Supervised Rank Aggregation

Yu-Ting Liu[1,2*], Tie-Yan Liu[1], Tao Qin[1,3*], Zhi-Ming Ma[4], and Hang Li[1]

[1]Microsoft Research Asia
4F, Sigma Center, No. 49,
Zhichun Road, Haidian District,
Beijing, 100080, China
{tyliu, hangli}@microsoft.com

[2] School of Science,
Beijing Jiaotong University,
Beijing 100044, China
liuyt_njtu@hotmail.com

[3]Department of Electronic
Engineering,
Tsinghua University,
Beijing 100084, China
qinshitao99@mails.thu.edu.cn

[4]Academy of Math and
Systems Science, Chinese
Academy of Science,
Beijing 100080, China
mazm@amt.ac.cn

ABSTRACT

This paper is concerned with rank aggregation, the task of combining the ranking results of individual rankers at meta-search. Previously, rank aggregation was performed mainly by means of *unsupervised* learning. To further enhance ranking accuracies, we propose employing *supervised* learning to perform the task, using labeled data. We refer to the approach as 'Supervised Rank Aggregation'. We set up a general framework for conducting Supervised Rank Aggregation, in which learning is formalized an *optimization* which minimizes disagreements between ranking results and the labeled data. As case study, we focus on Markov Chain based rank aggregation in this paper. The optimization for Markov Chain based methods is not a convex optimization problem, however, and thus is hard to solve. We prove that we can transform the optimization problem into that of Semidefinite Programming and solve it efficiently. Experimental results on meta-searches show that Supervised Rank Aggregation can significantly outperform existing unsupervised methods.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *Retrieval models*. H.3.4 [Information Systems Application]: Systems and Software- *performance evaluation (efficiency and effectiveness)*.

General Terms

Algorithms, Experimentation, Theory

Keywords

Rank aggregation, supervised learning, Markov Chain, Semidefinite programming

1. INTRODUCTION

Rank aggregation is to combine ranking results of entities from multiple ranking functions in order to generate a *better* one. The individual ranking functions are referred to as base rankers, or simply rankers, hereafter.

Rank aggregation can be classified into two categories [2]. In the first category, the entities in individual ranking lists are assigned scores and the rank aggregation function is assumed to use the *scores* (denoted as score-based aggregation) [11][18][28]. In the second category, only the orders of the entities in individual ranking lists are used by the aggregation function (denoted as order-based aggregation). *We focus on order-based aggregation in this paper*. Order-based aggregation is employed at meta-search, for example, in which only order (rank) information from individual search engines is available.

Previously order-based aggregation was mainly addressed with the *unsupervised learning* approach, in the sense that no training data is utilized; methods like Borda Count [2][7][27], median rank aggregation [9], genetic algorithm [4], fuzzy logic based rank aggregation [1], Markov Chain based rank aggregation [7] and so on were proposed. One exception is Borda Fuse [2] which also makes use of training data. However, it is different from the supervised learning method we propose in this paper.

We argue that in order to improve the accuracy of rank aggregation, it is better to employ a *supervised learning* approach in which we train an *order-based* aggregation function within an *optimization framework* using *labeled data*. At meta search, for example, labeled data can be documents and their relevancies to given queries. The key factors, thus, are (a) to assume that only order information from individual rankers is available, (b) to use labeled data, and (c) to train the aggregation function within an optimization framework. In this paper, we refer to the approach as 'Supervised Rank Aggregation'.

There are several advantages for taking the supervised learning approach. First, we can leverage the use of information existing in labeled training data. Second, we can apply existing optimization techniques to the problem. Third, it becomes easier to make domain or user adaptation. Certainly, it also has a disadvantage, that is, labeled data is needed and creating such data can be costly. This is, however, a shortcoming for any supervised learning method and we can leave it as future research topic.

In this paper, we first give a general framework for conducting Supervised Rank Aggregation. We show that we can define supervised learning methods corresponding to the existing unsupervised methods, such as Borda Count and Markov Chain based methods by exploiting the framework.

Then we mainly investigate the supervised versions of Markov Chain based methods in this paper, because previous work shows that their unsupervised counterparts are superior [24]. It turns out, however, that the optimization problems for the Markov Chain based methods are hard, because they are not convex optimization problems. We are able to develop a method for the optimization of one Markov Chain based method, called Supervised MC_2. Specifically, we prove that we can transform the optimization problem into that of Semidefinite Programming. As a result, we can efficiently solve the issue. (We plan to apply the same technique to the other Markov Chain methods in the future.)

Experimental results on meta-searches show that Supervised Rank Aggregation (i.e., Supervised MC_2) can achieve better performances than existing methods.

[*]This work was conducted when the first and the third authors were interns at Microsoft Research Asia

The rest of this paper is organized as follows. In Section 2, we introduce related work. In Section 3, we propose a general framework and specific methods for Supervised Rank Aggregation. In Section 4, we propose an optimization algorithm for the method of Supervised MC_2. Experimental results are reported in Section 5. Conclusions and future work are given in the last section.

2. RELATED WORK

The origin of research on rank aggregation can be traced back to the eighteenth century, when it was studied in social choice theory and applied into political elections [5]. In recent years, rank aggregation gets spotlight again in many new applications, such as genome database construction [26], document filtering [13], database middleware construction [10], spam webpage detection [7], meta-search [2][7][17][24], word association finding [7], multiple search [11], and similarity search [9].

There are two types of rank aggregation: score-based and order-based. In the former the aggregation function takes score information from the individual base rankers as input, while in the latter it only utilizes order information. Order-based aggregation fits well with meta-search, as in meta-search only order information from base rankers is available; this is also the main focus of the research in this paper.

Existing methods for order-based aggregation includes, for example, Borda Count [2][7][27], median rank aggregation [9], genetic algorithm [4], fuzzy logic based rank aggregation method [1] and Markov Chain based rank aggregation [7]. Borda Count ranks entities based on their positions in the ranking lists. For example, the entities are sorted according to the number of entities that are ranked below them in all the ranking lists. Median rank aggregation sorts the entities based on the medians of their ranks in all the ranking lists. Markov Chain based rank aggregation assumes that there exists a Markov Chain on the entities and the order relations between entities in the ranking lists represents the transitions in Markov Chain. The stationary distribution of the Markov Chain is utilized to rank the entities. Dwork *et al* [7] proposed four methods (denoted as MC_1, MC_2, MC_3, and MC_4) to construct the transition probability matrix of the Markov Chain.

The unsupervised methods described above implicitly conduct majority voting in their final ranking decisions. That is to say, these methods treat all the ranking lists equally and give high ranks to those entities ranked high by most of the rankers. This assumption may not hold in practice, however. For example, in meta-search, ranking lists are generated by different search engines with different capacities and accuracies. It is not reasonable to treat the results of the search engines equally.

To deal with the problem, Aslam *et al* [2] proposed Borda Fuse, which can be viewed as weighted Borda Count for meta-search. Specifically, different rankers are assigned different weights, while the weights are trained separately by using labeled training data. For example, the weights can be calculated based on the MAP (Mean Average Precision) scores of the base rankers. Experimental results show that Borda Fuse indeed improves upon Borda Count. The problem with Borda Fuse is that the weights of the ranking list are calculated independently and by using heuristics. It is also not clear whether the same idea can be applied to other methods.

We note that order-based rank aggregation in meta-search is similar to relevance ranking in document retrieval, but there are some clear differences. Therefore, the methods proposed for relevance ranking may not be directly applicable to order-based rank aggregation. In relevance ranking, a typical approach is to employ a linear combination model of the features to rank documents. One can also employ a supervised learning method to train the model. Each feature can be viewed as a ranker and the final ranking model can be viewed as an aggregation function. However, this final ranking model is more close to that of score-based aggregation, not that of order-based aggregation. How to apply a score-based method to order-based aggregation is still an open problem, and is out of the scope of this paper.

3. SUPERVISED RANK AGGREGATION

In this section, we first introduce a general optimization framework for order-based rank aggregation. We then define Supervised Rank Aggregation methods within the framework.

We first give some definitions and notations. Given a set of entities *S*, let *V* be a subset of *S* and assume that there is a total order among the entities in *V*. τ is called a *ranking list* with respect to *S*, if τ is a list of the entities in *V* maintaining the same total order relation, i.e.,$\tau = [d_1, \cdots, d_m]$, if $d_1 > \cdots > d_m$, $d_i \in V, i = 1, \cdots, m$, where $>$ denotes the relation and *m* denotes the size of *V*. If *V* equals *S*, τ is called a *full list*, otherwise, it is called a *partial list*. A special case of *partial list* is a *top-t list*, for which the first t^{th} entities are ordered in the list.

3.1 Optimization Framework

The goal of rank aggregation is to assign a real-valued score to each of the entities by aggregating all the ranking lists given by the base rankers, and then sort the entities according to their scores. Without loss of generality, hereafter we assume that it is in the descending order.

Let $\tau_1, \cdots, \tau_l$ denote the ranking lists with respect to S and n denotes the number of entities in *S*. We define the *aggregation function* as $\Psi: \tau_1, \cdots, \tau_l \mapsto x$, where x denotes the final score vector of all entities. That is, if $x = \Psi(\tau_1, \cdots, \tau_l)$, then all the entities are ranked by the scores in *x*.

For example in Borda Count, x is called Borda score, which is calculated as,

$$x = \Psi(\tau_1, \cdots, \tau_l) = \sum_{k=1}^{l} x^{(k)} \qquad (3.1.1)$$

where $x^{(k)} \triangleq \left(x_i^{(k)}\right)^T_{i=1,\cdots,n}$, $x_i^{(k)} = \#\{j | i >_{\tau_k} j\}$, and $i >_{\tau_k} j$ means that entity *i* is ranked higher than entity *j* in ranking list τ_k.

We assume that the aggregation function Ψ is parameterized by a parameter vector α. In a supervised learning approach to rank aggregation, we try to learn the optimal values of the parameters by using labeled training data. Typically training data may include ground truth indicating pairwise preferences of which entities should be ranked higher than the others. In the learning, we actually manage to find the aggregation function that minimizes the disagreements between the ground truth and the output of the aggregation function.

We represent the agreement between the output list of an aggregation function and the ground truth by using an inequality

$$Hx < 0$$

where x denotes the output of the function and H denotes a matrix representing the pairwise preference relationship between entities. For example, suppose that the scores produced by the aggregation function are $x = (x_1, x_2, x_3, x_4)^T$, and the ground truth indicates that entity 1 should be ranked higher than entity 2, and entity 4 should be ranked higher than entity 3. Then, the inequality becomes:

$$Hx < 0, \text{ where } H = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

By using such a matrix, we can bring any form of ground truth into our framework, and do not need assume a total order existing over all the entities in the training set.

There is no guarantee that there exists a parameter vector α that satisfies all the pairwise constraints in the ground truth. That is, disagreements may exist. We introduce 'slack variable' t to represent the differences (errors),

$$Hx < t, t \geq 0$$

To reduce training errors is equivalent to minimize the norm of t. Thus we can formalize Supervised Rank Aggregation as the following optimization problem.

$$\begin{aligned} &\min_{x,\alpha,t} \quad t^T t \\ &s.t.\ x = \Psi(\tau_1, \cdots, \tau_l; \alpha) \\ &\quad \alpha \in C \\ &\quad Hx < t, t \geq 0 \end{aligned} \tag{3.1.2}$$

where α denotes the parameter vector and C denotes a feasible region for α. The dimension of matrix H equals the number of pairs indicating pairwise preferences in the training data. The objective $t^T t$ actually denotes the empirical loss in the training data. When empirical loss is 0, the aggregation function Ψ satisfies all the pairwise constraints.

With different ways of instantiating and optimizing the aggregation function, we come to different methods for rank aggregation.

3.2 Methods

We show that we can define Supervised Rank Aggregation methods within the framework. In this paper we only consider the case in which the aggregation function is defined as a *linear model* of base rankers. Even the model is simple; it is powerful enough for accomplishing the tasks in this paper.

(1) Borda Fuse

Many rank aggregation methods are in fact based on majority voting. Borda Count [2][7][27] is such a method and the major assumption within it is that all the base rankers are equally important. As discussed above, it is more reasonable to give different weights to different rankers. In other words, we can consider using Borda Fuse

$$x = \Psi(\tau_1, \cdots, \tau_l) = \sum_{k=1}^{l} \alpha_k\, x^{(k)}$$

Note that Borda Fuse contains Borda Count as its special case.

With the optimization framework in (3.1.2), we can define Supervised Borda Fuse. Specifically we formalize it as the following optimization problem:

$$\begin{aligned} &\min_{x,\alpha,t} \quad t^T t \\ &s.t.\ x = \sum_{k=1}^{l} \alpha_k\, x^{(k)} \\ &\quad \sum_{k=1}^{l} \alpha_k = 1, \alpha_k \geq 0, k = 1, \dots, l \\ &\quad Hx < t, t \geq 0 \end{aligned}$$

where $x^{(k)}$ is the same as that in (3.1.1). Note that the parameter vector is comprised of weights of ranking lists and is to be optimized as well.

(2) Markov Chain based methods

Many other rank aggregation methods are based on Markov Chain. It is advantageous to employ the Markov Chain model in rank aggregation, particularly when the base rankers only output partial lists [8]. Experimental results show that the Markov Chain based methods outperform other methods [24]. That is why we focus on Markov Chain based approach in this paper.

Dwork *et al* [7] proposed four Markov Chain based models for rank aggregation, referred to as MC_1, MC_2, MC_3, and MC_4. The four models correspond to four different heuristic rules for constructing the transition probability matrix in Markov Chain.

Let us take MC_2 as example. The transitions in Markov Chain are defined as follows. If the current state is *i*, then we first select a ranking list τ_k uniformly randomly from the ranking lists $\tau_1, \cdots, \tau_l$ that contain state *i*, then select state *j* uniformly randomly from the set of states that are ranked not lower than state *i* in τ_k, and define *j* as the next state.

For a full list or top-*t* list, it is not difficult to verify that the transition matrix is arithmetic mean of transition probability matrices produced from individual ranking lists, referred to as base-transition matrices. Let $P_k \triangleq \left(p_{ij}^{(k)}\right)_{n\times n}$ denote the k^{th} base transition matrix produced by ranking list τ_k, in which each element $p_{ij}^{(k)}$ corresponds to the conditional probability of state *j* given state *i* in ranking list τ_k. The final transition matrix *P* is defined as

$$P = \frac{1}{l}\sum_{k=1}^{l} P_k$$

$$p_{ij}^{(k)} = \begin{cases} \frac{1}{m}, j >_{\tau_k} i \text{ or } j = i \\ 0, \text{ otherwise} \end{cases} \tag{3.2.1}$$

where $m = \#\{j | j >_{\tau_k} i \text{ or } j = i\}$.

The score vector x can then be computed by solving $x = P^T x$, with constraints $\sum_{i=1}^{n} x_i = 1, x_i > 0, i = 1, \cdots, n$.

In Supervised MC_2 we assign weighting coefficients to the base matrices P_k:

$$P = \sum_{k=1}^{l} \alpha_k\, P_k$$

Formally, Supervised MC_2 is defined as follows.

$$\begin{aligned} &\min_{x,\alpha,t} \quad t^T t \\ &s.t.\ x = \left(\sum_{k=1}^{l} \alpha_k\, P_k\right)^T x \\ &\quad \sum_{i=1}^{n} x_i = 1, x_i > 0, i = 1, \dots, n \\ &\quad \sum_{k=1}^{l} \alpha_k = 1, \alpha_k \geq 0, k = 1, \dots, l \\ &\quad Hx < t, t \geq 0 \end{aligned} \tag{3.2.2}$$

Similarly, we can construct the supervised versions of MC_1, MC_3, and MC_4. The only differences lie in the structures of the transition probability matrices.

a) Supervised MC_1:

The transition matrix of MC_1 can be written as

$$P = diag\left(\frac{1}{\sum_{j=1}^{n} q_{1j}}, \cdots, \frac{1}{\sum_{j=1}^{n} q_{nj}}\right) Q$$

where $Q \triangleq (q_{ij})_{n\times n} = \frac{1}{l}\sum_{k=1}^{l} Q_k,\ Q_k \triangleq \left(q_{ij}^{(k)}\right)_{n\times n}$ with

$$q_{ij}^{(k)} = \begin{cases} 1, j >_{\tau_k} i \text{ or } j = i \\ 0, \text{otherwise} \end{cases}.$$

We can derive Supervised MC_1 by assigning weighting coefficients to Q, and obtain the following optimization problem.

$$\begin{aligned}&\min_{x,\alpha,t} \quad t^T t\\ &s.t.\ x = \left(\sum_{k=1}^{l}\alpha_k Q_k\right)^T diag\left(\frac{1}{\sum_{j=1}^{n} q_{1j}},\cdots,\frac{1}{\sum_{j=1}^{n} q_{nj}}\right)x\\ &\quad \sum_{i=1}^{n} x_i = 1, x_i > 0, i = 1,\ldots,n\\ &\quad \sum_{k=1}^{l}\alpha_k = 1, \alpha_k \ge 0, k = 1,\ldots,l\\ &\quad Hx < t, t \ge 0\end{aligned}$$

b) Supervised MC$_3$:

The formulation of MC$_3$ is similar to that of MC$_2$, except the definition of $p_{ij}^{(k)}$:

$$p_{ij}^{(k)} = \begin{cases}\frac{1}{n}, & j >_{\tau_k} i\\ \frac{n-m}{n}, & j = i\\ 0, & \text{otherwise}\end{cases}, \quad \text{and } m = \#\{j \mid j >_{\tau_k} i\}.$$

Therefore, we can define Supervised MC$_3$ in a similar way as we define Supervised MC$_2$.

c) Supervised MC$_4$:

MC$_4$ is similar to MC$_1$, except that the following two facts differ.

(i) The definition of Q: $Q \triangleq (q_{ij})_{n\times n} = \frac{1}{l}\sum_{k=1}^{l} Q_k$, with $q_{ij}^{(k)} = \begin{cases}1, j >_{\tau_k} i\\ 0, \text{otherwise}\end{cases}$.

(ii) The definition of P: $P \triangleq (p_{ij})_{n\times n}$, with
$$p_{ij}^{(k)} = \begin{cases}\frac{1}{n}, & q_{ij} > \frac{1}{2}\\ \frac{n-m}{n}, & j = i\\ 0, & \text{otherwise}\end{cases}, \text{ and } m = \#\{j \mid q_{ij} > \tfrac{1}{2}\}.$$

Therefore, we can obtain Supervised MC$_4$, similar to Supervised MC$_1$.

In summary, *with the use of the optimization framework*, we can introduce new supervised aggregation methods, corresponding to most of the existing unsupervised rank aggregation methods. The key factor is that weights are assigned to the ranking lists and they are also trained within the optimization framework.

The question next is how to conduct the optimizations. For some forms of function Ψ in (3.1.2), the optimization is hard to solve, such as those in the Markov chain based methods. We know of no existing optimization techniques which can be straightforwardly applied, *because they are not convex optimization problems*. In our work we are able to find an optimization solution for Supervised MC$_2$ on the basis of Semidefinite Programming (SDP), as will be explained below.

4. AN OPTIMIZATION SOLUTION

In this section, we describe our solution to the optimization problem for Supervised MC$_2$ as in (3.2.2). We think that similar techniques can also be applied to other Markov Chain based methods, but leave it as future work.

Our method for Supervised MC$_2$ consists of three steps:

1) We modify the objective and constraints in (3.2.2) to make the feasible region convex.
2) We further transform the optimization problem into a quadratic optimization problem by employing the bound optimization technique.
3) Finally, we transform the quadratic optimization problem into a Semidefinite Programming problem.

Let us elaborate on the three steps in more details. Theoretical justifications of the transformations are given in a lemma and a proposition.

The first constraint in (3.2.2) represents an eigenvector problem. One can easily verify that the feasible region of the optimization problem is not convex. In general such a problem is hard to solve. We reformulate the original optimization problem by putting the first constraint into the objective function:

$$\begin{aligned}&\min_{x,\alpha,t} \quad t^T t + \left\|\left(\sum_{k=1}^{l}\alpha_k P_k\right)^T x - x\right\|_1\\ &s.t.\ \sum_{i=1}^{n} x_i = 1, x_i > 0, i = 1,\ldots,n\\ &\quad \sum_{k=1}^{l}\alpha_k = 1, \alpha_k \ge 0, k = 1,\ldots,l\\ &\quad Hx < t, t \ge 0\end{aligned} \tag{4.1}$$

where $\|\cdot\|_1$denote the ℓ_1-norm of a vector.

Then, the feasible region becomes convex and the objective function becomes one consisting of two parts. The first part $t^T t$ corresponds to training errors, and the second part $\left\|\left(\sum_{k=1}^{l}\alpha_k P_k\right)^T x - x\right\|_1$corresponds to an approximation of the stationary distribution. The second part of the objective function is not convex. We try to minimize a differentiable and convex upper bound of it. Lemma 1 gives the upper bound using the properties of ℓ_1-norm.

Lemma 1: Let $\Xi = (\xi_i)^T_{i=1,\cdots,n} = \left(\sum_{k=1}^{l}\alpha_k P_k\right)^T x - x$**, we have** $\|\Xi\|_1 \le 2 - 2\alpha^T Ax$**, where** $A = \begin{pmatrix}p_{11}^{(1)} & \cdots & p_{nn}^{(1)}\\ \vdots & \ddots & \vdots\\ p_{11}^{(l)} & \cdots & p_{nn}^{(l)}\end{pmatrix}$**.**

Proof: See Appendix.

The optimization problem then becomes

$$\begin{aligned}&\min_{x,\alpha,t} \quad t^T t + 2 - 2\alpha^T Ax\\ &s.t.\ \sum_{i=1}^{n} x_i = 1, x_i > 0, i = 1,\ldots,n\\ &\quad \sum_{k=1}^{l}\alpha_k = 1, \alpha_k \ge 0, k = 1,\ldots,l\\ &\quad Hx < t, t \ge 0\end{aligned} \tag{4.2}$$

By defining $\beta = (\alpha_1,\ldots,\alpha_l,x_1,\ldots,x_n,t_1,\ldots,t_m)^T$, where m is the number of rows in matrix H, and omitting the constant in the objective function which is irrelevant to the optimization, problem (4.2) becomes

$$\begin{aligned}&\min_{\beta}\ \beta^T H_0 \beta\\ &s.t.\ H_1\beta \le 0\\ &\quad H_2\beta = e_2\\ &\quad H_3\beta < 0\end{aligned} \tag{4.3}$$

with
$$H_0 = \begin{pmatrix}0 & -A & 0\\ -A^T & 0 & 0\\ 0 & 0 & I\end{pmatrix} \in R^{(l+n+m)\times(l+n+m)}$$
$$H_1 = \begin{pmatrix}-I_l & 0 & 0\\ 0 & 0 & -I_m\end{pmatrix} \in R^{(l+m)\times(l+n+m)} \tag{4.4}$$
$$H_2 = \begin{pmatrix}e_l^T & 0 & 0\\ 0 & e_n^T & 0\end{pmatrix} \in R^{2\times(l+n+m)}$$
$$H_3 = \begin{pmatrix}0 & -I_n & 0\\ 0 & H & -I_m\end{pmatrix} \in R^{(n+m)\times(l+n+m)}$$

where I_i is identity matrix of size i, and e_i is vector with size i in which all the elements are one.

The optimization in (4.3) is an optimization problem with quadratic objective function and linear constraints. The remaining issue is that the Hessian matrix H_0 is not positive definite and thus the objective function is not convex. In this situation, if we employ a method like Gradient Decent, the solution will be sensitive to the initial values, and will likely to become locally optimal. To cope with it, we further transform the optimization problem into a Semidefinite Programming (SDP), with the theoretical support from Proposition 2.

Proposition 2: Optimization problem (4.3) is equivalent to the following Semidefinite Programming problem,

$$\begin{aligned} &\max_{\lambda,\gamma} \quad \gamma \\ &s.t. \quad \lambda \geq 0 \\ &\begin{pmatrix} H_0 + \lambda_0 D & \frac{1}{2}U^T \\ \frac{1}{2}U & -\Lambda_2^T e_2 - 2\lambda_0 - \gamma \end{pmatrix} \succcurlyeq 0 \end{aligned} \quad (4.5)$$

Where $U = \Lambda_1^T H_1 + \Lambda_2^T H_2 + \Lambda_3^T H_3$, and $\lambda = (\lambda_0, \Lambda_1^T, \Lambda_2^T, \Lambda_3^T)^T$.

Proof: See Appendix.

Finally we can solve the optimization problem using the techniques of SDP[1], for example, the interior-point method SDPA [30] proposed in [12].

Our Supervised MC_2 algorithm can be summarized as follows.

Supervised MC_2:

Input: ranking lists $\tau_1, \cdots, \tau_l$

Output: weighting parameter α

Algorithm:

a) Construct base transition matrices $P_1, \cdots, P_l$ according to equation (3.2.1).

b) Create matrix A as shown in Lemma 1.

c) Create matrices H_0, H_1, H_2, H_3 as shown in equation (4.4).

d) Construct matrix U as shown in Proposition 2.

e) Call SDP tool [30] to solve problem (4.5) and get solution λ.

f) Compute β by equation (8.2.3).

g) Output the first l elements of β as parameter α.

5. EXPERIMENTS

In this section, we report the experimental results on meta-search using our method based on Supervised Rank Aggregation and existing methods. Our first experiment was conducted with TREC dataset, and the second was with data from real web search engines.

5.1 TREC Data

TREC datasets were used in many previous works on rank aggregation [2][18][19][20][24], in which heuristic models were used as base rankers. This motivated us to conduct our experiments with TREC dataset as well. We selected the OHSUMED dataset used in the filtering track of TREC 2000.

The OHSUMED dataset is a collection of 348,566 documents and 106 queries. The ground truth is provided by the TREC committee with three levels of relevance judgments: 'definitely relevant', 'possibly relevant', and 'not relevant' to the query. Based on these judgments, we can construct pairwise constraints for the training of Supervised MC_2.

[1] SDP is a hot research field in recent years [29], and many fast iterative algorithms have been developed [16][21][23][30].

In our experiment, we used 30 ranking models (features) [22] as base rankers. These include term frequency, inverse document frequency, document length, BM25 score [25], and their combinations.

Table 1. Results of different methods for meta-search with OHSUMED data

	Supervised MC_2	MC_1	MC_2	MC_3	MC_4	Borda-Count	Borda Fuse
P@1	**0.483**	0.337	0.376	0.357	0.308	0.349	0.349
P@2	**0.384**	0.324	0.345	0.316	0.294	0.310	0.320
P@3	**0.363**	0.290	0.325	0.280	0.289	0.310	0.306
P@4	**0.352**	0.284	0.312	0.285	0.275	0.280	0.295
P@5	**0.329**	0.266	0.312	0.268	0.280	0.276	0.292
P@6	**0.331**	0.265	0.298	0.265	0.276	0.265	0.272
P@7	**0.324**	0.269	0.296	0.263	0.276	0.266	0.264
P@8	**0.316**	0.269	0.292	0.263	0.274	0.265	0.265
P@9	**0.312**	0.266	0.287	0.260	0.270	0.264	0.261
P@10	**0.302**	0.264	0.288	0.255	0.267	0.254	0.258
MAP	**0.302**	0.275	0.286	0.271	0.269	0.267	0.272

Table 2. Results of different methods for meta-search with OHSUMED data

	Supervised MC_2	MC_1	MC_2	MC_3	MC_4	Borda-Count	Borda Fuse
N@1	**0.651**	0.534	0.573	0.553	0.516	0.544	0.544
N@2	**0.595**	0.530	0.553	0.533	0.513	0.525	0.531
N@3	**0.586**	0.517	0.546	0.515	0.513	0.526	0.525
N@4	**0.581**	0.519	0.546	0.518	0.512	0.514	0.523
N@5	**0.575**	0.514	0.549	0.514	0.517	0.518	0.525
N@6	**0.579**	0.517	0.548	0.518	0.520	0.515	0.520
N@7	**0.583**	0.523	0.554	0.521	0.527	0.522	0.522
N@8	**0.586**	0.530	0.558	0.527	0.533	0.525	0.528
N@9	**0.590**	0.536	0.562	0.532	0.539	0.530	0.533
N@10	**0.589**	0.541	0.568	0.534	0.542	0.531	0.537

Next, we conducted rank aggregation using our method. For comparison, we also implemented and tested other rank aggregation methods, including MC_1, MC_2, MC_3, MC_4, Borda Count, and Borda Fuse. The experiments were performed through 4-fold cross validation. We randomly split the query set into four subsets, used the first two of them for training, the third for validation, and the fourth for testing, and rotated this process four times to create four data sets. Then we took the average performance over the four trials as the final result for each method.

We used three measures in our experiments for ranking accuracy evaluations: Precision [3], Mean Average Precision (MAP) [3] and Normalized Discount Cumulative Gain (NDCG) [14][15]. When evaluating the performances in terms of precision, we regarded both 'definitely relevant' and 'possible relevant' as positive, and 'not relevant' as negative.

Table 1 shows the results in terms of precision at *n* (P@*n*) and MAP, and Table 2 shows the results in terms of DNCG at *n* (N@*n*) for all the methods.

From the results, we can see that Supervised MC_2 outperforms all the other methods, suggesting that it is better to employ Supervised Rank Aggregation proposed in this paper.

5.2 Web Search Data

We also tried to apply the rank aggregation methods directly to meta-search on the web.

5.2.1 *Experimental Results*

We randomly sampled 500 queries from the query log of a commercial search engine, as query set. Table 3 shows some example queries.

Table 3. Sample queries used in meta-search

Queries
Altavista, Astronomy Picture of the day, BBC, cadillac, daily nation, delta dental, family guy, fox theater, Google, group health, habitat for humanity, hotmail, Image Entertainment, imdb, jacksonville news, jetblue, kofax, laredo morning times, liberty university, michael Jordan, Microsoft, national zoo, NCAA football, ohio department of education, philips, prime outlets, southern baptist convention, Superbowl, tacoma news tribune, texas department of public safety, Tuesday Morning, ucla, university of Tennessee, venetian, etc.

Table 4. Results of different methods for meta-search with data from web search engines

	Supervised MC_2	MC_1	MC_2	MC_3	MC_4	Borda-Count	Borda Fuse
P@1	**0.864**	0.738	0.837	0.734	0.818	0.712	0.709
P@2	**0.692**	0.606	0.664	0.611	0.668	0.579	0.581
P@3	**0.620**	0.547	0.575	0.550	0.586	0.505	0.501
P@4	**0.560**	0.505	0.520	0.502	0.529	0.457	0.544
P@5	**0.525**	0.466	0.476	0.469	0.484	0.426	0.423
P@6	**0.496**	0.436	0.446	0.434	0.446	0.394	0.394
P@7	**0.471**	0.408	0.419	0.409	0.413	0.372	0.370
P@8	**0.447**	0.387	0.400	0.388	0.385	0.349	0.350
P@9	**0.426**	0.367	0.383	0.367	0.359	0.332	0.332
P@10	**0.407**	0.350	0.366	0.352	0.340	0.315	0.313
MAP	**0.410**	0.333	0.374	0.333	0.340	0.296	0.292

Next, we submitted the queries to six commercial web search engines, and collected the top-100 ranking lists of the queries returned by the search engines. We combined the results together and eliminated the duplicate pages. On average there were 362 unique pages per query. The overlap among the ranking lists of the search engines was small: there were on average 4 pages per query occurring in all the ranking lists. Then we asked human annotators to make relevance judgments on the pages. The relevance judgments were binary: relevant or irrelevant. Three annotators made judgments, and majority voting was finally conducted on the results.

We then conducted meta-search on the data through *4-fold cross validation* (in the same way as in Section 5.1. We applied our proposed method, and used MC_1, MC_2, MC_3, MC_4, Borda Count, and Borda Fuse as baselines. Table 4 shows the results in terms of P@*n* and MAP. From the results, we can see that our proposed method achieves the best results in terms of both MAP and P@*n*. Again this verifies the effectiveness of our proposed method for rank aggregation.

Table 5 shows the experiment results in terms of NDCG@n.

Table 5. Results of different methods for meta-search with OHSUMED data

	Supervised MC_2	MC_1	MC_2	MC_3	MC_4	Borda-Count	Borda Fuse
N@1	**0.741**	0.554	0.722	0.552	0.690	0.533	0.531
N@2	**0.631**	0.495	0.609	0.485	0.589	0.459	0.460
N@3	**0.605**	0.484	0.576	0.475	0.561	0.440	0.438
N@4	**0.592**	0.483	0.562	0.473	0.545	0.435	0.433
N@5	**0.585**	0.482	0.553	0.476	0.534	0.433	0.431
N@6	**0.582**	0.480	0.548	0.474	0.525	0.432	0.430
N@7	**0.579**	0.477	0.543	0.473	0.517	0.430	0.429
N@8	**0.575**	0.475	0.538	0.471	0.510	0.428	0.428
N@9	**0.572**	0.472	0.536	0.468	0.503	0.427	0.426
N@10	**0.569**	0.469	0.533	0.467	0.498	0.425	0.423

5.2.2 *Discussions*

We investigated why our proposed supervised method (Supervised MC_2) outperforms the baseline methods.

Figure 1 shows MAP and the weight to each search engine assigned by our method in the first trial of the cross-validation experiment. (The results from the other trials have the same tendencies).

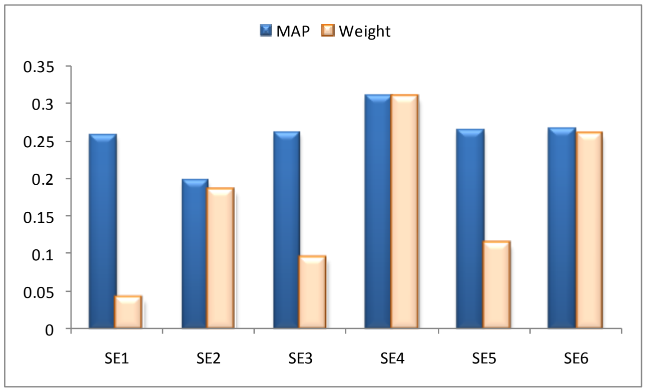

Figure 1. MAP and weights of search engines

From Figure 1, we have the following observations.

(a) *The weights of search engines are different from each other.*

This validates the correctness of our assumption that rankers should have different weights.

(b) *The weights of search engines do not necessarily correlate with their MAP values.*

Although the fourth search engine achieves the best MAP and obtains the largest weight at the same time, for the other engines, MAP and weight do not correlate. For example, the first search engine has a higher MAP than the second, but it has much smaller weight than the second. Our explanation to this is as follows. The weights of search engines not only depend on their performances, but also depend on the correlations among search engines. If a search engine highly correlates to the others, its weight (influence) will be reduced within the general optimization framework.

To verify the correctness of this explanation, we calculate the correlation coefficient between each pair of the six engines using the following formula, and present the results in Table 6. (Note that the correlation is symmetric.)

$$\mathrm{cor}(\mathrm{SE}i, \mathrm{SE}j) = \frac{1}{\#\{\mathrm{query}\}} \sum_{\mathrm{query}} \frac{\#\{(u,v) | (u >_{\mathrm{SE}\,i} v \text{ and } u >_{\mathrm{SE}\,j} v) \text{or } (u <_{\mathrm{SE}\,i} v \text{ and } u <_{\mathrm{SE}\,j} v)\}}{\#\{(u,v) | (u,v \in \mathrm{SE}\,i) \text{and } (u,v \in \mathrm{SE}\,j)\}}$$

where SEi denotes the i-th search engine, $u >_{\mathrm{SE}\,i} v$ means that document u is ranked higher than v by SEi for a given query, and $u, v \in \mathrm{SE}i$ means that documents u and v are returned by search engine SEi.

Table 6. Correlation among search engines

	SE1	SE2	SE3	SE4	SE5	SE6
SE1	1	0.502	0.524	**0.681**	0.615	**0.675**
SE2		1	0.335	0.437	0.433	0.502
SE3			1	0.661	0.530	0.597
SE4				1	0.617	0.696
SE5					1	0.613
SE6						1

From Table 6, we can see that the first search engine highly correlates to the forth and the sixth search engines, and therefore its weight is suppressed by the large weights of the two engines. In contrast, the second search engine only weakly correlates to the other engines, and thus it retains a large weight.

The observation can also give explanation to other results in the experiments. From Table 5, one may see an interesting phenomenon. Borda Fuse, as a supervised method, performs even worse than the unsupervised methods. As explained, Borda Fuse assumes that the weight of each base ranker only depends on its accuracy, and it neglects the correlation among base rankers. It seems that this is not appropriate anyway. Therefore, it appears better to perform rank aggregation using an optimization framework as we do.

6. CONCLUSIONS

In this paper, we have proposed a new approach to rank aggregation: Supervised Rank Aggregation. Our method is mainly designed for meta-search and is unique in that (a) takes order information from base rankers, (b) it makes use of labeled training data, and (c) it trains the final ranking function within a single optimization framework. We have set up a general framework for employing the approach. Specifically, we have formalized the learning problem as that of optimization. We propose an efficient algorithm to solve the optimization for one of the typical rank aggregation settings, namely the Markov chain based method. We have compared the performances of our proposed method with those of existing methods on meta-search. The results show that the proposed method can outperform the existing methods.
The contributions of this paper include

1) proposal on employing the supervised learning approach for rank aggregation;
2) formulation of the supervised learning approach as an optimization problem;
3) development of an optimization algorithm form the Markov Chain based learning method; and
4) empirical verification of the effectiveness of the proposed approach.

As future work, we plan to apply the techniques used in this paper to other supervised learning methods, and to apply the methods to other applications such as similarity search and genome informatics.

ACKNOWLEDGEMENTS

The authors would like to thank Wei-Ying Ma at MSRA for his suggestions and comments on this work. They are also grateful to Shisheng Li at USTC for his helps in the experiments. The authors would also like to thank the anonymous reviewers for their valuable comments on the paper.

REFERENCES

[1] Ahmad N. and Beg M. M. S. Fuzzy Logic Based Rank Aggregation Methods for the World Wide Web, In *Proceedings of the International Conference on Artificial Intelligence in Engineering and Technology*, Malaysia, 2002, 363-368.

[2] Aslam, J. A. and Montague, M. Models for Metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM Press, New York, 2001, 276-284.

[3] Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison Wesley, 1999.

[4] Beg, M. M. S. *Parallel Rank Aggregation for the World Wide Web*. World Wide Web. Kluwer Academic Publishers, vol 6, issue 1, 5-22. March 2004.

[5] Borda, J. C. *Mémoire sur les élections au scrutin*. Histoire de l'Acad´emie Royale des Sciences, 1781

[6] Boyd, S. and Vendenberghe, L. *Convex Optimization*. Cambridge, U. K. Cambridge Univ. Press 2003.

[7] Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. Rank Aggregation Methods for the Web. In *Proceedings of the 10th International World Wide Web Conference*. 2001, 613-622.

[8] Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. *Rank Aggregation revisited*. 2001. Manuscript.

[9] Fagin, R., Kumar, R., and Sivakumar, D. Efficient Similarity Search and Classification via Rank Aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. San Diego, 2003, 301-312.

[10] Fagin, R., Lotem, A., and Naor, M. Optimal Aggregation Algorithm for Middleware. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. Santa Barbara, California, United States, 2001, 102-113.

[11] Fox, E. A. and Shaw, J. A. Combination of Multiple Searches. In *Proceedings of the Second Text Retrieval Conference*, 1994.

[12] Fujisawa, K., Fukuda, M., Kojima, M., and Nakata, K. *Numerical Evaluation of the SDPA (SemiDefinite Programming Algorithm).* High Performance Optimization, Kluwer Academic Press, 267-301, 2000.

[13] Hull, D. A., Pedersen, J. O., and Schütze, H. Method Combination for Document Filtering. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, 1996, 279-287.

[14] Jarvelin, K.and Kekalainen, J. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, 2000, 41-48.

[15] Jarvelin, K.. and Kekalainen, J. *Cumulated Gain-Based Evaluation of IR Techniques*. ACM Transactions on Information Systems, 2002.

[16] Klerk, E. *Aspects of Semidefinite Programming: Interior Point Algorithms and Selected Applications. Applied Optimization Series*, Volume 65. Kluwer Academic Publishers, March 2002, 300 pp.

[17] Lam, K. W. and Leung, C. H. Rank Aggregation for Meta-search Engines. In *Proceedings of the 13th International World Wide Web Conference*. 2004.

[18] Manmatha, R.., Rath, T., and Feng, F. Modeling Score Distributions for Combining the Outputs of Search Engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, 2001, New York.

[19] Manmatha, R. and Sever, H. A Formal Approach to Score Normalization for Meta-search. In *Proceedings of HLT'02*, 2002, 88-93.

[20] Montague, M. and Aslam, J. A. Relevance Score Normalization for Meta-search. In *Proceedings of the 10th Conference on Information and Knowledge Management*. Atlanta, GA, 2001, 427-433.

[21] Monteiro, R. D. C. *First- and Second-Order Methods for Semidefinite Programming*. Georgia Tech, January 2003.

[22] Nallapati, R. Discriminative Models for Information Retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information retrieval*. ACM Press, 2004, 64-71.

[23] Pardalos, P.M. and Wolkowicz, H. *Topics in Semidefinite and Interior Point Methods*. Fields Institute Communications 18, AMS, Providence, Rhode Island, 1998.

[24] Randa, M. E. and Straccia, U. Web metasearch: Rank vs. Score based Rank Aggregation Methods. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, March 09-12, 2003, Melbourne, Florida.

[25] Robertson, S. E. *Overview of the Okapi Projects*. Journal of Documentation, Vol. 53, No. 1, 1997, pp. 3-7.

[26] Sese, J. and Morishita, S. *Rank Aggregation Method for Biological Databases*. Genome Informatics, 12: 506-507, 2001.

[27] Van Erp M. and Schomaker, L. Variants of the Borda Count Method for Combining Ranked Classifier Hypotheses. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*. Amsterdam, 2000, 443-452.

[28] Vogt, C. and Cottrel, G. W. *Fusion via a Linear Combination of Scores*. Information Retrieval, v.1 n.3, p.151-173, October 1999.

[29] Semidefinite Programming. http://www-user.tu-chemnitz.de /~helmberg/semidef.html.

[30] SDPA Online for Your Future. http://grid.r.dendai.ac.jp/sdpa/.

.

APPENDIX

Proof of Lemma 1

Lemma 1: Let $\Xi = (\xi_i)^T{}_{i=1,\cdots,n} = \left(\sum_{k=1}^{l} \alpha_k P_k\right)^T x - x$, we have $\|\Xi\|_1 \le 2 - 2\alpha^T A x$, where $A = \begin{pmatrix} p_{11}^{(1)} & \cdots & p_{nn}^{(1)} \\ \vdots & \ddots & \vdots \\ p_{11}^{(l)} & \cdots & p_{nn}^{(l)} \end{pmatrix}$.

Proof:

Define $P = \sum_{k=1}^{l} \alpha_k P_k$ and $F = (f_{ij})_{n\times n} = P^T$. It is clear that $f_{ij} = \sum_{k=1}^{l} \alpha_k p_{ji}^{(k)}$. Using f_i to denote the ith row of F, we can rewrite Ξ as

$$\Xi = (\xi_i)^T{}_{i=1,\cdots,n} = Fx - x$$

or,

$$\xi_i = f_i x - x_i = (f_{ii} - 1)x_i + \sum_{j=1, j\neq i}^{n} f_{ij} x_j \qquad (8.1.1)$$

Because the transition probability matrix has identical rows, for the right-hand side of equation (8.1.1), $(f_{ii} - 1)x_i$ is non-positive and the others are non-negative. Therefore, we can get an upper bound of $|\xi_i|$ as follows by using the properties of ℓ_1-norm

$$|\xi_i| \le (1 - f_{ii})x_i + \sum_{j=1, j\neq i}^{n} f_{ij} x_j = (1 - 2f_{ii})x_i + \sum_{j=1}^{n} f_{ij} x_j$$

Applying the result to each element in Ξ yields

$$\|\Xi\|_1 \le \sum_{i=1}^{n} (1 - 2f_{ii})x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} x_j$$

Considering that $\sum_{i=1}^{n} x_i = 1$ and $\sum_{i=1}^{n} f_{ij} = 1$, we obtain

$$\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} x_j = \sum_{j=1}^{n} x_j \left(\sum_{i=1}^{n} f_{ij}\right) = \sum_{j=1}^{n} x_j = 1$$

and

$$\sum_{i=1}^{n} (1 - 2f_{ii})x_i = \sum_{i=1}^{n} x_i - 2\sum_{i=1}^{n} f_{ii} x_i = 1 - 2\sum_{i=1}^{n} f_{ii} x_i$$

If further considering $f_{ii} = \sum_{k=1}^{l} \alpha_k p_{ii}^{(k)}$, we obtain

$$\|\Xi\|_1 \le 2 - 2\sum_{i=1}^{n} f_{ii} x_i = 2 - 2\sum_{i=1}^{n} \left(\sum_{k=1}^{l} \alpha_k p_{ii}^{(k)}\right) x_i$$

By using matrix form to represent this inequality, we eventually have

$$\|\Xi\|_1 \le 2 - 2\alpha^T \begin{pmatrix} p_{11}^{(1)} & \cdots & p_{nn}^{(1)} \\ \vdots & \ddots & \vdots \\ p_{11}^{(l)} & \cdots & p_{nn}^{(l)} \end{pmatrix} x = 2 - 2\alpha^T A x.$$

Proof of Proposition 2

Proposition 2: The optimization problem in (4.3) is equivalent to the following Semidefinite Programming problem,

$$\begin{aligned} &\max_{\lambda,\gamma} \quad \gamma \\ &s.t. \ \ \lambda \ge 0 \\ &\begin{pmatrix} H_0 + \lambda_0 D & \frac{1}{2}U^T \\ \frac{1}{2}U & -\Lambda_2^T e_2 - 2\lambda_0 - \gamma \end{pmatrix} \succcurlyeq 0 \end{aligned} \qquad (8.2.1)$$

where $U = \Lambda_1^T H_1 + \Lambda_2^T H_2 + \Lambda_3^T H_3$, and $\lambda = (\lambda_0, \Lambda_1^T, \Lambda_2^T, \Lambda_3^T)^T$.

Proof:

Considering that $\beta = (\alpha_1, \ldots, \alpha_l, x_1, \ldots, x_n, t_1, \ldots, t_m)^T$, we always have

$$\beta^T D \beta = \alpha^T \alpha + x^T x \le \left(\sum_{k=1}^{l} \alpha_k\right)^2 + \left(\sum_{i=1}^{n} x_i\right)^2 = 2$$

where $D = diag(e_{l+n}^T, 0_m^T)$.

It is clear that if we add this redundant constraint to the optimization problem (4.3), its optimal solution will not change because the feasible region has not changed. In this way, we can transform the optimization problem (4.3) into the following quadratically constrained quadratic optimization (QCQP) problem.

$$\begin{aligned} \min_\beta\ & \beta^T H_0 \beta \\ s.t.\ & \beta^T D\beta \le 2 \\ & H_1\beta \le 0 \\ & H_2\beta = e_2 \\ & H_3\beta < 0 \end{aligned} \tag{8.2.2}$$

The Lagrangian of (8.2.2) is

$$L(\lambda,\beta)$$
$$= \beta^T H_0\beta + \lambda_0(\beta^T D\beta - 2) + \Lambda_1^T H_1\beta + \Lambda_2^T(H_2\beta - e_2) + \Lambda_3^T H_3\beta$$
$$= \beta^T(H_0 + \lambda_0 D)\beta + U\beta - \Lambda_2^T e_2 - 2\lambda_0$$

where $\Lambda_1 = (\lambda_1, \lambda_2, \dots, \lambda_{l+m})^T$, $\Lambda_2 = (\lambda_{l+m+1}, \lambda_{l+m+2})^T$, $\Lambda_3 = (\lambda_{l+m+3}, \lambda_{l+m+4}, \dots, \lambda_{l+n+2m+2})^T$, $\lambda = (\lambda_0, \Lambda_1^T, \Lambda_2^T, \Lambda_3^T)^T$, and $U = \Lambda_1^T H_1 + \Lambda_2^T H_2 + \Lambda_3^T H_3$.

According to the optimization theory [6], if the infimum of $L(\lambda,\beta)$ with respect to β exists, one can transform the minimization of the objective function in the primal problem (8.2.2) to the maximization of the dual function $g(\lambda) = \inf_\beta L(\lambda,\beta)$. The condition for this transformation is existence of the infimum of $L(\lambda,\beta)$. We will discuss this condition in the following three cases.

1) If $(H_0 + \lambda_0 D)$ is positive-definite, then function $L(\lambda,\beta)$is a convex quadratic function of β. Therefore, we can find the infimum from the optimality condition:

$$\nabla_\beta L(\lambda,\beta) = 2(H_0 + \lambda_0 D)\beta + U^T = 0$$

which yields

$$\beta = -\tfrac{1}{2}(H_0 + \lambda_0 D)^{-1}U^T \tag{8.2.3}$$

Accordingly, we get the dual function

$$g(\lambda) = -\Lambda_2^T e_2 - 2\lambda_0 - \frac{1}{4}U(H_0 + \lambda_0 D)^{-1}U^T$$

which is a concave quadratic function of λ. If $(H_0 + \lambda_0 D)$ is strict positive semidefinite, using the pseudo-inverse in [6], we can get a relaxation on the above condition. That is if $U \in ran(H_0 + \lambda_0 D)$, we can get the dual function as follows[2],

$$g(\lambda) = -\Lambda_2^T e_2 - 2\lambda_0 - \frac{1}{4}U(H_0 + \lambda_0 D)^\dagger U^T$$

2) Otherwise function $L(\lambda,\beta)$ has no lower bound, thus the problem (8.2.2) has no solution.

With the above discussions, we conclude if and only if $(H_0 + \lambda_0 D) \succcurlyeq 0$, $L(\lambda,\beta)$ has an infimum and the corresponding optimization problem (8.2.2) can be transformed to its dual problem[3]. As a result, we can solve the dual problem in (8.2.4) and get the solution for (8.2.2).

$$\begin{aligned} \max_\lambda\ & g(\lambda) \\ s.t.\ & \lambda \ge 0 \\ & (H_0 + \lambda_0 D) \succcurlyeq 0 \end{aligned} \tag{8.2.4}$$

One can also find that $g(\lambda)$ is the Schur complement [6] of $(H_0 + \lambda_0 D)$ in the matrix $\begin{pmatrix} H_0 + \lambda_0 D & \frac{1}{2}U^T \\ \frac{1}{2}U & -\Lambda_2^T e_2 - 2\lambda_0 \end{pmatrix}$. In this situation, (8.2.4) can be further formulated as a Semidefinite Programming (SDP) problem with respect to variables γ and λ.

$$\begin{aligned} \max_{\lambda,\gamma}\ & \gamma \\ s.t.\ & \lambda \ge 0 \\ & \begin{pmatrix} H_0 + \lambda_0 D & \frac{1}{2}U^T \\ \frac{1}{2}U & -\Lambda_2^T e_2 - 2\lambda_0 - \gamma \end{pmatrix} \succcurlyeq 0 \end{aligned} \tag{8.2.5}$$

For problem (8.2.2), there exists a β that makes the following two inequalities true: $\beta^T D\beta < 2$, $H_1\beta < 0$, and $H_3\beta < 0$. That is, problem (8.2.2) is strictly feasible, and thus the optimal values of (8.2.2) and its Lagrange dual problem (8.2.4) are equivalent [6].

Recall that optimization problem (4.3) and (8.2.2) are equivalent; according to the strong duality theorem [6], problem (4.3) is equivalent to the SDP problem (8.2.5).

[2] $M^\dagger$ is the pseudo-inverse of matrix M [6].

[3] $M \succcurlyeq 0$ means that matrix M is semi-positive definite [6].