An Iterative Rating Method: Application to Web-based Conference Management

Philippe Rigaux
LRI, Université Paris-Sud, France
rigaux@lri.fr

Abstract

Given a large set of items and a set of users, we consider the problem of collecting user preferences – or ratings – on items. The paper describes a simple method which provides an approximate solution to the problem without requiring each user to rate each item. The method relies on an iterative process. Each step, or ballot, requires each user to rate a sample of the items. A collaborative filtering algorithm is then performed to predict the missing ratings as well as their level of confidence (which is initially 0). Performing a new ballot allows to improve the accuracy of predictions. The administrator of the system is responsible for stopping the iteration when a satisfactory level is reached.

We apply this method to the assignment of reviewers to papers prior to the review phase of conference management, and describe its implementation in the MyReview web-based system.

1 Introduction

Many scientific conferences nowadays use during the submission phase a web software for collecting papers and assigning reviewers, (see, for instance, [4, 3, 2, 5]). These softwares also provide an interface to submit reviews and supply some more or less advanced support for discussion among reviewers when there is no agreement on a paper.

It is commonly agreed that the most time-consuming task when using this kind of software is the assignment of reviewers to papers. The basic solution is a manual process where the administrator (usually the program chair of the conference) selects, for each paper, 3 or 4 members of the program committee (PC). However, performing this task manually is both tedious and difficult, because of the many constraints which must be taken into account: all reviewers should be assigned approximately the same number of papers, each paper should be evaluated by expert reviewers, there should be no conflict of interest, etc. The size of the problem is also an issue: there might be up to several hundreds of submitted papers, and the PC commonly consists of about 40-50 members.

There is currently little support for this task in the existing softwares. At best (for instance in CyberChair [5]), reviewers can select the papers they wish to review and a pre-assignment, based on these preferences, is computed and proposed to the
administrator. This proposition is unfortunately quite partially useful, mostly because information provided by the reviewers is sparse (many papers are ignored) and poor.

The assignment of papers can actually be seen as a specific instance of a more general problem, namely the recommendation of items to users in web-based systems. Recommender systems [13] provide personalized recommendations for products or services to users, and their techniques are intensively used in e-Commerce systems to dynamically propose items based on the user’s taste and expectations [1, 8, 14].

Recommender systems rely on knowledge discovery techniques to collect and predict preferences of users. Among these techniques, collaborative filtering algorithms can be used to find neighbors of a user, i.e., users who share the same preferences. Clearly, applying these techniques to our problem, considering PC members as “users” and papers as “items”, can facilitate the assignment of papers, and improve its reliability. There are however some particularities of our problem that deserve to be mentioned. On the negative side, the number of members in a PC is small with respect to the number of users which visit an e-commerce site, and this limits the ability to find close neighbors during the collaborative filtering process. On the positive side, whereas recommender systems in e-commerce applications dispose only of implicit and incomplete data to discover the profile of users, a program committee consists of a small set of cooperative and active “users”, who can be asked for explicit and reliable information regarding their preferences.

We describe in this paper a recommendation method based on collaborating filtering and tailored to the above specificities – a small group of cooperative users. The method allows to compute a predicted rating for each pair (PC member, paper), using a multi-step process which improves continuously the confidence level of the ratings. In short, each step, or ballot, consists of the following operations: (1) for each user, a sample of papers whose rating is expected to lead to the best confidence level improvement is selected, (2) each user is requested to rate the papers from its sample and (3) a collaborative filtering algorithm is performed to obtain a new set of predicted ratings, based on the users ratings made so far. Step 3 results in a new level of confidence. Depending on the context, the administrator can then take a new ballot, or skip to the paper assignment task.

The method has been implemented in a web-based conference management system, freely available for use, which is briefly described at the end of the paper. Note that, although we describe the method and its implementation with respect to a specific application, it might be relevant in a broader context. A significant problem with recommendation systems is the sparsity of information during the initial stages of use, and thus the lack of precision of the predictions. The method proposed here allows to create a “seed” for the prediction algorithms, by considering a sample of users, representative of the many profiles which can be met on the Web. Ratings can be collected, up to the confidence level required to provide a sound support for the recommendation module. The sample constitutes then a reliable basis for determining the preferences of users.

In the rest of this paper we present first an overview of the most popular web-based conference management systems (Section 2) and discuss the assignment problem. We provide then (Section 3) some background on the techniques used in the MYREVIEW system. The current implementation is described in Section 4. Section 5 concludes the paper and outlines future work.
2 The paper assignment problem

The presentation that follows is based on investigations of some popular systems: CyberChair [5], BYU [2], ConfMan [4] and the Microsoft Conference Management Toolkit (CMT) [3]. A list (partly out-of-date) of other softwares, often developed ad hoc for specific events, can be found at \url{http://www.acm.org/sigs/sgb/summary.html} In the following we restrict the discussion to the assignment of papers to reviewers.

In all systems, the administrator can establish, before the submission phase of the conference, a list of research topics, and ask the reviewers to choose the topics they are interested in. This list of topics, when it exists, is also proposed to the authors in the paper submission interface. The topic information provides a first basis for the assignment process, and a simple approach consists in building an assignment proposal by matching reviewers and papers topics.

It is however very difficult to obtain a satisfactory result using this method. A first problem is that matching a reviewer’s topic with a paper’s topic does not actually guarantee that the reviewer is the best expert for the paper. Second the distribution of reviewers and papers over the list of topics is unpredictable, and some papers will find a lot of experts willing to review them, while some others will simply not have enough reviewers. Finally the information provided by topics is “binary”: given a pair (reviewer, paper), the paper topic matches one of the reviewer’s topics, or it does not. This gives no clues on how to handle the assignment when all the possible topic-based matchings have been considered.

As an improvement, several systems allow the reviewers to browse the list of abstracts (which must usually be submitted about one week before full papers) and to express explicitly their preferences. This solves the first problem mentioned above, because the explicit selection of a paper is much more reliable that the topic-based matching. However the two others problems still prevent a fully relevant automatic assignment. Because the information is sparse – and because many reviewers will tend to choose the papers they eagerly want to review – the choice for papers which have not been enough selected cannot be safely driven by an automatic process.

The systems (among the four that we consider) which seem the most advanced with respect to the assignment functionality are CyberChair and CMT. CMT uses a greedy algorithm, assigning a paper to the reviewers who gave the higher preference, but limiting the number of papers assigned to a reviewer to a threshold. When the system cannot find a reviewer, a matching of both the reviewers and paper topics is used. If this fails the result is unpredictable. The heuristic used by CyberChair, described in [6], is a bit more involved. Still, the proposition obtained with these algorithms is often reported to be quite partially useful, and sometimes misleading.

We believe that the rich set of information collected to describe both the papers content and the reviewers preferences could be used in a much more effective way. The method proposed in the current paper relies both on topics matching and reviewers preferences. However, in addition to serve as a basis for paper assignment, the way this information is collected permits an (as much as possible) accurate prediction of preferences of reviewers on the papers that they did not explicitly rated. The level of confidence on these predictions is provided to the administrator who can decide – depending both on his/her expectations and on the degree of cooperation of the program committee – whether a new set of reviewers preferences must be collected.
Table 1: Table of symbols used in the paper

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
<td>The set of users</td>
</tr>
<tr>
<td>$n$</td>
<td>The number of users, $</td>
</tr>
<tr>
<td>$u, u_1, \ldots u_n$</td>
<td>Users</td>
</tr>
<tr>
<td>$P$</td>
<td>The set of papers</td>
</tr>
<tr>
<td>$m$</td>
<td>The number of papers, $</td>
</tr>
<tr>
<td>$p, p_1, \ldots p_m$</td>
<td>Papers</td>
</tr>
<tr>
<td>$S$</td>
<td>Size of the sample rated by a user</td>
</tr>
<tr>
<td>$K_{u_1, u_2}$</td>
<td>Nb. of papers co-rated by $u_1$ and $u_2$</td>
</tr>
<tr>
<td>$r_{u, p}$</td>
<td>Rating of the user $u$ for the paper $p$</td>
</tr>
</tbody>
</table>

When the level of confidence reaches a satisfactory level, the automatic assignment algorithm can be run. Then, unlike the traditional methods, this algorithm knows the level of preference of each user with respect to each paper, and can compute, given this information, the best possible assignment. Some manual, hopefully limited, modifications can then be introduced by the administrator to complete the task.

3 Recommandation techniques

We first provide a short introduction to collaborative filtering and discuss its applicability to our specific problem. All the symbols used throughout the paper are summarized in Table 1.

Collaborative filtering

We consider a set $U$ of users and a set $P$ of papers. Each user can rate a paper to express his willingness/expertise to review this paper. In the following a user rating is a number ranging from 0 (“I do not want this paper”) to 4 (“I am eager to review this paper”). This can be represented as a table, as follows.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bill</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td>1</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rachel</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>George</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the best case, each user provides a rating for each paper. The assignment of papers can then be solved optimally (and automatically) using advanced algorithms. Our system relies on a variant of known techniques for optimal weighted matching in bipartite graphs [12]. This algorithm delivers the best possible assignment (we shall provide some details and a short description in Section 4). However, in practice, the number of papers is often large and it is difficult to ask for a comprehensive rating. Users rate only a small subset of the papers, and the rating table is sparse, with many
unknown rating values. In order to use the automatic assignment algorithm, we must then “predict” the missing rating values.

The basic idea of collaborative filtering is to match together users who share the same taste [11, 16, 7]. In our context, if two users provided almost the same ratings to the papers they have in common, one can assume that this will remain true for the other papers. For instance Bill and Rachel provided similar ratings for papers 1, 3 and 5. Since Bill’s rating is 2 for paper 4, it can be predicted that the rating of Rachel will be close to 2 as well. Bill and Rachel are said to be neighbors. Essentially, collaborative filtering techniques consist in determining as accurately as possible the neighbors of each user, in order to derive predictions from the ratings of these neighbors.

The effectiveness of these techniques depends on many factors [10], and in particular on the numbers of neighbors which can be associated to a given user, as well as on the significance of the neighborhood relationship. To illustrate these concepts, consider, in the example above, the case of Alice. There is only one other user with common ratings (George), and both users have only one co-rated paper. Clearly one cannot obtain accurate predictions in such a situation.

Our implementation relies on the classical formulas first proposed by the GroupLens project [11] (see also the site http://www.cs.umn.edu/Research/GroupLens/). The correlation \( c_{u_1,u_2} \) between two users \( u_1 \) and \( u_2 \) is the Pearson correlation coefficient, determined over the papers for which both \( u_1 \) and \( u_2 \) have provided ratings.

\[
c_{u_1,u_2} = \frac{\sum_j (r_{u_1,j} - \bar{r}_{u_1})(r_{u_2,j} - \bar{r}_{u_2})}{\sqrt{\sum_j (r_{u_1,j} - \bar{r}_{u_1})^2 \sum_j (r_{u_2,j} - \bar{r}_{u_2})^2}}
\]  

where \( j \) ranges over the set of papers, \( \bar{r}_{u_1} \) is the average rating for user \( u_1 \) and \( r_{u_1,j} \) denotes the rating of user \( u_1 \) on paper \( j \). One can verify that if two users provide exactly the same ratings, the correlation is 1. The predicted rating of a user \( u \) for a paper \( p \) is an average of the rates of the other users who rated \( p \), weighted by the correlation between \( u \) and each user.

\[
r_{u,p} = \bar{r}_u + \frac{1}{C} \sum_{i=1}^n c_{u,i}(r_{i,p} - \bar{r}_i)
\]  

where \( C \) is the sum of correlations, used to normalize the result, and \( i \) ranges over the \( n \) users that constitute the “neighborhood” of \( u \), to be discussed next.

**Estimating the accuracy of predictions**

As mentioned previously, the accuracy of the result depends on several factors which are now discussed more precisely. Consider first equation 1. The confidence in the correlation between two users depends on the number of co-rated papers. When the computation is based on a small sample, the correlation value is not really significant. The significance weighting indicator, proposed in [7], is \( sw_{u_1,u_2} = \min(1, \frac{c}{K}) \), where \( c \) is the number of papers co-rated by \( u_1 \) and \( u_2 \), and \( K \) is the minimal number of co-rated papers for which the correlation is considered as fully significant.

A second important factor is the “neighborhood” of a user \( u \), in other words the subset of \( U \) which is used to predict a rating for \( u \) in equation 2. The issue is discussed in [15] which proposes several approaches, among which the “best-n” seems to yield
the best result. It simply consists in taking the $n$ users having the largest correlation with $u$.

Both the significance weights and the neighborhood selection can be seen as improvements over the basic formulas 1 and 2. They also provide a mean to estimate the accuracy of the prediction related to the user $u$ and the paper $p$. Since our problem is characterized by the small number of users, we use a confidence level indicator $cl(r_{u,p})$ which is the average of the correlations with the users involved in the computation of $r_{u,p}$ weighted by their significance.

$$cl(r_{u,p}) = \frac{\sum_{i=1}^{n} s w_{u,u_i} c_{u,u_i}}{\sum_{i=1}^{n} s w_{u,u_i}}$$

(3)

The symbols meaning is similar to that of Equation 2.

**Application to conference management**

Let us now turn our attention to web-based conference management. As pointed out in the introduction, an important difference with the traditional context (for instance e-commerce applications) is the small number of users/reviewers. This limits severely the number of co-rated items. Indeed, let $m$ be the number of papers and $S$ the size of the sample explicitly rated by each user (we assume that all users rate the same number of papers). The probability $Pr(u, p)$ for a user $u$ to rate a paper $p$ is $\frac{n}{m}$. Given two users $u_1$ and $u_2$ and a paper $p$, the probability $Pr(u_1, u_2, p)$ that both users rate $p$ is $Pr(u_1, p) \times Pr(u_2, p) = \frac{n^2}{m^2}$, if we assume that the ratings are collected independently. Finally, given two users $u_1$ and $u_2$, the number $K_{u_1,u_2}$ of co-rated papers is the statistical expectation $E_{S_1,S_2}(|S_1 \cap S_2|)$ of the cardinality of $S_1 \cap S_2$, where $S_1$ is the sample rated by $u_1$ and $S_2$ is the sample rated by $u_2$. By definition we have:

$$E_{S_1,S_2}(|S_1 \cap S_2|) \overset{def}{=} \sum_{S_1,S_2} |S_1 \cap S_2| Pr(S_1, S_2)$$

which is equal to $\sum_{S_1,S_2} X(p, S_1, S_2) Pr(S_1, S_2)$ where $X(p, S_1, S_2)$ is 1 if $p \in S_1 \cap S_2$ and 0 else. If we assume that $S_1$ and $S_2$ are chosen randomly, this is equal to $\sum_p (\sum_{S_1,S_2} X(p, S_1, S_2) Pr(S_1, S_2)) = \sum_p (Pr(u_1, u_2, p)) = m \times \frac{n^2}{m^2}$. The number of co-rated items, under the previous assumptions, is $\frac{n^2}{m}$, proportional to the square of the sample size, and inversely proportional to the number of papers. Let for instance $m = 100$ and $S = 20$. The probability $Pr(u, p)$ is 1/5, and the probability $Pr(u_1, u_2, p)$ is $1/5 \times 1/5 = 1/25$. The number $C_{u_1,u_2}$ of papers co-rated by $u_1$ and $u_2$ is expected to be $100/25 \approx 4$.

In order to improve $K_{u_1,u_2}$ we can rely on one (or both) of the following approaches:

1. **Raise the size $S$ of the sample.** Note that in the best case, $S = m$ and $K_{u_1,u_2} = \frac{m^2}{m} = m$: all papers are co-rated.

2. **Restrict the set of papers possibly rated by a user.** This can be achieved by using a partitioning strategy to group users in clusters, and by assigning each cluster to a subset of papers. Given two users $u_1$ and $u_2$ in the same cluster $CL$, $K_{u_1,u_2}$ is now inversely proportional to the size of the subset associated to $CL$ $<< m$. 
We can indeed take advantage of the fact we have a small, but cooperative set of users to improve the quality of the predictions. First, clustering can be achieved by exploiting the research topics of the conference. The system can group reviewers in clusters, using their selected topics and asking each reviewer to provide a rating for the papers associated to its clusters (i.e., papers whose topic matches the topics of the cluster). As a result, the amount of co-rated papers is likely to be improved significantly since two users who declared the same topic interest will naturally tend to rate the same papers.

The second improvement consists in raising the size of the sample of papers rated by the reviewers. Choosing a good sample size depends however on several conference-specific factors, in particular the number of submitted papers (it seems not reasonable to ask for rating of 600 papers) and the degree of willingness of the program committee to cooperate to the process. A flexible approach consists in using an iterative process which augments gradually the size of the sample at each step. This leads to the following iterative rating method:

**Algorithm** ITERATIVE-RATING

**Input:** \( U, P \), user ratings for a set of pairs \( R \in U \times P \)

**Output:** predicted ratings for the pairs in \( U \times P \setminus R \)

1. Partition the set of users in clusters, and associate a set of papers to each cluster.
2. (Initial ballot). Create a rating box for each user, and put in this box the papers of his cluster.
3. Ask each user to rate the papers in his rating box. Remove papers from the box when they are rated.
4. Compute the correlations (Formula 1), the predictions (Formula 2) and the confidence level (Formula 3) of the predictions (formula 3).
5. If the confidence level is not satisfactory, take a new ballot as follows. For each user \( u \), select a new sample of papers whose rating by \( u \) will best improve the confidence level. Put them in the rating box of \( u \). Go to step 3.

Several details, in particular the creation of clusters and the selection of new samples, are left to the implementation and will be described in the next section. It is easily seen that each iteration in this algorithm allows to collect more ratings, and therefore improves the level of confidence. In the simplest case, only one ballot is taken, and the reviewers are required to rate the papers that correspond to their preferred topics. This is the traditional situation. In the extreme case, all possible iterations are run, and each user explicitly rates each paper (this strategy is used in some conferences), which results in an optimal confidence level regarding the reviewer preferences.

### 4 Assignment in MyReview

**MyReview** is a web-based conference management software which has been implemented in may-july 2003, and used initially for managing the ACM conference
on Geographic Information Systems\textsuperscript{1}. The main goals of the system are to provide and easy-to-install and easy-to-manage software, based on the most up-to-date and widespread technologies. It proposes the traditional functionalities of such systems, namely paper submission, reviewer assignment, discussion on conflicting reviews, selection of papers, mailing to all actors, etc.

From the beginning, a particular effort has also been devoted to address the identified weaknesses of other, popular, systems, particularly the presentation/logic independence, and the support for assignment of papers to reviewers. In the following we focus on the latter aspect, but the interested reader is referred to the demo site for a larger overview of the system.

\url{http://myreview.lri.fr}

The prediction module is an optional part of the system. Whether the administrator chooses or not to use the module, it is always possible to modify “manually” the assignment through a web interface. This option is probably valid for small conferences, but when the administrator has to deal with a lot of papers, it should be completed with the tools supporting the assignment process.

![Figure 1: The rating box of a user](image)

\textbf{The automatic assignment module}

Let us first outline the automatic assignment function. It is based on an optimal weighted matching algorithm (WMA) for bipartite graphs \([12, 9]\) that delivers the best possible assignment. More precisely, the WMA applies to a bipartite graph \(G = (V, E)\), with \(V = U \cup W\). Every edge in \(G\) is of the form \(\{u_i, w_j\}\) where \(u_i \in U\) and \(w_j \in W\). Further, its is assumed that \(G\) is \textit{complete} (an edge exists for each pair \((u_i, w_j)\)), and \textit{weighted}, i.e., we are given a number \(w_{ij} \geq 0\) for each edge \((u_i, w_j)\). A \textit{matching} \(M\) of \(G\) is a subset of the edges with the property that no two edges of \(M\) share the same node. The \textit{weight} of \(M\), denoted \(wt(M)\), is \(\sum_{e \in M} wt(e)\).

\textsuperscript{1}See \url{http://www.esri.com/events/acm/index.htm}.
A (weighted) matching $M_o$ of $G$ is optimal if every other matching $M$ of $G$ is such that $wt(M_o) \geq wt(M)$.

In our application, $U$ is the set of papers, $W$ the set of reviewers, and the ratings represent the weights. The WMA computes a matching $M$ which assigns one paper to each reviewer, so as to optimize the sum of the ratings in $M$. This is illustrated in Figure 2 which shows the graph $G$ together with the rating/weight on each edge. The matching $M$ is represented by thick lines: it can be verified that its weight is $wt(M) = 3 + 4 + 1 = 8$, and that any other matching yields a lower value. Note that, if no ones like a paper (for instance Paper 2), it will get reviewers with low ratings but this is unavoidable.

![Figure 2: The weighted matching algorithm](image)

The assignment module performs $k$ steps, where $k$ is the number of reviewers to assign to each paper (the default is 3). Each step runs the WMA and results in the assignment of a new reviewer to each paper. When a reviewer has been selected during a step, he is no longer considered as a candidate for the papers that remain to be assigned.

Since the automatic assignment can only be used if the graph is complete, we must provide a value for missing ratings, as described below.

**Initial rating of papers**

When all the papers abstracts have been submitted, the administrator can run a script which determines, for each user, the subset of the papers for which the user must enter a rating. These papers are inserted in the rating box of this user. The size $S$ of this subset is a parameter which can be set by the administrator (its default value is 20). Each reviewer can then access a password-protected form displaying the papers of his rating box, along with a pre-defined list of preferences, ranging from 0 (I do not want this paper) to 4 (I am eager to review this paper). An example of this form is displayed in Figure 1 (the presentation of the screen shots is based on the ACM GIS conference graphical design, which can be customized easily; papers and reviewers are of course fictive).

During the initial ballot, the selection of the papers inserted in the rating box of a user $u$ is based on the research topics. All the papers whose topic matches one of the topics of $u$ are selected. If there are more than $S$ (the size of the rating box) papers, then only $S$ papers, chosen at random, are kept, else the rating box is completed with other papers, randomly selected. Reviewers that share the same topics interest will tend to get the same papers in their initial rating box and this increases the quality of correlations, estimated by the number of co-rated papers.
All the ratings which are explicitly provided by reviewers are stored with the highest confidence level (i.e., 1) and are never updated. Predicted rating will be stored with a confidence level which is usually lower than 1, and are replaced each time the prediction algorithm is run.

**Computing predictions**

Once all the papers in the rating boxes have been rated by the reviewers, the administrator can run the predictive algorithm which stores the following results in the database:

- The correlation for each pair of reviewers, along with the number of co-rated papers.
- The predicted rating for each pair (reviewer, paper), along with the confidence level.

A web page, reserved to the administrator, shows a list of the papers with both the user ratings and the predicted ratings (see Figure 3). For each paper, the candidate reviewers and their preferences are listed, sorted by their rating in descending order. Two threshold are considered, one for the rating, and another for the confidence level. When a value is below one of its associated threshold, the reviewer is no kept. When two ratings are equal, those with the higher confidence level are presented first. Note that this constitutes only a suggestion of the choices that would be made by the automatic assignment. The administrator is free, at this point, to ask the reviewers for other ratings, or to use the automatic assignment function which can be run independently. He can also choose to create manually the assignment, using the suggested order to help the selection of reviewers. In all cases the rating (explicit or predicted), along with its level of confidence, is proposed as a hint to select reviewers.

**Additional ballots**

When the suggested assignment is considered as not satisfactory by the administrator because of low confidence levels, a new ballot of rating can be taken. The ballot
results in a new sample of paper inserted in the rating box of users, and initializes the same workflow: reviewers must rate the new papers in their rating box, a new (more accurate) set of predictions can be generated, etc. The difference with the initial constitution of the rating box lies in the selection of papers. The goal, during the additional ballots, is to upgrade the confidence level of the predictions which can possibly lead to the assignment of a paper to the reviewer. Therefore, for each user, the papers whose predicted rating is high and the confidence level is low are selected and inserted in the rating box, up to the size $S$ of the sample.

5 Concluding remarks

We described in this paper a method to generate predicted preferences of reviewers during the submission phase of conference management. The method is based on the traditional information used in web-based systems (research topics and declared preferences) but goes further by providing indicators which help the assignment process, automatic or manual. We rely on predictive techniques which proved their effectiveness in modern web systems, and adapted these techniques to the specific features of the application.

The method has been fully implemented in the MYREVIEW system, ready to be tested. We already used the system for one conference. One ballot of rating were performed and we used the suggestions during a manual assignment. The number of papers submitted to the conference was relatively small (80) but for larger settings, using several ballots as well as the automatic assignment functionality can prove to be quite helpful. We hope that the flexibility of the approach will allow to use appropriately the functionalities in various contexts.

Acknowledgments

I am quite grateful to Miki Hermann who proposed and implemented the weighted matching algorithm.

References


