4 users and 4 images and a corresponding solution to the instance could be the ranking relation $(3, 2, 1, 0)_{\preceq}$. This ranking relation places I_0 highest, although every user who has voted on I_0 has voted lower on this image than on the alternatives. A more fair ranking relation could be one that places I_0 lowest and I_1 highest, since everyone who has voted on I_1 has placed their highest vote on this image. This fair ranking relation could now place I_3 second highest, since U_0 weigted this image heigher than I_2. The ranking relation would now be $(0, 2, 3, 1)_{\preceq}$.

2.5 Suggested properties of an IRP method

There are some properties that an IRP method should respect. These include normalization and rank reversal (Hochbaum & Levin, 2006) but also the "Newcomer's Rush" problem. In this section, I will outline these properties and the problems that can arise if they are not achieved.

2.5.1 Normalization

Normalization or balancing is an attempt to address the problem of users only using a subset of the weight interval.

An example could be a user that votes 1 (highest permissible weight) on every image she reviews. Without normalization, an image reviewed by this user could stand a better chance of getting a higher score than one not reviewed by this user. To overcome this problem, the IRP method could interpret the user's voting as stating that the images should share rank, since she has given them equal weights. If this is the case, I will say that the IRP method supports normalization natively.

Another problem that may arise is an inflation in the weights used. If many users have an average above $\frac{1}{2}$ it could lead to a higher average of all weights. A perfect normalization would ensure an average of $\frac{1}{2}$ together with a full coverage of the permissible scale.

However it is impossible to ensure both of these properties. When stretching the weigths to fit the complete interval one removes the possibility of adjusting the average.

The original problem of normalization is to ensure that each user uses the complete scale interval. Because of this, I have chosen to normalize in a way that ensures full coverage of the scale, and hence leaves the average to the users.

Let $u \in U$ and $v \in V$. The normalized weight $norm(W)^{u,v}$ of the original weight $W^{u,v}$ can now be expressed as:

$$norm(W)^{u,v} := \frac{(W)^v - min(W)}{max(W) - min(W)}$$

This simple computation ensures that the complete interval is used, by subtracting the minimum and dividing by the size of the used interval. The lowest weight will now be 0 while the highest will be 1. Hence, all scores are in the interval $[0..1]$ and the complete interval is used.

The normalization is only possible to compute when $max(W') - min(W') \neq 0$. However, when $max(W') - min(W') = 0$ the weights must be constant and hence stretching the weights to the full weight interval is impossible. Instead normalization can be skipped in this particular situation, leaving all the weights constant.

This normalization workaround is not a complete solution since it only stretches the weights to fit the complete interval. Any user with at least one vote of weight 0 and at least one vote of weight 1 will be left untouched by this normalization workaround. The best situation would be for the IRP method to support normalization natively.

2.5.2 Prevention of Rank Reversal

Rank reversal as described in (Hochbaum, 2006) is the problem that a new image, with low weights only, can swap images in the top of the ranking. Nearly redundant images should not have the power to dominate the top of the rank. It is impossible to describe a generic workaround to this problem, because it depends on how weights influence one another, which will differ from one method to the other. The IRP method will have to support this natively.

2.5.3 Prevention of Newcomer's Rush

Some IRP methods stabilize the ranks of the images over time, as more and more votes are made, hence the higher the value $\mid V^i \mid$ the lower the effect of a vote on image i. This may introduce a problem I have called the "Newcomer's Rush". The problem is that a new image, with a low count of votes, can rise to the top of the rank very fast because of a small number of high weights. This problem can cause the top of a ranking to be dominated by "newcomers"[4].

A generic workaround for the Newcomer's Rush problem is to assign a default vote to each image the user has not yet voted on. The weight of this default vote could be the average of all the user's weights, $avg(W^u)$. Using this method, all the users who have not yet voted on the new image would contribute to its rank using default votes. If

[4]hence the name, "Newcomers Rush"

$avg(W^u) = \frac{1}{2}$ for all users, the new image would have a lot of default votes with weight $\frac{1}{2}$, hence holding it back and preventing it from rising to the top.

However this might prevent new changes altogether in very large instances. The method can be relaxed by only placing default votes from a selected number of users, e.g. $\log(n)$ or $\sqrt{n}$ users. This could lead to trouble, because the influence of the default vote would change depending on the users selected. This problem can be solved by simply adding control-users and have these assign the default votes with a weight of $\frac{1}{2}$ (or perhaps a weight matching the default over all weigths in the system).

However some methods rely on the rating of each user and adjusts a user's influence accordingly. Default votes might work unexpectedly with such methods and should be used with caution.

In this paper the instances are small, so the simple linear version of the method will be used when nothing else is specified.

3 The Average-Point Method

3.1 Group ranking using Average-Point

The Average-Point (AP) method is one of today's most popular group ranking methods[5]. Its simplicity makes it easy to implement, but as we shall see in this section, this simplicity leads to some serious problems.

The method works by producing a ranking relation $\preceq$ such that:

$$i, j \in I : i \preceq j \Leftrightarrow avg(W^i) \leq avg(W^j)$$

In other words, the method defines the rank of an image solely from the weight-average of its votes.

I have ranked example 2.4-1 using the AP method:

Example 3.1-2:

U/I	0	1	2	3
0	-	0.5	0.3	0.4
1	0.6	-	-	0.8
2	0.5	0.8	-	0.6
3	0.6	0.7	0.7	-
Sc	0.56	0.66	0.5	0.6

Example 2.4-1 from section 2.3 using the AP method

In this example, each user has added one image that she cannot vote on herself. User U_0 has contributed with image I_0, user U_1 with image I_1 and so on. The last row shows

[5]The first ten results of a search for "rate my" on www.google.com yields 6 websites that obviously uses the Average Point method. The resulting 4 sites does not have ranking.

the scores of each image. Since this method uses these scores to produce the ranking relation, the relation itself will be: $(2, 0, 3, 1)_{\preceq}$. Even though every user who has voted on both I_0 and I_2 has chosen to rate I_0 lowest, I_2 ends up lowest in the final ranking. This is due to one of the shortcomings of the AP method. The method does not compare the vote to the user's other votes and misses out on this information. A vote of 6, when compared to the user's other votes of 7 and 8 on the alternatives, might seem better than a vote of 5 compared to 3 and 4. This problem is caused by the lack of normalization.

3.2 Image Ranking using Average-Point

In this section I will take a look at the properties suggested in section 2.5 and describe how the AP method is influenced by them.

3.2.1 Normalization

The AP method itself does not provide normalization. However the generic normalization workaround from section 2.5.1 can be used. As an example I have applied normalization to example 2.4-1:

Example 3.2-3:

U/I	0	1	2	3
0	-	1	0	0.5
1	0	-	-	1
2	0	1	-	0.33
3	0	1	1	-
Sc	0	1	0.5	0.61

Example 2.4-1 using the AP method with normalization

This results in a final ranking of $(0, 2, 3, 1)_{\preceq}$. The main difference from the previous ranking is that I_0 now has a lower score than I_2. Apart from this, the scores are further apart from each other, as a result of stretching the users' choices to fit the complete interval.

3.2.2 Rank Reversal

Since the AP method computes the scores of each image using votes on the particular image only, it is not subject to rank reversal. This is a result of simplicity, since adding a new image cannot influence the other images' scores in any way, hence it cannot influence the other images' pairwise ranking.

3.2.3 Newcomer's Rush

The AP method is subject to the Newcomer's Rush problem. To show this I have added a new image to example 2.4-1:

Example 3.2-4:

U/I	0	1	2	3	4
0	-	0.5	0.3	0.4	1
1	0.6	-	-	0.8	-
2	0.5	0.8	-	0.6	-
3	0.6	0.7	0.7	-	-
Sc	0.57	0.67	0.5	0.6	1

Example 2.4-1 expanded with an extra image

The new image I_4 only has a single vote and this vote now dominates its score, and hence its ranking. The resulting scores show that the new image will take the lead in the final ranking. This problem could be relaxed using default votes, as introduced in section 2.5.3. After normalization, the table would look like this:

Example 3.2-5:

U/I	0	1	2	3	4
0	-	1	0	0.5	1
1	0	-	0.5	1	0.5
2	0	1	-	0.33	0.44
3	0	1	1	-	0.67
Sc	0	1	0.5	0.61	0.6525

Example 3.2-4 with default votes and normalization

The new image is no longer the highest ranked, because it is dominated by the default votes instead of a single vote. The high 1.0-weight does not dominate the ranking anymore.

4 The PageRank Method

4.1 Group ranking using PageRank

The Page-Rank (PR) method (Page, 1999) was originally developed to rank websites based on their incoming hyperlinks. The method works by organizing every website in a directed flow graph, $G = (V_G, E_G)$, so that the edge from (v_i, v_j) exists iff the website represented by v_i links to the website represented by v_j. Let $\delta^-(v_i)$ denote all the incoming edges to v_i, and $\delta^+(v_i)$ denote all outgoing edges from v_i. The flow of edge (v_i, v_j) is now defined as:

$$f(v_i, v_j) := \frac{1}{\mid \delta^-(v_i) \mid}$$

describing how many percent of v_i's flow is propagated through v_j.

The main goal of this method was to propagate the ranking through the links in such a way that an incoming link from a highly ranked website is worth more than one from a lowly ranked website. The PageRank of a particular vertex is the sum of its incoming flow:

$$PR(v_i) = \sum_{v \in \delta^-(v_i)} f(v)$$

The result of the PageRank method is a vector of positive scores, where each score is in the range $[0..1]$. The score vector is normalized and hence the sum of all scores is 1.

Since PR was originally intended for websites, which either linked to another site or not, the weight set in this method is $\{0, 1\}$.

4.1.1 Random-Surfer model

The Random-Surfer model is used in the PR model to simulate the probability that a surfer jumps to a random website, instead of following an outgoing link (Page, 1999). The probability is determined by the factor $d = 0.85$, which states that the probability of a user leaving the website through a link is 85% while the probability of a user leaving the website by jumping to any known page is 15%. In particular, the PageRank of a vertex while using the Random-Surfer model is:

$$PR(v_i) = \frac{d}{N} \cdot \sum_{v \in \delta^-(v_i)} f(v) + \frac{1-d}{N}$$

where N is the total number of known websites.

The Random-Surfer model extends the original method in a way that eliminates flow-sinks, because all vertices have N outgoing edges, which ensures continuous distribution of the flow. It will also ensure that each vertice has at least $\frac{1-d}{N}$ flow to distribute, even if it has no incoming edges, and hence no incoming flow.

4.1.2 Computing the ranking relation

In order to compute the ranking relation using the PR method a score vector is computed from the flow matrix A, which is defined from the flow graph. The score vector is computed by computing the eigenvector of A, such that $AR = \lambda R$ (Page, 1999).

The power method (Leon, 2006) is used to iteratively compute the eigenvector, x_1 of A. If $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of A and $x_1, x_2, \ldots, x_n$ the corresponding eigenvectors and:

$$\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n$$

the power method will approximate x_1 from any non-zero vector R_0, using the recursion:

$$R_{i+1} = \frac{A \cdot R_i}{|| \; A \cdot R_i \; ||_1}$$

The matrix A must have an eigenvalue λ_1 which is strictly greater than any other eigenvalue of A. It is shown by Perron's theorem (Leon, 2006) that any positive $n \times n$-matrix has such an eigenvalue. The random-surfer model ensures that all values of A are at least $\frac{d}{N}$, which implies that A is positive.

Each iteration can be done in two simple steps, the first being the calculation of the product AR_i, followed by normalization of the resulting vector. When dealing with the PR method, this normalization can be achieved by simply maintaining the norm, since $|| R_0 ||_1 = 1$. This can be achieved by adding $(|| R_i ||_1 - || R_{i+1} ||_1) \cdot E$ to R_{i+1}, where E is a normalized vector, e.g. R_0. This will maintain a norm of 1 throughout each iteration. This can be proved using simple induction over the number of iterations:
In the 0th iteration, R_0 is normalized because $\sum R_0 = 1$.
In the ith iteration, let $d := || R_{i-1} ||_1 - || R_i ||_1$ and $|| R_{i-1} ||_1 = 1$ then:

$$|| R_i + d \cdot R_0 ||_1 = || R_i ||_1 + || d \cdot E ||_1$$

$$= || R_i ||_1 + d \cdot || E ||_1 = || R_i ||_1 + d$$

$$= || R_i ||_1 + || R_{i-1} ||_1 - || R_i ||_1 = || R_{i-1} ||_1 = 1$$

Hence the norm is maintained in each iteration and the R_i vector is a unit vector. However the factor d will always equal 0, because each element in A is in the range $[\frac{1-d}{N}..1]$ (garenteed by the Random-Surfer model) and each of A's columns sums to 1:

$$d = || R_{i-1} ||_1 - || R_i ||_1 = 1 - || R_i ||_1$$

Taking a closer look at $|| R_i ||_1$ yields:

$$|| R_i ||_1 = \sum_{j=0}^{N} | R_i^j | = \sum_{j=0}^{N} R_i^j = \sum_{j=0}^{N} [A \cdot R_{i-1}]^j = \sum_{r=0}^{N} \sum_{c=0}^{N} A^{r,c} \cdot R_{i-1}^c$$

$$= \sum_{c=0}^{N} \sum_{r=0}^{N} A^{r,c} \cdot R_{i-1}^c = \sum_{c=0}^{N} (R_{i-1}^c \cdot \sum_{r=0}^{N} A^{r,c}) = \sum_{c=0}^{N} (R_{i-1}^c \cdot 1) = || R_{i-1} ||_1 = 1$$

It is now clear that $d = 1 - 1 = 0$ and that it is safe to skip the normalization step of the algorithm described in (Page, 1999) when applying the Random-Surfer model. (Page, 1999) describes a method for applying custom user rankings when using the PageRank method by changing the values of the vector E. The idea is that a value of E describes the possibility of a random surfer choosing to visit its associated site. However, as we have seen, E can be safely removed from the computations when applying the Random-Surfer model. The conclusion is that the PageRank method does not support custom user rankings when using the Random-Surfer model.

The criteria of the power method are satisfied and the sequence $\{R_i\}$ will converge against the eigenvector x_1, such that:

$$A \cdot R = \lambda_1 x_1 = \lambda_1 R$$

Perron's theorem states that the eigenvector R is positive and since it is also normalized, we must have $x \in R \Rightarrow x \in [0..1]$ which corresponds to the expected result of the PageRank method.

The approximation will converge with the same speed as $\frac{\lambda_1}{\lambda_2}$ (Leon, 2006) . According to (Page, 1999), the method uses roughly $\log(n)$ iterations when applied on a large graph of websites. However the graph of websites is very sparse, and hence the computation will converge faster than when used on a more complicated, dense graph. A pleasant property is that choosing R_0 close to x_1 will ensure faster convergence. When dealing with ranking, it is very likely that the instance has only gone through minor changes compared to the size of the instance, and hence using the previous eigenvector when reranking the instance might speed up the process significantly.

When the score relation Sc has been computed, the ranking relation is easily defined from it:

$$\forall i, j \in I : i \preceq j \Leftrightarrow Sc(i) \leq Sc(j)$$

4.2 Image Ranking using PageRank

Since the PR method uses the weight set $\{0, 1\}$, it will need some modification in order to become an IRP-method. The first modification will allow for weighted voting in the interval $[0..1]$. The method uses the graph, G, to define the flow between vertices. In the original PR method, the flow of v_i is shared equally amongst all websites that have incoming links from v_i. In order to allow weights in $[0..1]$, this flow has to be split unequally in such a way that images with higher weights get more flow. This can be done by defining the flow of edges so that:

$$f(V_G^{u \in U}, V_G^{\{i | \exists V^{u,i}\}}) = \frac{d}{N} \cdot \frac{W^{u,i}}{\sum(W^u)} + \frac{1-d}{N}$$

where V_G refers to a vertex in G and V is the set of votes.

This will assign flow to each image that user u has voted on. The flow denotes how many percent of the user's flow the image should get.

Since images do not vote, they will give their flow back to their owner. This prevents the flow from getting stuck:

$$f(V^{i \in I}, V^{U^i}) = \frac{d}{N} \cdot 1.0 + \frac{1-d}{N}$$

A problem arising is that an image will get a higher rank, no matter what vote a user places on it. This is the result of more flow passing through the image. In the standard method, no vote implies a vote of $\frac{1-d}{N}$. Another related problem is that whenever a user votes on a new image, this image gets some of the user's flow, hence some of her influence. The rest of the images that she has voted on will now get a lower score. In order to workaround these problems, I will use default votes in this method, as described in section 2.5.3.

To see how this method works, I have ranked example 2.4-1:

Example 4.2-6:

U/I	0	1	2	3
0	-	0.5	0.3	0.4
1	0.6	-	0.5[6]	0.8
2	0.5	0.8	-	0.6
3	0.6	0.7	0.7	-
Sc	0.115	0.140	0.113	0.132

Example 2.4-1 ranked using the PR method

The ranking relation derived from these scores is $(2, 0, 3, 1)_{\preceq}$ and is identical to the ranking relation computed using the normalized AP method. The PA method also returns scores on each user, which can be used to rank the users. In this case, the users' ranking relation is $(2, 0, 3, 1)_{\preceq}$. Since every user has added one image only, their scores equal the score of this image and the ranking relation is identical to that of the images.

4.2.1 Normalization

The PR method does not support normalization natively, since it was designed for binary weights[7]. Because of this, there was no need for normalization in the original method. However, normalization can be achieved in the IRP method using the general normalization method as described in section 2.5.1.

As an example of PR with normalization I have ranked the previous example with normalization applied:

Example 4.2-7:

U/I	0	1	2	3
0	-	1	0	0.5
1	0.66	-	0	1
2	0	1	-	0.33
3	0	1	1	-
Sc	0.056	0.186	0.088	0.170

Example 5.2-6 with normalization

Defining a ranking relation from these scores yields $(0, 2, 3, 1)_{\preceq}$. Notice how the two lowest ranked images, I_0 and I_2, swapped when compared to the ranking relation achieved without normalization.

When relating the scores of I_0 and I_2 to the flow network of the PR method, one notices that I_0 is dominated by U_1 and that I_2 is dominated by U_3. However, U_1 will obtain a higher score than U_2 because of the high score of I_1. Because of this, the vote of U_1 will have more influence than the vote of I_3.

To show this, I have reranked the instance, setting $W^{U_1,I_0} = 0.75$:

[7]weights that are either 0 or 1

Example 4.2-8:

U/I	0	1	2	3
0	-	1	0	0.5
1	0.75	-	0	1
2	0	1	-	0.33
3	0	1	1	-
Sc	0.083	0.180	0.080	0.149

Example 4.2-7 reranked with $W^{U_1,I_0} = 0.75$

The ranking relation is now be $(2,0,3,1)_{\preceq}$. What looks like a minor change in a single weight was enough to swap the ranks of I_0 and I_2. Notice how the U_1 contributes with $\frac{75}{75+100} \approx 43$ percent of its flow to I_0 and that this is enough to beat the 50 percent with which U_3 contributes to I_2.

4.2.2 Rank Reversal

The PR method does not prevent rank reversal. To illustrate how rank reversal can influence the ranking when using the PR method, I have created a small example inspired by the rank reversal example from (Hochbaum, 2006). I have ranked a small instance with only 3 users and 3 images. Each user has one image, as in the previous examples:

Example 4.2-9:

U/I	0	1	2
0	-	0.5	0.3
1	0.5	-	0.5
2	0.3	0.4	-
Sc	0.160	0.186	0.154

A small 3 by 3 instance

The resulting ranking relation is $(2,0,1)_{\preceq}$. I will now add a new image to the instance. The new image will have low votes only, hence the image should only have minor influence on the resulting ranking relation:

Example 4.2-10:

U/I	0	1	2	3
0	-	0.5	0.3	0.1
1	0.5	-	0.5	0.1
2	0.3	0.4	-	0.1
3	0.8	0	0	-
Sc	0.172	0.148	0.121	0.058

Example 4.2-9 after adding a new image

Notice that the new user, U_3, has obtained a very low score as a consequence of the low rated image. However, the new ranking relation has changed to $(3, 2, 1, 0)_{\preceq}$, which shows that I_0 and I_1 have swapped. The new user, with a low rank and a single lowly ranked image managed to swap number 1 for number 2. However, user U_1 could regain the first place by simply boycotting the new user by voting 0 on her image. This would lower the influence of user U_3, and as a consequence lower the score of I_0. In fact, this would lower the score of I_0 to 0.140 and at the same time raise the score of I_1 to 0.157, hence swapping the two users back in the ranking relation.

4.2.3 Newcomer's Rush

The PR method does not prevent Newcomer's Rush, but the generic workaround for this problem can be used as a partial solution. The workaround will keep back newly added images to some extent, but the effect of the workaround will vary from case to case. The reason for this is to be found in the way PR propagates the flow. The workaround addresses the problem by adding default votes that will keep back the rise of newly added images. However, the effect of these default votes will depend on the user rankings.

When a new image is added to an existing instance, where the generic workaround has been applied, it will get a default vote from every user. However as a result of dynamic user scores these default votes will have different influence. Because of this, it is hard to predict how the instance will react to the new image. If a user with a high user score assigns a high vote to the image, it could make the image rush to the top, if the other users' scores are low enough.

If the owner of the new image now assigns a vote of 1 on the voting user's images and a vote of 0 on every other image (to avoid the default votes), this would lead most of the flow back to the user who voted on the new image, hence the new image would get even more flow which leads to a higher score.

To illustrate this scenario, I have ranked an instance that expands from 4 to 5 users. Below is the original normalized 4-users-instance:

Example 4.2-11:

U/I	0	1	2	3
0	-	1	.4	0
1	0	-	.75	1
2	1	0	-	0.667
3	0	1	0.8	-
Sc	0.089	0.138	0.141	0.132

A normalized 4 users by 4 images instance

The corresponding ranking relation $(0, 3, 1, 2)_{\preceq}$. I will now add a new image, I_4 and add a vote of 1 from the highest ranked user, U_2, to this image. Furthermore, to maximize the flow through I_4, U_4 will add a vote of 1 to I_2 and votes of 0 to the other images:

Example 4.2-12:

U/I	0	1	2	3	4
0	-	1	0.4	0	0.466
1	0	-	0.75	1	0.583
2	1	0	-	0.667	1
3	0	1	0.8	-	0.6
4	0	0	1	0	-
Sc	0.065	0.076	0.168	0.077	0.114

Example 4.2-11 after adding a new image

The ranking relation is now $(0, 1, 3, 4, 2)_{\preceq}$. Notice that I_2 is still the highest ranked image, but that I_4 has managed to gain a second place. The other images have dropped considerably in score, depending on how much flow they are sending through I_4. There is a partial sink consisting of the images I_2 and I_4, and of the users U_2 and U_4. I call this a partial sink because it leaks flow to other images and users, while keeping a lot of the flow to itself in each iteration of the flow-network computations. All this is happening without U_2 participating in any other way than placing a single vote. The partial sink itself is established solely by U_4.

5 The Close Rankings Method

5.1 Group ranking using Close Rankings

The Close Rankings (CR) method (Hochbaum, 2006) solves the general intensity-only group ranking problem using methods from operation research. The method attempts to minimize the distance between the final ranking and the ranking of each individual user. This is achieved by defining an optimization problem that minimizes this distance.

Formally this problem is defined as:

$$\begin{array}{lll} \texttt{Min} & \sum_{i<j} F_{ij}(z_{ij}) & & (1) \\ \texttt{subject to} & x_i - x_j = z_{ij} & \texttt{for } i < j & (2) \\ & -n \leq x_j \leq n & j = 1, \ldots, n & (3) \end{array}$$

where F is a convex function and x_i is the score of image i.

After solving this optimization problem, the ranking relation $\preceq$ can be defined from the x variables, which contain the scores:

$$I_i \preceq I_j \Leftrightarrow x_i \leq x_j$$

In (Hochbaum, 2006) the following convex function F_{ij} is proposed:

$$F_{ij}(z_{ij}) := \sum_{l \in L} R^l_{i,j} \cdot \left(D^l_{i,j} - z_{ij}\right)^2$$

where $R^{i,j}$ is the rank confidence, $D^{i,j}$ is the user's preferred difference of i and j. L is the set of reviewers and z_{ij} is the difference variable to which the optimization solver will have to assign a value.

5.1.1 Proposed improvements

I have found a problem with the objective function proposed in (Hochbaum, 2006). The function has the problem of not differentiating between a positive or negative difference. Assuming some reviewer has chosen $D^{i,j} = 2$. The contribution to the objective function would be $R^{i,j} \cdot (2 - z_{ij})^2$. The problem is that $(2+1)^2 = (2-5)^2 = 9$. This might not seem problematic at first, but notice that the contribution to the objective value is equal whether the reviewer's chosen rank is reversed or not. In order to prevent this I have chosen to modify the proposed function to the new function:

$$F_{ij}(z_{ij}) := \sum_{l \in L} \frac{3}{4} R^l_{i,j} \big(D^l_{i,j} - z_{ij}\big)^2 + \frac{1}{4} R^l_{i,j} \big(D^l_{i,j} + sgn(D^l_{i,j}) - z_{ij}\big)^2$$

where $sgn(x) := \frac{x}{abs(x)}$. The new function rewards the objective value for choosing a difference that corresponds to the direction the user has chosen. Examining the example from earlier, we now get ($R^l_{i,j} = 1$):

$$\frac{3}{4}(2+1)^2 + \frac{1}{4}(2+1+1)^2 = \frac{3}{4}9 + \frac{1}{4}16 = 6.75 + 4 = 10.75$$

whereas:

$$\frac{3}{4}(2-5)^2 + \frac{1}{4}(2-5+1)^2 = \frac{3}{4}9 + \frac{1}{4}4 = 6.75 + 1 = 7.75$$

Intuitively this can be interpreted as placing $\frac{3}{4}$th of the relevance on the user's own choice, and $\frac{1}{4}$th of the relevance on an exaggerated version of the same vote. This gives the function a hint of direction.

This direction will only be computable for $D \neq 0$, since the sign function sgn is only computable for $x \neq 0$. However when $D = 0$ the reviewer wants i and j to share rank, hence there is no direction to hint and the original object function can be used in this case.

5.1.2 Computing the ranking relation

In order to compute the ranking relation using the CR method a score vector is computed by solving the optimization problem introduced in section 5.1. To solve the problem, it is first converted to an unrestricted minimization problem with a quadratic object function (Hochbaum, 2006). This is done by simply substituting z_{ij} with $x_i - x_j$. This removes all z variables from the problem and hence eliminates all constraints. In order to avoid the remaining boundary constraints, $-n \leq x_j \leq n$, x_1 is fixed to 0, which guarentees that all x variables are in this particular range (Hochbaum, 2006).

As an example of this conversion, consider the original optimization problem:

$$\begin{aligned} &\texttt{Min} && \textstyle\sum_{i<j} F_{ij}(z_{ij}) && (4)\\ &\texttt{subject to} && x_i - x_j = z_{ij} \quad \texttt{for } i<j && (5)\\ & && -n \leq x_j \leq n \quad j = 1,\dots,n && (6) \end{aligned}$$

This problem will be converted to:

$$\texttt{Min} \quad \sum_{i<j} F_{ij}(x_i - x_j) \tag{7}$$

where x_1 is replaced by 0. Since F_{ij} is quadratic, the resulting function is quadratic separable and can be solved with any general method for unconstrained convex optimization. In (Hochbaum, 2006), Newton's method and the conjugate gradient method are proposed. Newton's method will yield the optimal solution in a single iteration, because of the function being quadratic. The conjugate gradient method will terminate after n lines' search, using $O(nk)$ time to compute the search direction in the k'th iteration and using a total of $O(n^3)$ time.

The conjugate gradient method (Heath, 2002) has the advantage of using several iterations, which can be used to speed up the process of reranking a previously ranked instance, by using the previous solution to the instance as an initial solution. This might speed up the process of reranking a previously ranked instance, if reranking is performed adequately frequently.

The conjugate gradient method works by modifying the search direction from the subgradient method that converges against the global minimum by subtracting the current gradient from the current position. In order to describe the conjugate gradient method, I start by explaining the subgradient method.

The subgradient method locates the minimum of the convex function f by repeating the following iteration:

$$s_i := -\Delta f(x_i)$$

$$x_{i+1} := x_i + \alpha s_i$$

Δf is the first derivate of f, s_i is the i'th search direction, x_i is the i'th point and α_i is the i'th stepsize.

This iteration looks rather simple, however there is the problem of choosing the value of α_i in each iteration. The aim is to choose this value in such a way that $f(x_i + \alpha s_i)$ is minimized. So far, I will ignore this and say that α_i minimizes this expression in each iteration and instead move on to show how the iterations can be expanded into the conjugate gradient method. I will however return to the subject and explain how α_i can be chosen using golden section search.

The conjugate gradient method works like the subgradient method, except in the way the search direction, s_i, is chosen. Initially s_0 is set to the negative derivate, $-f'(x_0)$. In each iteration, the quantity β_i is computed, and the new search direction s_i can is now computed as $s_i := -f'(x_i) + \beta_i s_{i-1}$.

The iteration of the conjugate gradient method is:

$$x_{i+1} := x_i + \alpha_i \cdot s_i$$

$$g_{i+1} := \Delta f(x_{i+1})$$

$$\beta_{i+1} := \frac{g_{i+1}^T \cdot g_{i+1}}{g_i^T \cdot g_i}$$

$$s_{i+1} := -g_{i+1} + \beta_{i+1} \cdot s_i$$

where $s_0 := -\Delta f(x_0)$.

The conjugate gradient method has a running time of $O(n)$ iterations where n is the number of dimensions. However this running time depends on α_i minimizing the expression $f(x_i + \alpha_i \cdot s_i)$ a one-dimensional minimization (only variable being α_i), which can be achieved using golden gection search.

The golden section search is a one-dimensional minimization method that has the advantage of only recomputing the function value in one point in each iteration. This can be achieved by computing the function values in a, x_1 and b such that $a < x_1 < b$. The point, x_2 is now chosen from the largest interval of $[a, x_1]$ and $[x_1, b]$. As an example, say that x_2 is chosen from $[x_1, b]$ and let the minimum of the interval $[a, b]$ be δ, then:

$$f(x_1) < f(x_2) \Rightarrow \delta \in [a, x_2]$$

$$f(x_1) > f(x_2) \Rightarrow \delta \in [x_1, b]$$

Let $[a_i, b_i]$ be the interval in iteration i. In each iteration, we change either a_i or b_i moving the two points closer to each other. We also have that $a_i < b_i$ in each iteration. Hence $b_i - a_i$ must converge against 0, therefore $[a_i, b_i]$ must converge against some point ϵ. Since the iteration invariant states that $[a_i, b_i]$ will contain the minimum, this minimum must be ϵ and hence $\delta = \epsilon$.

The remaining problem is that the golden section search (so far) might not converge at a consistent rate. To ensure consistent convergence we need to reduce the length of the interval by the same fraction at each iteration. This can be achieved by fixing the relative positions of the two points x_1 and x_2 to τ and $1 - \tau$, with $\tau = \frac{\sqrt{5}-1}{2}$. This choice ensures that the new subinterval chosen is τ relative to the previous interval, and that the interior point will be at position τ or $1 - \tau$ relative to the new interval.

This ensures a steady convergence and gives us the needed optimization for the conjugate gradient method.

5.2 Image Ranking using Close Rankings

The CR method is easily applied as an IRP method. The only change that has to be made is in the definition of the F_{ij} function. This function will now have to use the variables available in the IRP. Given a pair of images, $i, j \in I$, it will need to compute the sum of the weight differences for each user, $D^u_{i,j} := W^{u,i} - W^{u,j}$ factored by the vote relevance, $R^u_{i,j}$. The function can now be defined as:

$$F_{i,j \in I}(z_{ij}) := \sum_{u \in U} \left(\frac{3}{4} R^u_{i,j} \left(D^u_{i,j} - z_{ij} \right)^2 + \frac{1}{4} R^u_{i,j} \left(D^u_{i,j} + sgn(D^u_{i,j}) - z_{ij} \right)^2 \right)$$

As an example of a ranking produced by this method, I have ranked example 2.4-1 ($R^u_{i,j} = 1$):

Example 5.2-13:

U/I	0	1	2	3
0	-	0.5	0.3	0.4
1	0.6	-	-	0.8
2	0.5	0.8	-	0.6
3	0.6	0.7	0.7	-
Sc	0	0.481	0.044	0.325

Example 2.4-1 ranked with the CR method

The ranking relation produced from these scores is $(0, 2, 3, 1)_{\preceq}$. This ranking relation is equal to the ranking relation achieved using the AP method with generic normalization applied.

5.2.1 Differentiating rankings using confidence factors

So far, I have ignored the rank confidence factor, R, by fixing it to 1. The rank confidence factor is used to scale the penalty of each rank comparison in the objective function, to allow more differentiated rankings.

Each confidence factor is in the range $[0..1]$ and it is possible to associate such a factor to any or all comparisons. Because the confidence factors can differ for each comparison, they can be used to respect the relevance of each particular vote. This can be achieved by simply defining the confidence factors from the relation from votes into relevances.

The score relation can be defined by any variable associated with votes. In this paper, I will look at a function defined by the age of the two votes compared in days, A^v and the rank of the voting user. The rank of the voting user can be used directly, as it is a number in the range $[0..1]$. However using the age of a particular vote needs a little more work, as it can be any positive number.

I will start by defining a function, AR, that converts a given age in days into a usable relevance in the range $[0..1]$:

$$a < 10 \Rightarrow AR(a) = 1$$

$$a \geq 10 \Rightarrow AR(a) = \frac{1}{\log_{10}(a)}$$

I can now define the confidence factor R^u_{ij} as follows:

$$R^u_{ij} = ScU^u \cdot AR(A^{V^{u,i}}) \cdot AR(A^{V^{u,j}})$$

This will ensure that all votes older than 10 days will start to loose influence as a function of their age. I have chosen to use the logarithmic function, in order to ensure that the rate which the influence is reduced slows down by time. This ensures that the vote will never loose all its influence. As an example, a vote which is 14 days old will loose 13 percent of its influence one that is 30 days old will loose 32 percent of its influence and one that is 365 days old will loose 61 percent of its influence.

This kind of differentiated ranking could be interesting in systems that want to focus on the newest trends among users by giving newer votes more influence than older votes.

(Hochbaum, 2006) proposed that confidence factors could be defined by the difference D in the particular vote, such that a higher value of D ment a higher confidence. This would imply that votes with higkh intensities would be given more confidence.

Another way of differentiating could be by defining the confidence factor from the number of categories shared by two images. A comparison between two different portraits could gain more influence, than one between a portrait and a landscape photography. One could go even further and say that a comparison between two portraits of the same person should gain even higher influence. This method allows one to filter out comparisons, that might have no relevance at all. It might be meaningless to compare a portrait to a landscape photography, and hence the competition between such two images might not be interesting. This method allows the system to compensate for such situations in a well-defined manner.

5.2.2 Normalization

The CR method ensures normalization because it converts the given weights into distances between images before ranking. Because of this, it does not matter whether a user only uses a subset of the weight interval, since the weights are not used directly.

5.2.3 Rank Reversal

The CR method has a native feature to prevent rank reversal (Hochbaum, 2006). This works by simply adding constraints of the form $x_i \leq x_j$, where x_i has a lower rank than x_j in the original ranking. By doing this, the original ranking is locked while ranking new images. The problem with this method is that it only works as long as the original images do not need to be reranked. The method will not work if the original ranking is supposed to change, hence this method will not work in any dynamic system. Since image ranking, as described in this paper, is very dynamic (users can change a vote whenever they like) there is no way to prevent rank reversal using this method.

5.2.4 Newcomer's Rush

The CR method does not prevent the Newcomer's Rush problem. If only one user has voted on an image, this vote will dominate the rank of the image. However, because of normalization the vote will have to be high compared to the user's other votes. This problem can however be relaxed by using the default votes method as described in section 2.5.3. Since the CR method uses the distances between the weights, and not the weights themselves, this would imply that images that a user has not yet voted on should be ranked equally.

6 Comparison of Ranking Methods

In this section I will compare the three ranking methods, by ranking different types of instances. The purpose is to illustrate the differences between the three methods, while discussing the pros and cons associated with each one of them.

The instances I have chosen to discuss are 1) the simple instances, where the ranking is based on weights-only, hence ignoring user rankings and vote relevance, and 2) the more advanced instances, where user rankings and vote relevance are taken into account.

The instances ranked in this section are more realistic than those ranked earlier in the paper, as they will have more than just one image per user.

In the comparisons the AP method and the PR method are used with normalization applied. The CR method is used with the modification proposed in section 5.1.1.

To ease comparison between methods, I list the ranking relation for each method instead of the individual scores.

6.1 Ranking simple instances

When ranking simple weight-only systems with no user rankings nor vote relevance, the AP method and the PR method will return ranking relations that are similar. The flow in the PR method might have some effect, but the two methods are very similar in the way they rank each individual image. The main difference is the PR method's dynamic user rankings.

The CR method might give a very different result, since it ranks the whole system at once and hence tries to respect all votes at once, not looking at any individual image at any time.

Example 6.1-14 is a ranking of an instance with 5 users who have submitted a total of 12 images. Each user has voted on every image, except for her own:

Example 6.1-14:

U/I	0	1	2	3	4	5	6	7	8	9	10	11
0	-	0.7	0.6	0.8	0.4	-	0.8	0.5	0.7	0.4	-	0.7
1	0.9	-	0.1	0.5	0.5	0.8	-	0.3	0.7	0.9	0.7	-
2	0.3	0.6	-	0.5	0.1	0	0.5	-	0.1	0.8	0.3	0
3	0.1	0.3	0.2	-	0.6	0.8	0.6	0.1	-	0.8	0.7	0.7
4	0.7	0.7	0.2	0.4	-	0.4	0.4	0.8	0.9	-	0	0.1
AP	7	5	12	4	11	6	2	10	3	1	8	9
PR	7	5	12	4	11	6	3	10	2	1	8	9
CR	7	2	12	5	10	6	3	11	4	1	8	9

A dense 5 users by 12 images instance.
The last three lines show for each method the image's position in the ranking relation.

Notice the similarity between the ranking relations produced by the AP method and the PR method. The only difference here is whether I_8 or I_6 should be number 2. The CR method is a bit more different, but this is mainly because it sets I_1 as number 2, hence shifting the rest of the ranking relation.

It is very difficult to see why the three methods differ the way they do, because of the complexity of the PR and CR methods. However, it is possible to investigate how much they differ. One interesting number to compare is the number of pairwise preference reverses, when comparing the final ranking relation to each user's preferred ranking relation. This method is used in (Hochbaum, 2006) as part of the discussion of the CR method. The following table shows the number of pairwise preference reverses for each method:

AP	PR	CR	CR′
76	77	74	75

The Close Rankings method has managed to reverse less user preferences than the other methods, by placing I_1 at number 2. It is not surprising that CR does well in this test, since it is the only method that ranks by trying to lower the number of preference reverses. The CR′ entry represents the original Close Rankings method as proposed in (Hochbaum, 2006) without the modification suggested in section 5.1.1. Applying the modification will result in the reversion of one less pairwise comparison.

6.2 Ranking with user rankings

In this section I introduce a vector of user scores and rerank the instance used in section 6.1. I will rerank the instance using the CR ranking method only, since it is the only method among the three ones discussed methods that supports custom user scores. The user scores are defined in the vector R:

$$ScU = \{(U_0, 0.8), (U_1, 0.3), (U_2, 0.5), (U_3, 0.2), (U_4, 0.6)\}$$

After reranking the instance with these scores, the following result is achieved:

Example 6.2-15:

M/I	0	1	2	3	4	5	6	7	8	9	10	11
AP	7	5	12	4	11	6	2	10	3	1	8	9
PR	7	5	12	4	11	6	3	10	2	1	8	9
CR	6	1	12	5	11	8	2	7	3	4	10	9

The most significant change is that I_1 is now the highest ranked by CR, while I_9 has dropped to number 4. This change can be explained by looking closer at the chosen user scores. Notice that U_0 and U_4 are the highest ranked users and that they both have a fairly high vote on I_1. U_0 has the highest influence, and also the largest positive value for the z_{19} variable in the CR objective function, namely $0.7 - 0.4 = 0.3$. In short, the users who preferred I_1 to I_9 have been given a high rank. As a result of respecting the user scores, the number of pairwise preference reverses has increased to 80. However, if one adjusts the counter to respect user scores, by adding the user score to the counter instead of 1, then we get the following result of pairwise preference reverses:

AP	PR	CR
39.8	37.4	35.8

As can be seen, the numbers of pairwise reverses are quite close, even though the CR method has changed to reflect the user rankings, while the other methods have not. The PR method and the CR method deviate by 4.5% against the 4.1% earlier.

6.3 Ranking with vote relevance

I will now rerank the instance from section 6.2 with vote relevance, by first assigning an age to each vote, and then defining the relevance from the ages using the function introduced in section 5.2.1.

The ages have been chosen in a way that should inflict a change in the position of I_1. In order to achieve this, the votes on I_1 have been given ages according to their weights, such that votes with high weights are old, while votes with low weights are young. The idea is that I_9 will regain the highest rank, because the old votes on I_1 will have low relevance.

The ages can be seen in the following table:

U/I	0	1	2	3	4	5	6	7	8	9	10	11
0	-	240	55	131	0	-	3	91	11	85	-	25
1	22	-	12	87	12	23	-	64	20	17	8	-
2	91	150	-	41	5	39	6	-	31	6	23	18
3	9	7	3	-	3	8	4	191	-	10	1	9
4	11	129	8	19	-	12	23	101	44	-	12	52

Below is a table of the ranking relation produced by CR, CR with user rankings and CR with both user rankings and vote ages:

Example 6.3-16:

M/I	0	1	2	3	4	5	6	7	8	9	10	11
CR	7	2	12	5	10	6	3	11	4	1	8	9
CR_{u}	6	1	12	5	11	8	3	7	2	4	10	9
$\mathrm{CR}_{\mathrm{u,v}}$	6	3	12	5	11	7	2	8	4	1	10	9

As a consequence of the carefully placed ages, I_9 has regained the highest rank while I_1 has dropped to a third place. This example shows how the CR method can be influenced by any factor that can be described as a parameter of a user and an image pair. In this case, I chose to use the ages of each vote, but there is no limit to how complex the ranking could be made using this simple approach.

7 Testing Running Time

In this section, I will test the practical running time of the PageRank method and the Close Rankings method, when modifying the instance beeing ranked. The purpose is to find out how many iterations the two methods would need in order to rerank a previously ranked instance when this instance has undergone a slight change.

When ranking example 6.1-14 the PageRank method used 42 iterations, while the Close Rankings method used 5. The two methods both have a running time of $O(i \cdot n^2)$ where i is the number of iterations, which makes the number of iterations interesting, since this is what distinguises the two running times.

I have reranked example 6.1-14 after changing $1, 2, 4, 8$, and 16% of the votes using the solution from the original problem as the initial solution. The number of iterations used to rerank the instance is shown in the following table:

M/P	0%	1%	2%	4%	8%	16%
PR	1	5	5	7	9	5
CR	1	4	4	4	4	5

Both methods use very few iterations to adjust to previous solution to the changed instance. However, the instance is quite small. To test this on a larger scale, I have created a randomly generated instance with 50 users and a total of 500 images. The CR method used 4 iterations when ranking this instance the first time, while the PR method used 46 iterations.

M/P	0%	1%	2%	4%	8%	16%
PR	3	3	3	3	4	1
CR	1	4	2	4	4	2

None of the methods use anything near the original number of iterations, when reranking the instance. Both methods use less than 4 iterations to adapt the solution to the new instance. As can be seen, the CR method performs very well on this instance, using only 4 iterations against the PR method's 46.

The implementations used in this paper are very simple, both being almost direct implementations of the pseudo code. Because of this, the number of iterations might be dramatically higher than the number needed in an optimized implementation. Especially the performance of the CR method might be improved by use of optimization heuristics and cuts.

8 Real Life Example

In order to provide a real life example I have created a survey regarding the teaching quality provided by lectures and professors at DIKU [8]. The survey was very simple. All the participant had to do was to inform how many times she had been involved in a course or project where the lecture or professor has participated in the teaching.

The result is collection of votes on 14 teachers by 79 students. In this section I will rank this data in order to find the students favourite teacher.

[8]Computer Science Department at the University of Copenhagen

I will start by ranking the data by transforming it into an Image Ranking Problem. The instance will consist of 14 images (on for each teacher) and 80 users (1 for each student and 1 to image parent).

In order to simplify the comparisons I will only compare the top five of the final ranking relations. To ease reading I will use abbreviations of the names. The full list of names, including their abbreviations, can be found in appendix A.1.

This ranking gave the folling relation when using the AP method with normalization, the PR method with normalization, the CR method with the modification suggested in section 5.1.1 and finally the CR′method without the modification:

AP	PR	CR′	CR
DP	DP	DP	DP
MZ	MZ	JGS	MZ
JKS	NA	MZ	JKS
NA	JGS	NA	NA
FH	FH	FH	FH

The rankings are very similar in that every method has chosen to build the top five from the teachers $\{\mathtt{DP}, \mathtt{MZ}, \mathtt{JGS}, \mathtt{NA}, \mathtt{FH}\}$. Another similarity is, that each method has chosen DP as number one, and FH as number 5. One difference that is interesting is that CR' has chosen JGS as number two, while CR has chosen MZ. The small modificiation in the objective function has swapped number two with number three. A significant difference.

The following table shows the number of pairwise preferrence reverses:

AP	PR	CR′	CR
3375	3386	3379	3376

The three methods are again very simlilar. Only the PR method deviates with a higher number of reverses. The CR and CR′are only 3 reverses apart, yet these 3 reverses are spared by swapping two of the highest ranked "images".

But who was in fact the best teacher? Who was most liked by the students? Well, in this ranking, students who has not even met the teacher they are voting on would count as much as the student who had had several courses with this teacher. However, I did collect information about the number of courses each participant had had with each teacher. I will now use this information to define the belief factor of each vote in the CR and CR′method. I define the belief factor of each vote as the number of courses taken by the participant and teached by the teacher divided by 25 (the maximum):

$$ScV^{v^{u,p}} = \frac{C^{u,t}}{25}$$

where u is the participating user, t is the teacher and $v^{u,p}$ is the vote by u on p (e.g. if user u_0 votes on t_0 and has had 4 courses related to this teacher the belief of this vote would be $\frac{4}{25}$).

In order to determine the belief factor of a comparison we simply the belief factor of each teacher being compared.

The top five result of this CR ranking is:

CR′	CR
JGS	DP
DP	JGS
JSP	JSP
NA	JSA
JSA	NA

These ranking relations are quite different from the previous relations. The teachers {JSP, JSA} is now part of the top five instead of {FH, MZ}. An interesting observation is that the modification in the objective function has placed DP as number one instead of JGS.

The conclusion on this ranking is, that the students favourite teacher is DP closely followed by JGS. However everyone who made it to the top five is of course well liked. The full ranking relation can be found in appendix A.2

9 Future Applications

The following two case studies is not test cases but rather a look at future applications for the Group Ranking methods discussed. They do not include any ranking at this stage but could very well lead to interesting appliances in the near future.

9.1 The AES Selection Process

In 1997 the National Institute of Standards and Technology of the USA (NIST) launched a series of conferences in order to select an encryption algorithm as the Advanced Encryption Standard (NIST, 2000).

More than 20 years earlier, NBS (now known as NIST) selected the Data Encryption Standard (DES). However, improvements in technology and computation power demanded a more secure standard.

In the year 2000, NIST selected the block cipher Rijndael as the AES. Rijndael was selected from a pool of five finalists and the selection was based on massive amounts of cryptanalysis, comments from cryptoanalytics and finally votes from the AES conference attendees.

In this section I will take a look at the voting process described in (NIST, 2000) and show how the Close Rankings method could be used in such a voting process.

At the final conference there was 246 attendees from which 167 voted. The voting process worked by having each attendee fill out a form which including the following questions:

1. If NIST selects one (1) algorithms for the standard, which one should it be?
2. If NIST selects two (2) algorithms for the standard, which two should it be?
3. If NIST selects three (3) algorithms for the standard, which three should it be?

4. If NIST selects four (4) algorithms for the standard, which four should it be?

The idea was to get a good picture of each attendees beliefs, hence to establish a preferred ranking relation for each attendee.

To simplify this section, as it is just a case study, I will only look at the results from the first question and discuss how the Close Rankings method could have been applied.

The vote count of the first question can be seen in the following table:

Algorithm	Votes
MARS	13
RC6	23
Rijndael	86
Serpent	59
Twofish	31

According to this vote count Rijndael has won the compitition. However, if each attendee had provided a full preferred ranking relation the scores might have looked different.

Lets say that everyone who did not choose Twofish as number one chose it as number two and that everyone who did not choose Rijndael as number one chose it number five. This would result in 126 preferred ranking relations with Rijndael as number five, 181 ranking relations with Twofish as number two and 31 ranking relations with Twofish as number one. This might have resulted in Twofish winning over Rijndael. I will not rank this data to check whether Twofish would actually win, because the data is just imaginary and contradicts the actual results presented in (NIST, 2000).

It does however prove my point. When ranking something as important as the AES candidates, one should not rely on simple ranking methods, such as human evaluation of votes. Also, when there are just five candidates it is possible to ask each attendee for a preferred ranking relation. The final ranking relation can then be computed by a group ranking method that respects the preferred ranking relation of each attendee, such as the Close Rankings method. One could even adjust the belief factor according to the amount of experience in cryptanalasis each attendee has.

To be fair, the AES competition ended 5 years before the publication of the Close Rankings method. However one can hope that newer group ranking methods will be used in the upcomming SHA-3 competition (NIST, 2007).

9.2 The Netflix Competition

Netflix[9] is a multinational online DVD rental service. In October 2006 they launched an interesting competition which invited everyone with an interest in statistics to try and solve one of Netflix's computation hard problems: find out what the users want[10].

All participants are given a rather large dataset listing users together with their votes on movies. The dataset includes 480189 users and 17770 movies. The dataset also includes

[9] www.netflix.com
[10] www.netflixprize.com

a listing of user votes without weights. The problem is now to assign the correct weights to these movies, hence to predict future user ratings based on their previous ratings. In order to win the main prize of 1 million dollars, the result has to be more than 10% better than Netflix own system, which is currently a straightforward statistical linear model.

I believe that the Close Rankings model could be used for this particular problem. The idea is to compute the personal ranking relation for each user based on the movies this user has rated. Let such a user be U_0.

This ranking relation can now be expanded with a movie that U_0 has not yet rated. Because U_0 has not rated the new movie, there will be no connection between this movie and those in the original relation.

This problem can be solved by using the preferred ranking relations of the other users. Simply find the set of users who has rated both the newly added movie and some other movie rated by U_0 and add them to the system.

When adding the preferred pairwise rankings of a new user we will only need to add those preferred pairwise rankings that include both the new movie we want to predict the ranking of and some other movie that U_0 has ranked. The confidence of this particular pairwise preferrence could be defined by the difference between the pairwise rankings U_0 has chosen for the shared movie and those the new user has chosen.

In fact, this idea just uses the Close Rankings method together with transitivity to predict the users ranking of the new movie.

In short the method for predicting U_i's ranking of movie M_j could be described as follows:

1. compute U_i's ranking relation of all movies U_i has rated
2. add M_j to the problem instance
3. add preferred pairwise rankings for all users, $u \in U \backslash \{i\}$, who has voted on movie M_j and on at least one movie that U_i has voted on
4. set the confidence of each new pairwise ranking according to the difference between U_i's preferred pairwise rankings and the new users' preferred pairwise rankings
5. set the confidence of pairwise rankings by user U_i to the highest possible (or simplify the problem by fixing them)

The only problem with this method is the humongous amount of computation power needed. Consider a user who has voted on each of the 17770 movies. The object function of the Close Rankings problem would now contain:

$$\sum_{i=1}^{17769} i = \frac{17769 * (17769 - 1)}{2} = 157859796$$

terms. It might be possible to reduce this problem to a simpler one, but remember that this is just the problem of computing the first ranking relation. After this is the problem of expanding this instance with one more movie and who knows how many users and pairwise rankings.

However i do believe that if anyone solves these computational problems they will stand a good chance of bringing home the main prize.

10 Conclusion

I have given a formalization of the Image Ranking Problem and discussed the problems that might arise when trying to solve this problem fairly.

When analysing the Average-Pont method, I discovered that this method lacks normalization and that it does not prevent the problem of Newcomer's Rush. However it was possible to apply the generic workarounds for both problems.

I managed to convert a weight-only group ranking problem into a graph problem that could be solved by the PageRank method. During my analysis of this method I found a simple proof that the computations associated with this method do in fact terminate. As a result I could formulate another proof that some of the lines of the pseudo code presented in (Page, 1999) could be skipped safely.

During my study of the Close Rankings method, I discovered a flaw in the way it handles pairwise comparisons. I have given a simple modification that relaxes this flaw.

I have shown that all three methods discussed in this project rank on different backgrounds. The simpliest is the Average-Point method that only uses the weights to rank the images. The PageRank method expands this by letting weights placed by users, who themselves have highly weighted images, count more. Finally the Close Rankings method generalize this concept by allowing a confidence factor to be placed on each comparison of two votes.

I have discussed future appliances where it would be interesting to see the Close Rankings method at work. I have discussed how this method could be used in competitions like the AES selection process. I have also discussed how this method might be used to win the Netflix competition and given an example of how the problem for this competition could be formulated.

With respect to future research it would be interesting to see how the close rankings method would perform in a true web environment. Perhaps to see an implementation that would allow web developers to use this method for ranking without too much hazzle. This would include efficient implementation of the conversion from weights to intensities together with extraction of data from popular database management systems (DBMS).

Also, it would be interesting to investigate the possibilities of the PageRank method. Specifically the possibility of implementing custom user rankings. This might be possible with the use of a big brother node. Each image could then share part of its flow with the big brother node, and afterwards using the big brother node to redistribute this flow in a prioritized manner.

Finally I can conclude that the the three methods rank very similarly on simple systems. There are differences, but these might not be significant enough to compensate for performance issues when comparing with the speed of the Average Point method. However if a more flexible ranking method is needed, the Close Rankings method is recommendable. The confidence factors on each pairwise comparison make this method the most flexible of them all. I would say that the Close Rankings method is the answer to my original Problem Specification. It certainly was possible to solve the Image Ranking Problem while respecting the voting users' rankings together with the relevance of each vote and the trend amongst all users.

11 References

Ali, I., Cook, W. D. & Kress, M. (1986)
Ordinal ranking and intensity of preference: a linear programming approach
Management Science, 32:12, 1642-1647

Boyd, S., Mutapcic, A., Xiao, L. (2003)
Subgradient Methods
Notes for EE392o, Stanford University

Chandran, B., Golden, B. & Wasil, E. (2005)
Linear programming models for estimating weights in the analytic hierarchy process
Computers and Operations Research, 32:9, 2235-2254

Heath, M. T. (2002)
Scientific Computing, An Introductory Survey, Second Edition
University of Illinois, McGraw-Hill, ISBN: 0-07-239910-4

Hochbaum, D. S. & Levin, A. (2005)
Methodologies and Algorithms for Group-Rankings Decision
Management Science, 52-9 (September 2006), 1394-1408, ISSN: 0025-1909

Keener, J. P. (1993)
The Perron-Frobenius theorem and the rating of football teams
SIAM review, 35:1, 80-93

Leon, S. J., (2006)
Linear Algebra with Applications
Pearson Prentice Hall; 7 edition (2006), ISBN: 0-13-200306-6

National Institute for Standards and Technology (NIST) of the USA (2000a)
Report on the Development of the Advanced Encryption Standard (AES)
http://csrc.nist.gov/archive/aes/round2/r2report.pdf

National Institute for Standards and Technology (NIST) of the USA (2000b)
AES3 Evaluation Feedback Summary
http://csrc.nist.gov/archive/aes/round2/conf3/AES3FeedbackForm-summary.pdf

National Institute for Standards and Technology (NIST) of the USA (2007)
Federal Register Vol. 72, No. 212 / Friday, November 2, 2007 / Notices
http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf

Page, L., Brin, S., Motwani, R. & Winograd, T. (1999)
The PageRank Citation Ranking: Bringing Order to the Web
Stanford University

Saaty, T. (1977)
The Analytic Hierarchy Process
McGraw-Hill, New York

Saaty T., Vargas L. (1984)
Comparison of eigenvalue, logarithmic least squares and least squares methods in estimating ratios
Math. Modeling 5 309-324

A Real Life example

A.1 Teacher list

This is the complete list of professors and lectors with their abbreviations:

GS	Georg Strøm
DP	David Pisinger
BV	Brian Vinter
FH	Fritz Henglein
JSA	Jørgen Sand
JGS	Jakob Grue Simonsen
JSP	Jon Sporring
EJ	Eric Jul
MZ	Martin Zachariasen
AF	Andrzej Filinski
JJK	Jyrki Juhani Katajainen
PB	Philippe Bonnet
RG	Robert Glück
NA	Nils Andersen

A.2 Full ranking relation

The following table shows the full ranking relation computed using the CR method:

1	DP
2	JGS
3	JSP
4	JSA
5	NA
6	BV
7	FH
8	AF
9	RG
10	MZ
11	JJK
12	GS
13	EJ
14	PB