iRank: Supporting Proximity Ranking for Peer-to-Peer Applications*

Yongquan Fu and Yijie Wang
National Key Laboratory for Parallel and Distributed Processing, School of Computer
National University of Defense Technology, China
yongquanf@nudt.edu.cn, wwyyjj1971@vip.sina.com

Abstract

Proximity ranking according to end-to-end network distances (e.g., Round-Trip Time, RTT) can reveal detailed proximity information, which is important in network management and performance diagnosis in distributed systems. However, to the best of our knowledge, there has been no similar work on this subject in the P2P computing field. We present a distributed rating method iRank, that enables proximity rankings by providing discrete ratings in a distributed manner. It formulates the proximity ranking as a rating problem that faithfully captures the proximity based on noisy distance measurements scalably and practically. The primary challenge in inferring proximity rankings is enforcing distributed ratings with complex rating policies. Our solution is based on reconstructing ratings by decomposing a centralized rating method Maximum Margin Matrix Factorization (MMMF) into independent sub-problems, that can be efficiently solved in a decentralized manner. By relaxing the dependence on infrastructure nodes that are a single point of failure and limit scalability, iRank can gracefully handle network churns. Through real network latency data sets, we demonstrate that iRank can predict ratings with low distortion, which are smaller than 20 percentage worse than the centralized method, in the context of synthetic complex rating policies.

1 Introduction

This paper introduces a proximity ranking problem into the P2P computing field, which asks to provide ranking information of a set of decentralized nodes which may join or leave independently, based on their pairwise network distances, (e.g., Round Trip Time, RTT, router-level hops, or routing-path weights). For simplicity, we assume the RTT metric is adopted to represent the proximity. Informally, the proximity ranking can be described as for each node i, sorting the rest of nodes in the system by their RTTs towards node i in an ascending order.

The motivations behind the proximity ranking problem are as follows: 1) when diagnosing Internet-scale distributed systems in case of performance degradation in data delivery, users can decide the gains in switching to a new server compared to all the rest of candidate servers, with the proximity ranking results; 2) In the context of network management in the overlays, we may wish to provide the proximity ranking as public performance information for research purpose, thereby anonymizing performance details to avoid the network inference activities by potential malicious users.

Providing proximity ranking in the context of decentralized nodes involves several non-trivial challenges. First, since end hosts may dynamically join or leave systems, the number of hosts constantly changes, so it is hard to assign system-wide ordinal numbers towards the other nodes. Second, each node can not communicate with all nodes in the system due to limited network bandwidths and processing speeds. Third, deploying centralized servers that maintain the memberships of nodes or the ranking information introduce potential single points of failures or performance bottlenecks.

To resolve above challenges, This paper formulates a *weak ranking problem* instead, by relaxing the constraint of providing globally unique ranking positions as follows. First, we compress the global ranks into a deterministic number of ratings, e.g, "5 stars", by aggregating network distances into a set of intervals. For each node, the ranks of other nodes with identical ratings are also the same. As a result, network proximity are preserved while avoiding global rankings. Second, each node computes ratings to a small number of nodes, according to chosen rating functions, then infers ratings of other nodes that are not directly rated in a distributed manner. To the best of our knowledge, we are the first to formulate and solve the proximity ranking

This work was supported in part by the National Natural Science Foundation of China under Grant No.60873215 and No.60621003; the National Grand Fundamental Research 973 Program of China under Grant No.2005CB321801; Specialized Research Fund for the Doctoral Program of Higher Education No.200899980003; A Foundation for the Author of National Excellent Doctoral Dissertation of PR China No.200141.

problem in the context of decentralized nodes.

The weak ranking problem has close relation with the collaborative rating field, which seeks to find efficient matrix factorizations to estimate a partial observed rating matrix. Maximum Margin Matrix Factorization (MMMF) is a popular approach to approximate rating matrices [6, 8]. However, it assumes that the matrix are stored in a central repository, which is reasonable for movie or product ratings. On the contrary, in our decentralized rating context where nodes may join or leave independently, we can not store ratings in a centralized manner.

In this paper, to estimate ratings in a distributed manner, we propose a novel weak ranking method, iRank based on decomposing the well-established fast MMMF [6] into independent sub-problems. First, two rating functions are designed as motivating examples of rating policies. Second, each node maintains a low-dimensional vector triple by solving fast MMMF with a small number of sampled nodes (called landmarks); then the ratings between any pair of nodes are reconstructed based on rounding the products of vectors triples into discrete values with thresholds. We demonstrate through real network latency data sets that ratings can be reconstructed accurately with low dimensional vectors, in a distributed manner.

2. Related work

Ranking by network distances has close relation with the research of network coordinate methods for latency estimation [3, 2, 1]. For example, GNP [3] represents the first coordinate computation methods for network distance estimation, where nodes update their coordinates based on a nonlinear optimizing process. Vivaldi [1] is a popular decentralized coordinate method with the spring field simulation. IDES [2], which is based on Singular Value Decomposition (SVD) or Nonnegative Matrix Factorization (NMF), tries to factorize the network distance matrix into two low dimensional sub-matrices with small errors. Network coordinates are suitable for long-term latency estimation, but fail to preserve the ranking information of network distances [10]. In this paper, we seek to directly estimate ranking information, based on predicting ratings of network distances, which can include application preferences and communication histories to enhance the reliability and stability of ranking.

On the other hand, our distance ranking by rating methodology is inspired by well-known collaborative filtering techniques. Both our weak ranking problem and the collaborative filtering could be formalized as a matrix completion problem [8]: completing entries in a partially observed data matrix D (network distance rating matrix in our settings) with an approximated matrix X; moreover, for practical storage and processing, both of them demand a low rank representation of X. However, finding a low-rank matrix minimizing discrete rating distortions is a non-convex problem, with multiple local minima [8]. To this end, MMMF [8, 6] provides an alternative convex optimization formulation $X = UV$ by constraining the norms of U and V instead of their dimensionality, which can be viewed as constraining the overall "strength"of the factors in a factor model. As a result, this low-norm factorization formulation of MMMF leads to a convex constraint [6, 9].

We are aware that Rish and Tesauro [7] have adopted MMMF to predict the binary performance metrics of end-to-end continuous bandwidth and latencies, in a centralized manner. Besides, they incorporate active learning techniques to choose the best training samples for classifier accuracy. Instead, we are interested in providing ranking information by estimating ratings in a distributed manner. Furthermore, we incorporate semantics of users to design flexible rating functions.

3. Motivation

3.1. Weak Ranking Problem

Assumes that there exists a *rating alphabet* $Y = \{1, \ldots, m\}$ where $m \ll N$, and each node determines its ranking policy by defining a rating function that maps RTT measurements towards other nodes to items in Y. The rating function can be seen as a mapping function from the ranges of RTT measurements to discrete values, i.e., $f : [r_{left}, r_{right}] \rightarrow N$. The *weak ranking problem* is to estimate ratings based on local information in a distributed manner. Specifically, each node maintains ratings to a small number of nodes according to chosen rating functions, then infers ratings of other nodes that are not directly rated in a distributed manner.

Target to the weak ranking problem, we seek to design distributed algorithms that predict weak rankings between end nodes with low-dimensional vectors, inspired by the research of network coordinates. The algorithmic effectiveness builds upon several principles: (i) *Fine-granularity*. The rating function should map pairwise distances into variable-length intervals according to application preferences. (ii) *Accuracy*. The estimated ratings should be close to real ratings, based on low dimensional vectors. (iii) *Scalability*. The prediction process should be implemented in a distributed manner, and avoid the situation that any nodes may be performance bottlenecks. (iv) *Resilience*. The performance of ranking predictions should degrade gracefully when a subset of nodes are unreachable or measurements are failed.

3.2. Background of MMMF

Assume that we are given a partially observed rating matrix, then the missing items can be filled with the MMMF method. Here we introduce the idea, which is the basis of our method. Given a pairwise rating matrix $D_{n\times n}$, where $D_{ij} \in \{0, 1, \ldots, R\}$, R represents the maximum rating, and $D_{ij} = 0$ indicates that node i does not rated node j. Let Ω denote the whole set of nodes. For any matrix Y, let Y_{i*}, Y_{*j} denote the ith row and jth column of Y, respectively. Let $\|Z\|_F$ denotes the Froebenius norm: $\sqrt{\Sigma_{ij} Z_{ij}^2}$. Let X be the approximated rating matrix.

MMMF [6, 8] approximates the rating matrix D with a linear factor model $X = UV$, wherein U is $n \times d$, and V is $d \times n$, and the dimension d is $d \ll n$ for efficient computation and storage. To find appropriate matrices of U and V, MMMF minimizes a regularized loss function with Froebenius norms of U and V for capacity control and overfitting prevention [9]. More specifically, MMMF minimizes the objective $J(U, V, \theta)$ with loss function h:

$$J(U,V,\theta) = \sum_{r=1}^{R-1} \sum_{i,j\in\Omega, D_{ij}>0} h(T_{ij}^r(\theta_{ir} - U_{i*}V_{*j})) + \frac{\lambda}{2}(\| U \|_F^2 + \| V \|_F^2) \quad (1)$$

where

$$T_{ij}^r = \begin{cases} +1 & r \geq D_{ij} \\ -1 & r < D_{ij} \end{cases}$$

That is, T_{ij}^r imposes penalty on each threshold θ_{ir} violated by current prediction $U_{i*}V_{*j}$. Furthermore, smoothed hinge loss $h(z) = max(0, 1 - z)$ (as in Support Vector Machine) makes J differentiable and optimization easier. Lastly, λ is the regularization parameter for the norms of U and V.

MMMF predicts the discrete ratings of $X_{ij} \in R$ ($X_{ij} = U_{i*}V_{*j}$) by comparing X_{ij} with $R-1$ thresholds θ_{ir} learned by node i. An efficient implementation of MMMF is based on the Polak-Ribiere variant of Conjugate Gradients adopted by [6], with the consecutive gradient independence test, in order to find when to restart the direction of exploration. This approach is verified to be efficient and fast [6].

4. iRank

4.1. Design Overview

The main idea of iRank is to predict discrete ratings of network distances with low dimensional vectors based on MMMF [6], in a distributed manner. First, a small subset of landmark nodes initialize the vector triples based on MMMF process. Second, newly-joining nodes or existing nodes update their vector triples by probing a set of nodes that have initialized their vector triples. The benefits of adopting MMMF are as follows. First, viewed as a factor model to approximate the data, MMMF can model an infinite number of factors where a few of them are allowed to be very important. That is, even each node has complex rating functions, MMMF can still accurately predict ratings. Second, MMMF leads to convex optimization problems that are separated into subproblems and tractable with current optimization techniques.

iRank includes the following components: (i) *Rating Function*: Each node can have its personalized rating function specified according to its preferences. For simplicity, we present two simple rating functions, as motivating examples in Section 4.2. (ii) *Vector Space*: Each node i is assigned a low dimensional rating vector triple $< u_i, v_i, \theta_i >$. The first two vectors u_i, v_i which are of dimension d, predict continuous rating scores, and the third vector θ_i specifies thresholds for final discrete ratings. This formulation is based on MMMF, and is resilient to missing measurements or node failures. (iii) *Distributed Rating Computation*: we present a decentralized low-dimensional vector optimization process based on separating MMMF into independent problems, without the reliance of any infrastructure nodes. First, a set of nodes compute their rating vector triples to bootstrap the computation process. Second, all nodes independently contact a set of randomly sampled online nodes that has computed their rating vector triples to update their vector triples. As a result, iRank can gracefully adapt to the network churns.

4.2. Rating Functions

It should be reminded that different nodes can use different rating functions, to satisfy their specific preferences. To demonstrate example rating functions, we introduce two simple rating functions named Uniform Intervals (UI) and Exponential Intervals (EI). Assume that we have learned the maximum distance of all node-pairs as Δ, and the maximum number of distance ranges as Λ. For simplicity, the maximum distance can be set as a threshold value, such that all latency measurements above the threshold value are put into the maximum range. UI is defined as $[\frac{i\Delta}{\Lambda}, \frac{(i+1)\Delta}{\Lambda}] \rightarrow N, i \in [1, \Lambda]$, where all distance ranges are identical. EI is defined as $[a^i, a^{i+1}] \rightarrow N, i \in [1, \Lambda]$, where distance ranges are exponentially increased according to the parameter a, whose default value is 2. UI-based ratings assign equal importance to different distance ranges; while EI-based ratings express preferences towards short distances against long distances. Clearly, there is a natural tradeoff between the number of distance ranges and the granularity of the rating functions. More distance ranges produce finer-granularity ratings, but may lead to lower stability of ratings

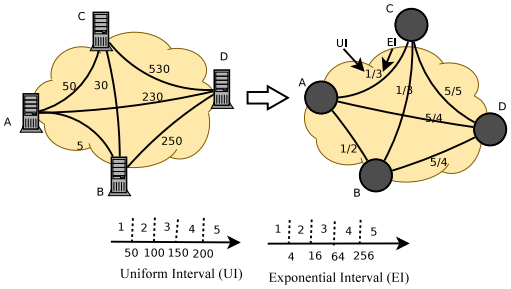

Figure 1. An illustration of weak ranking problem, the parameter a=4

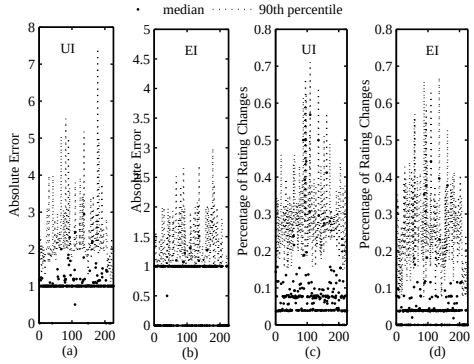

Figure 2. Stability of ratings, where the x-axis corresponds to the indexes of nodes

due to dynamic network conditions.

For instance, network distances could be mapped to "5 stars" range intervals, as shown in Figure 1. Note that low network latencies correspond to smaller ratings, due to the definitions of UI and EI for convenience.

4.3. The Stability of Ratings

We conduct stability test of our example rating functions, with a ping trace file which records the network distance measurements between 226 PlanetLab nodes during a four-hour-long period [5]. The maximum distance for rating functions is set as the 90th percentile of all pairwise distances to balance the distributions of mapped rating values. The size of the rating alphabet is 10.

We compute the absolute error of the ratings for each available pair of hosts, defined as $error_{t+1}(i,j) = \alpha \times error_t(i,j) + (1-\alpha) \times |L_{t+1}(i,j) - L_t(i,j)|$,where α is a weight factor, set as 0.9, and $L_t(i,j)$ denotes the rating of node *j* by node *i* at time t. The missing item is set as 0. As seen from Figure 2, with UI or EI based discrete ratings, the median absolute errors for most hosts are ≤ 2 for UI based ratings and ≤ 1 for EI based ratings; moreover, 90th percentile of absolute errors of most EI based ratings are below 2, while 90th percentile of most UI based ratings incur comparatively larger absolute errors, which are also below 3. It can be concluded that the ratings are around the initial ratings.

We also quantify the perturbation of ratings by each node, defined as the ratio of the number of rating changes to the number of overall measurements. Figure 2 indicates that the median value and 90th percentile of perturbation ratios are rather small, mostly are below 30%. Furthermore, most of the median rating changes of UI and EI are smaller than 0.1. As a result, the ratings can be cached for applications. Meanwhile, there are several nodes which incur relatively larger absolute errors or rating changes, this is because the changes of network distances span more than one distance ranges specified by UI or EI rating functions. Also, we found that the degrees of the rating instability become severe, when the length of the rating alphabet increases.

4.4. Distributed Rating Computation

A. Centralized Rating Computation. A centralized rating process is straightforward based MMMF in Section 3.2. First, each node i selects a subset of nodes as neighbor nodes; then node i computes the ratings of these neighbor nodes based on its rating function; and node i sends the rating vectors of its neighbor nodes to a centralized node. Second, the centralized node uses MMMF to estimate the low dimensional factorization matrices U and V.

Based on MMMF, the vector space for rating estimation can be described as follows: each node is assigned a rating vector triple, with the form $< u_i, v_i, \theta_i >$, where u_i is the ith row vector of U, v_i is the ith column vector of V, and θ_i is the ith row vector of θ. Then, the rating prediction for node j by node i is computed as t, where $\theta_{i(t-1)} \leq u_i v_j \leq \theta_{it}, t \in [1, R]$.

B. Distributed Implementation. The distributed computation of iRank is based on decomposing MMMF into separable sub-problems. More Specifically, by decomposing the objective (1) of MMMF into independent problems as follows:

$$J(U,V,\theta) = \sum_{i \in \Omega} J_{i,\Omega} \tag{2}$$

$$J_{i,\Omega} = \sum_{r=1}^{R-1} \sum_{j \in \Omega, D_{ij} > 0} (h(T_{ij}^r(\theta_{ir} - u_i v_j)) + h(T_{ji}^r(\theta_{jr} - u_j v_i))) + \frac{\lambda}{2}(\| u_i \|_F^2 + \| v_i \|_F^2) \tag{3}$$

After decomposition, node i can minimize the optimization objective (3), by independently contacting each node j in Ω. By taking full use of objective decomposition of MMMF,

we propose a novel distributed rating computation process, which is divided into two phases as follows.

Initialization: A set of random nodes (named as landmarks) select their personalized rating functions, probe each other, then compute the ratings of other landmarks based on measurement results, and send the ratings to a bootstrap node R_n (R_n could be any node in the system). With the rating matrix about landmark nodes, node R_n adopts the MMMF process in Section 3.2 to compute the rating vector triples for landmarks. Then, node R_n disseminates the vector triples to the corresponding landmarks.

The communication and computation overheads of the initialization process are typically low, which depend on the size of landmarks. Furthermore, the initialization process is robust to missing items due to unreachable measurements in the real world. Specifically, by simply setting the ratings of missing items in the rating matrix as zeros, MMMF can easily handle incomplete rating matrix.

Update: Existing nodes periodically update their rating vector triples to adapt to the dynamic network conditions; meanwhile, newly-joining nodes need to initialize their vector triples, both in a decentralized manner.

To compute the rating vector triple scalably, node i randomly samples a set of nodes B, and optimizes the objective in the form of (3), by contacting nodes in B instead of the whole set of nodes Ω. Observe that the optimization process needs both ratings from node i to each node in B, and the ratings in reverse directions. The gradients of the optimization process can be shown as:

$$\frac{\partial J}{\partial u_{ia}} = \lambda u_{ia} - \sum_{r=1}^{R-1} \sum_{m|m\in B, D_{im}>0} T^r_{im} h'(T^r_{im}(\theta_{ir} - u_i v_m)) v_{ma}$$

$$\frac{\partial J}{\partial v_{ia}} = \lambda v_{ia} - \sum_{r=1}^{R-1} \sum_{m|m\in B, D_{im}>0} T^r_{mi} h'(T^r_{mi}(\theta_{mr} - u_m v_i)) u_{ma}$$

$$\frac{\partial J}{\partial \theta_{ir}} = \sum_{m|m\in B, D_{im}>0} T^r_{im} h'(T^r_{im}(\theta_{ir} - u_i v_m))$$

Now we can adopt the conjugate gradient method as the MMMF, over a randomly subset of nodes B, in a distributed manner.

Remark: The size of landmarks in the initialization process is a natural trade-off between the system-wide accuracy and the efficiency: smaller size incurs lower communication overheads, but may lead to less accurate local minima due to sparseness coverage of the network distance space; while larger size increases the network communication costs, however, by covering a broader range of nodes, it can improve the prediction accuracy. The neighborhood size $|B|$ represents the local view of each node, which is a trade-off similar with the set of landmark nodes in the initialization process. The size of B is equal to the size of landmarks by default.

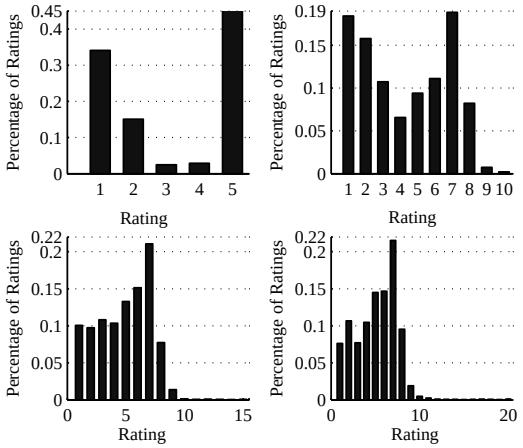

Figure 3. Distributions of ratings.

5. Evaluation

5.1. Experiment Setup

We have implemented iRank with Matlab 7.6. We set the tradeoff parameter λ of iRank and MMMF as $10^{\frac{1}{1.9}}$, which performs quite well empirically. The performance metrics include: Zero-one error (ZOE) and Mean absolute error (MAE) between predicted ratings $\hat{X}$ and actual ratings D. ZOE and MAE are popular metrics in the collaborative rating field [6, 8]. Let $S = \{ij \mid D_{ij} > 0\}$ represents the set of pairs of nodes with non-zero ratings. ZOE is defined as $ZOE(\hat{X}, D) = \frac{1}{|S|}\sum_{ij\in S} \mathbf{1}_{\hat{X}_{ij}\neq D_{ij}}$, which accounts for the percentage of mis-predicted ratings. MAE is defined as $MAE(\hat{X}, D) = \frac{1}{|S|}\sum_{ij\in S} |D_{ij} - \hat{X}_{ij}|$, representing the degrees of dissimilarities between predicted ratings and original ones.

We compare iRank with MMMF and IDES [2]. Note that IDES is not designed to predict ratings, it estimates continuous network latencies. We only intend to show the feasibility of predicting ratings with network coordinates. The maximum iterations of iRank, MMMF and IDES are 100.

The real-world network latency data sets we evaluated include a pairwise RTT matrix of 512 DNS servers from the p2psim project [4], and network latency matrices from PlanetLab (denoted as PL) from [2]. We presented median results from 10 repeated experiments.

The distributions of ratings of all nodes on the K512 data set are shown in Figure 3. The ratings are clearly separated, when the sizes of the rating alphabets are 10, 15 and 20, except 5. There are also several rating numbers corresponding to small-sized items when the sizes of the rating alphabets are 15 and 20. Similar results are found on the PL data set. Finding rating functions that optimally partitions rat-

ings with balanced pairs of nodes are open problems.

5.2. Results

We compare the rating performance of iRank with MMMF [6] and IDES. Due to space limits, rating results on the K512 data set are presented, similar results are found on the PL data set. The size of neighbors of iRank and IDES are set as 35, since we found that more neighbors do not significantly improve the rating performance. The number of ratings (denoted as layers) varies from 5 to 20. 50% nodes use the UI rating function and the others use the EI rating function. To ensure fair comparisons, we compute network coordinates for all nodes, then we convert the estimated network distance matrix into a rating matrix based on rating policies adopted by each node. The results are shown in Figure 4.

First, the rating accuracy of MMMF is better than that of iRank or IDES, for the formal case, since MMMF and iRank share the same optimization process, while MMMF find optimal solutions with global rating information, and iRank use only local information to find local minima; for the latter case, this is because IDES is easily caught by bad local minima when predicting RTTs. Second, iRank performs much better than IDES, and with the increment of dimensions, the performance gains become larger. Third, all three methods converge to stable rating estimation when the size of the rating alphabet exceeds 10. These results indicate that iRank can estimate relatively accurate ratings, whose ZOE and MAE is about averaged 15% and 20% worse than those of the centralized MMMF.

Furthermore, We vary the degree of missing items on the network distance matrix, to further evaluate the performance of rating estimation. We find that both iRank and MMMF approaches do not evidently change the accuracy of rating estimations, as the growth of missing percentages from 0 to 40%; while IDES incurs significantly larger errors as the increments of missing elements, with ZOE around 0.85 and MAE over 2 when the missing percentage is 40%.

6. Conclusions and Future Directions

We introduce the weak ranking problem for P2P applications based on ratings; and design a novel distributed rating estimation method, iRank for scalable and accurate weak ranking. We demonstrated with low dimensionality and small-sized neighbors, iRank can efficiently estimate ratings with promising accuracy, through experiments on real-world network latency data sets.

Two directions are quite interesting. First, the rating functions can embed information from complex application preferences, and thus are expected to be extended broadly. Second, incorporating partial order information of pairwise network latencies into iRank may improve the rating estimation quality substantially.

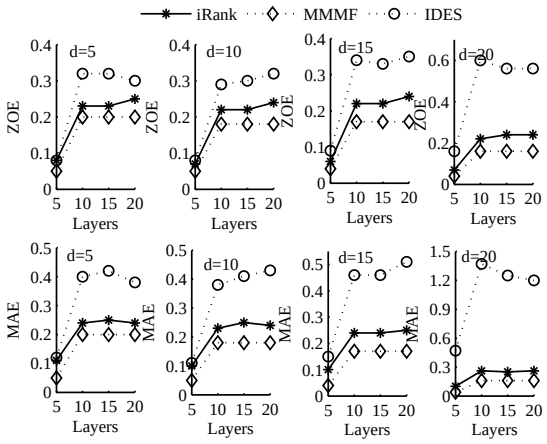

Figure 4. Comparisons of rating accuracy on the K512 data set by varying the dimension d and the number of ratings.

References

[1] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *SIGCOMM*, pages 15–26, 2004.

[2] Y. Mao and L. K. Saul. Modeling distances in large-scale networks by matrix factorization. In *Internet Measurement Conference*, pages 278–287. ACM, October 25-27 2004.

[3] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *INFOCOM*, pages 170–179, 2002.

[4] P2PSim. http://pdos.csail.mit.edu/p2psim/.

[5] Pings.4hr. http://www.eecs.harvard.edu/ syrah/.

[6] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 713–719. ACM, August 7-11 2005.

[7] I. Rish and G. Tesauro. Estimating end-to-end performance by collaborative prediction with active sampling. In *Integrated Network Management*, pages 294–303. IEEE, May 21-25 2007.

[8] N. Srebro, J. D. M. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, December 13-18 2004.

[9] M. Weimer, A. Karatzoglou, and A. Smola. Improving maximum margin matrix factorization. *Machine Learning*, 72(3):263–276, 2008.

[10] R. Zhang, Y. C. Hu, X. Lin, and S. Fahmy. A hierarchical approach to internet distance prediction. In *ICDCS*, page 73. IEEE Computer Society, July 4-7 2006.