Information filtering via preferential diffusion

Linyuan Lü[1,2,*] and Weiping Liu[2]

[1]*Web Sciences Center, University of Electronic Science and Technology of China, Chengdu 611731, People's Republic of China*

[2]*Department of Physics, University of Fribourg, Chemin du Musée 3, CH-1700 Fribourg, Switzerland*

(Received 27 February 2011; revised manuscript received 11 May 2011; published 29 June 2011)

Recommender systems have shown great potential in addressing the information overload problem, namely helping users in finding interesting and relevant objects within a huge information space. Some physical dynamics, including the heat conduction process and mass or energy diffusion on networks, have recently found applications in personalized recommendation. Most of the previous studies focus overwhelmingly on recommendation accuracy as the only important factor, while overlooking the significance of diversity and novelty that indeed provide the vitality of the system. In this paper, we propose a recommendation algorithm based on the preferential diffusion process on a user-object bipartite network. Numerical analyses on two benchmark data sets, *MovieLens* and *Netflix*, indicate that our method outperforms the state-of-the-art methods. Specifically, it can not only provide more accurate recommendations, but also generate more diverse and novel recommendations by accurately recommending unpopular objects.

DOI: 10.1103/PhysRevE.83.066119 PACS number(s): 89.75.Hc, 89.20.Ff, 05.70.Ln

I. INTRODUCTION

The development of information technology brings great impact on human society. Therein, the most significant aspect is the revolutionary change in the ways of life. Twenty years ago, if one wanted to buy something, he or she had to personally go to a physical shop and purchase, and then bring the things back home. It was impossible for him or her to compare the commodities in different markets located at different places in a short time. Now with the growth of the Internet and World Wide Web, we can almost manage our life at home. When we want to buy a book, we don't need to go to bookstores any more to find it on bookshelves one by one; instead, what we need to do is type the title of this book on the website of Amazon—an online retailer of books. If we want to buy a cell phone, we can compare the prices on different web shops at the same time without any transportation fee. Formerly, we usually went to a bar after working and enjoyed making friends there; now, we prefer online dating that allows us to reach people over the world. In a word, the Internet benefits us by providing a much more convenient way to get what we want. However, as a coin has two sides, the Internet also brings us confusion—we face information overload. As we know, not all online information is good or true or favored by surfers. Therefore, we need to distinguish and select between valuable information and junk. In this sense, to get what we want or the most satisfying things becomes more and more difficult, since we face many more choices than before. A useful information filtering technology is a search engine [1,2], by which users can find the relevant information with properly chosen keywords or tags. However, search engines have two disadvantages that limit their applications. First, they lack the consideration of personalization and thus return the same results to people no matter what their preferences are. Secondly, search engines require the users to know exactly what they want and extract some proper keywords to do the searching. However, sometimes tastes or preferences cannot easily be expressed by keywords or the users don't even know what they want at all. In these cases, the search engines are of no avail.

To address these problems, recommender systems rise in response to the proper time and conditions, which form or work from a specific type of information filtering technique that attempts to recommend information items, such as movies, TV programs, videos, music, books, news, images, and web pages, that are likely to be of interest to the users. The recommender systems don't require specified keywords provided by users; instead they use the users' historical activities and possible personal profiles to uncover their preferences or potential interests. Many recommendation algorithms have been developed, including collaborative filtering (CF) [3–5], content-based analysis [6], spectral analysis [7,8], and iterative self-consistent refinement [9,10]. What most have in common is that they are based on similarity, either of users or objects or both. Such an approach is under high risk of providing poor coverage of the space of relevant items. As a result, with recommendations based on similarity rather than difference, more and more users will be exposed to a narrowing band of popular objects. Although it seems more accurate to recommend popular objects than niche ones, being accurate is not enough [11]. It was pointed out that the recommendations that are most accurate are sometimes not the recommendations that are useful to users. For example, would you use such a system that recommends the movies you indeed like but have seen before or just watched in the cinema? Diversity and novelty are also important criteria of algorithmic performance. A possible way to increase the recommendation diversity is utilizing the tags of objects [12–14]. Another promising way is considering the dissimilar users' contribution. It was shown that, under the framework of collaborative filtering, the dissimilar users can contribute to both accuracy and diversity of personalized recommendation [15]. However, these improvements are very limited.

Recently, some physical dynamics, including mass diffusion [16,17] and heat conduction process [18], have been

*linyuan.lue@unifr.ch

applied to design recommender systems. Zhou *et al*. proposed a network-based inference method (NBI) by considering the three-step mass diffusion starting from the target user on a user-object bipartite network [16]. This method has been demonstrated to be more accurate than the classical CF algorithm, with lower computational complexity. However, it has difficulty in generating diverse recommendations. The heat conduction process has been found to be effective in providing a diverse recommendation at the cost of accuracy. This diversity-accuracy dilemma can be effectively solved by coupling these two processes [19]. It was shown that not only does the hybrid algorithm outperform other methods but that, without relying on any semantic or context-specific information, it can be tuned to obtain significant gains in both accuracy and diversity of recommendations.

With the same motivation, we proposed an algorithm based on a preferential mass diffusion process on user-object bipartite networks, without consideration of heat conduction, which may stealthily hurt accuracy. Numerical analyses on two benchmark data sets show that our method can give higher accurate as well as more diverse and novel recommendations than the hybrid algorithm, because of its high accurate recommendations on low-degree objects.

II. PREFERENTIAL DIFFUSION METHOD

A recommender system can be represented by a bipartite network $G(U,O,E)$, where $U = \{u_1,u_2,\ldots,u_m\}$, $O = \{o_1,o_2,\ldots,o_n\}$, and $E = \{e_1,e_2,\ldots,e_q\}$ are the sets of users, objects, and links, respectively [5]. Denote by $A_{m\times n}$ the adjacency matrix, where the element $a_{i\alpha}$ equals 1 if u_i has collected object o_α, and 0 otherwise.

The essential task of a recommender system is to generate a ranking list of the target user's uncollected objects. The original diffusion-based recommendation algorithm, called network-based inference (NBI), was proposed in Ref. [16]. It was referred to as ProbS algorithm in Ref. [19]. NBI works by assigning objects an initial level of resource denoted by the vector $\mathbf{f}$ (where f_α is the resource possessed by object o_α), and then redistributing it via the transformation $\mathbf{f}' = W\mathbf{f}$, where

$$w_{\alpha\beta} = \frac{1}{k_{o_\beta}} \sum_{l=1}^{m} \frac{a_{l\alpha}a_{l\beta}}{k_{u_l}} \tag{1}$$

is the resource transfer matrix, and $k_{o_\beta} = \sum_{i=1}^{n} a_{i\beta}$ and $k_{u_l} = \sum_{\gamma=1}^{m} a_{l\gamma}$ denote the degrees of object o_β and user u_l, respectively. For a target user u_i, we assign one unit resource on those objects already collected by u_i for simplicity; thus the initial resource vector $\mathbf{f}$ can be written as

$$f_\alpha = a_{i\alpha}. \tag{2}$$

That is to say, if object o_α is collected by user u_i then it has one unit resource, otherwise 0. With this initial resource vector, the result of NBI is equivalent to a three-step random walk process starting from the target user on a bipartite network [20]. Therefore, the NBI score of an object is indeed proportional to the probability that a random walker released at the target user happens to arrive at this object after three steps (i.e., user-object-user-object). Note that, if the initial resource vector is normalized by the target user's degree, namely $f_\alpha = a_{i\alpha}/k_{ui}$, the NBI score and the random walk probability are exactly the same. In fact, the process of NBI is equivalent to resource allocation, which is also a random-walk-based process. Given the initial resource distribution as shown in Eq. (2), the resource of each object will be redistributed according to Eq. (1), where $w_{\alpha\beta}$ indicates how much proportion of resource object α gives to object β. Then after the resource-allocation process, we obtain the final resource possessed by each object by summing up all the resources distributed from other objects. The recommendation list for user u_i is generated by ranking all his or her uncollected objects in decreasing order according to their final resource.

Another method referred to as HeatS in Ref. [19] employs a process analogous to heat diffusion across the user-object network. In this algorithm, the objects that the users have already collected are considered as hot sources, while the others as cold points. After two steps of heat diffusion, the cold points that obtain higher heat will be selected as the relevant objects. Similar to ProbS, HeatS also redistributes resources in a manner akin to a random-walk process. However the difference is significant in the diffusion process: the HeatS algorithm redistributes a resource via a nearest-neighbor averaging process, while the ProbS algorithm works by equally distributing the resource to the nearest neighbors. Mathematically, the difference lies in the transition matrix. For HeatS, it reads

$$w_{\alpha\beta} = \frac{1}{k_{o_\alpha}} \sum_{l=1}^{m} \frac{a_{l\alpha}a_{l\beta}}{k_{u_l}}. \tag{3}$$

Clearly, the transition matrix of HeatS is row normalized, while for ProbS it is column normalized. Figure 1 gives an example of the resource spreading processes with ProbS and HeatS algorithms on a user-object bipartite network. The target user is indicated by the shaded circle. Since the target user has collected the first and third objects, we assign each of them one unit resource. For ProbS, the resource will be evenly

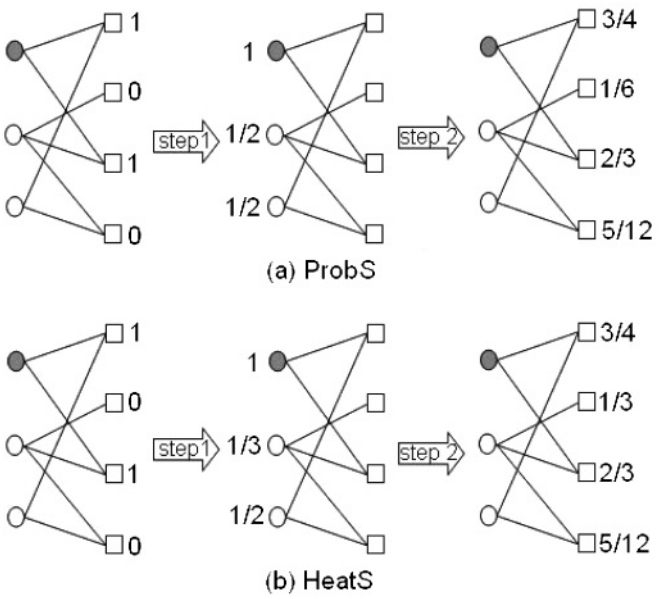

FIG. 1. (a) ProbS and (b) HeatS algorithms [Eqs. (1) and (3)] at work on the bipartite user-object network. Users are shown as circles; objects are squares. The target user is indicated by the shaded circle.

distributed to its neighbors. Thus, after one step diffusion from object side to user side, the three users will respectively obtain 1, 1/2, and 1/2 units of resource, which will be evenly redistributed back to those users' neighboring objects during the second step from user side to the object side. On the contrary, in HeatS the resource is redistributed via an averaging procedure, with users receiving a level of resource equal to the mean amount possessed by their neighboring objects, and objects then receiving back the mean of their neighboring users' resource levels. Note that in ProbS total resource levels remain constant, whereas in HeatS this is not so. It has been pointed out that ProbS has high recommendation accuracy yet low diversity, while HeatS, which is designed specifically to address the challenge of diversity, succeeds in seeking out novel or niche objects and thus enhancing the personalization of individual user recommendations but with relative low accuracy [19]. An effective way to solve the accuracy-diversity dilemma is to combine HeatS (i.e, heat conduction) and ProbS (i.e., mass diffusion) by incorporating the hybridization parameter λ into the transition matrix normalization [19]:

$$w_{\alpha\beta} = \frac{1}{k_{o_\alpha}^{1-\lambda} k_{o_\beta}^{\lambda}} \sum_{l=1}^{m} \frac{a_{l\alpha} a_{l\beta}}{k_{u_l}}, \tag{4}$$

where $\lambda = 0$ gives the pure HeatS algorithm and $\lambda = 1$ gives the ProbS (i.e., NBI).

A heterogenous initial resource distribution NBI algorithm (HNBI) was proposed by Zhou *et al.* [21], where the initial resource of object o_α is proportional to $k_{o_\alpha}^{\theta}$.Thus the initial resource vector of HNBI can be written as $f_\alpha = a_{i\alpha} k_{o\alpha}^{\theta}$, where θ is a negative parameter. It was shown that HNBI can give more accurate recommendations than the standard NBI. There are other two advanced recommendation algorithms. One is an improved algorithm that eliminates redundant correlations (called RENBI for short) [22], which is defined as

$$\mathbf{f}' = \left(W + \eta W^2\right)\mathbf{f}, \tag{5}$$

where the elements of matrix W are defined by Eq. (1), the initial resource vector $\mathbf{f}$ is defined by Eq. (2), and η is a free parameter. This method has been approved to outperform some classical methods, such as the global ranking method, the cosine-similarity-based collaborative filtering [23], NBI, and HNBI for both accuracy and diversity by considering the high-order correlations between objects.

Based on the mass diffusion method and motivated by enhancing the algorithm's ability to find unpopular and niche objects, we propose a preferential diffusion (PD) method for recommendation in user-object bipartite networks. The basic idea is that at the last step (i.e., diffusing from users to objects), the amount of resource that an object o_α receives is proportional to $k_{o_\alpha}^{\varepsilon}$, where $\varepsilon \leqslant 0$ is a free parameter. In this case, the resource transfer matrix reads

$$w_{\alpha\beta} = \frac{1}{k_{o_\beta} k_{o_\alpha}^{-\varepsilon}} \sum_{l=1}^{m} \frac{a_{l\alpha} a_{l\beta}}{\mathcal{M}}, \tag{6}$$

where $\mathcal{M} = \sum_{r=1}^{n} a_{lr} k_{o_r}^{\varepsilon} = k_{ul} E(a_{lr} k_{o_r}^{\varepsilon})$. $E(a_{lr} k_{o_r}^{\varepsilon})$ indicates the mean value of $k_{o_r}^{\varepsilon}$ over all the objects having been collected by user u_l. Here we consider the simplest initial resource vector defined by Eq. (2). Clearly, when $\varepsilon = 0$, it will degenerate to NBI. Notice that, if we consider the NBI algorithm as a three step diffusion from target user to final objects (i.e., user→object→user→object), then the HNBI algorithm is essentially equivalent to the algorithm with preferential diffusion only at the first step, while PD considers the third step. However, their motivations are essentially different. HNBI emphasizes that users who cocollected unpopular objects are more similar to each other than those who co-collected popular objects. Thus the target user distributes more resource to his or her more similar users by giving more resource to their cocollected unpopular objects. However, after the third step diffusion, the resource still can be centralized on some popular objects. The PD algorithm directly punishes the popular object by assigning more resource to the low-degree objects at the last step. Experimental results show that considering the preferential diffusion at the last step is much more effective than at the first step. In order to show that preferential diffusion at first step (i.e., HNBI) and at last step (i.e., PD) play different roles in recommendation, we further investigate the PD algorithm with heterogenous initial resource distribution, called HPD, which is controlled by two tunable parameters. Compared to all the mentioned algorithms in this paper, HPD

TABLE I. Algorithmic performance for *MovieLens* data. The precision, intrasimilarity, hamming distance, and popularity are corresponding to $L = 50$. HNBI is an abbreviation of NBI with heterogenous initial resource distribution, proposed in Ref. [21]. RENBI is an abbreviation of redundant-eliminated NBI, proposed in Ref. [22]. HPH refers to the hybrid method that combines ProbS and HeatS algorithms. PD is an abbreviation of the preferential diffusion method presented in this paper. HPD is an abbreviation of PD with heterogenous initial resource distribution. The parameters (ranging in the interval [0,1] for HPH and [−1,0] for the remaining five algorithms with step 0.05) for the parameter-dependent algorithms are set as the ones corresponding to the lowest ranking scores [for HNBI, $\theta_{\text{opt}} = -0.80$; for RENBI, $\eta_{\text{opt}} = -0.75$; for HPH, $\lambda_{\text{opt}} = 0.20$; for PD, $\varepsilon_{\text{opt}} = -0.85$; for HPD, $(\theta,\varepsilon)_{\text{opt}} = (-0.25, -0.8)$]. Each number is obtained by averaging over five runs with independently random division of training set and probe set. The entries corresponding to the best performance over all methods (except HPD) are emphasized in black.

Algorithms	Ranking score	Precision	Intrasimilarity	Hamming distance	Popularity
NBI	0.106	0.071	0.355	0.617	233
HNBI	0.101	0.074	0.340	0.680	220
RENBI	**0.082**	**0.085**	0.326	0.788	189
HPH	0.085	0.083	0.296	0.821	167
PD	**0.082**	0.084	**0.282**	**0.847**	**155**
HPD	0.081	0.086	0.278	0.858	153

TABLE II. Algorithmic performance for *Netflix* data. The precision, intrasimilarity, hamming distance, and popularity are corresponding to $L = 50$. The parameters (ranging in the interval [0,1] for HPH and [−1,0] for the remaining five algorithms with step 0.05) for the parameter-dependent algorithms are set as the ones corresponding to the lowest ranking scores [for HNBI, $\theta_{\rm opt} = -0.70$; for RENBI, $\eta_{\rm opt} = -0.75$; for HPH, $\lambda_{\rm opt} = 0.20$; for PD, $\varepsilon_{\rm opt} = -0.85$; for HPD, $(\theta,\varepsilon)_{\rm opt} = (-0.2, -0.8)$]. Each number is obtained by averaging over five runs with independently random division of training set and probe set. The entries corresponding to the best performance over all methods (except HPD) are emphasized in black.

Algorithms	Ranking score	Precision	Intrasimilarity	Hamming distance	Popularity
NBI	0.050	0.050	0.366	0.424	2366
HNBI	0.047	0.051	0.341	0.545	2197
RENBI	**0.039**	**0.062**	0.336	0.629	2063
HPH	0.045	0.057	0.311	0.625	1998
PD	0.041	0.057	**0.295**	**0.639**	**1900**
HPD	0.040	0.057	0.266	0.708	1742

performs the best over all five evaluation metrics considered in this paper (see Sec. III for the definitions of evaluation metrics). Comparing Eq. (4) to Eq. (6), we can find that if we assume that for user u_l who has collected object o_β, the approximation $E(a_{lr}k^{\varepsilon}_{o_r}) \approx k^{\varepsilon}_{o_\beta}$ holds, namely the mean value of $k^{\varepsilon}_{o_r}$ over all the objects having collected by user u_l always equals $k^{\varepsilon}_{o_\beta}$, PD is equivalent to the hybrid algorithm by setting $\varepsilon = \lambda - 1$. However, this assumption is too strong to be satisfied in reality.

Note that we didn't consider the preferential diffusion at the second step from the object side to the user side (PDII for short). The main reason is that this method may lead to some illogical results. Consider the case that the target user u_i selected a very popular object o_α that was also selected by another user u_j, who is assumed to be a new user of the system and only selected o_α. Via the PDII method, u_j will obtain more resources from o_α than other users who also selected o_α, leading to the conclusion that u_j is more similar to u_i. Apparently, this result is wrong, since a new user usually selects popular objects, which is a common behavior in such kind of systems [24], and it is unreasonable to say this new user is more similar to the target user just according to such a common behavior. In addition, we have tested the performance of the PDII method. Compared to the standard NBI method, the improvement of accuracy (measured by ranking score) is very slight—around 1% on *MovieLens* data and 0.6% on *Netflix* data. Therefore, we didn't consider this method for further analysis.

III. DATA AND METRICS

To test the algorithmic performance, we use two benchmark data sets. The *MovieLens* (http://www.grouplens.org/) data consists of 1682 movies (objects) and 943 users who can vote for movies with five level ratings from 1 (i.e., worst) to 5 (i.e., best). The original data contains 10^5 ratings. Here we only consider the ratings higher than 2. After coarse gaining, the data contains 82 520 user-object pairs. The *Netflix* data (http://www.netflixprize.com/) is a random sampling of the whole records of user activities in *Netflix.com*. It consists of 10 000 users, 6000 movies, and 824 802 links. Similar to the *MovieLens* data, only the links with ratings no less than 3 are considered. After data filtering, there are 701 947 links left. To test the algorithmic performance, the data (i.e., known links) is randomly divided into two parts: the training set E^T contains 90% of the data and the remaining 10% of data constitutes the probe set E^P. Notice that any isolate object cannot be recommended to users through the algorithms considered in this paper. Therefore, to ensure the connectivity of the whole network, each time before moving a link to the probe set, we first check if this removal will result in an isolate user or object,

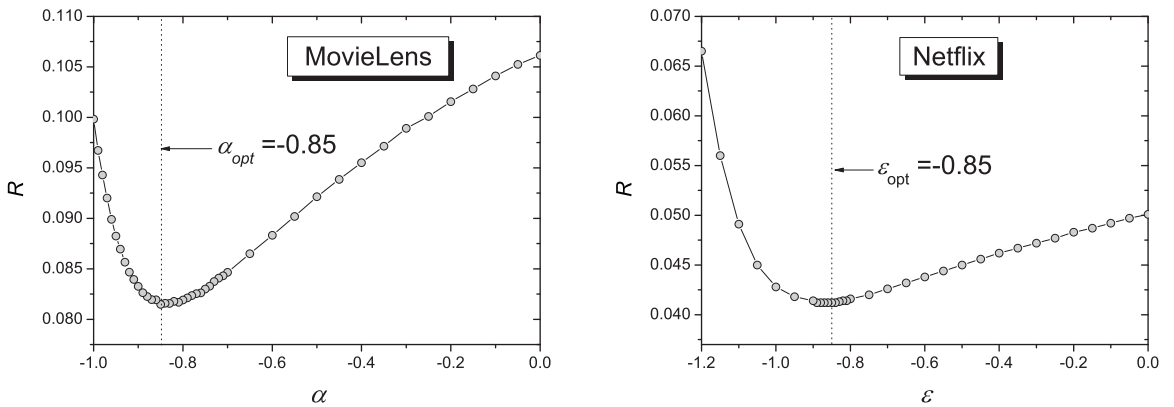

FIG. 2. Ranking score R vs ε. Each data point is obtained by averaging over five runs, each of which has an independently random division of training set and probe set. The optimal parameters ε for *MovieLens* and *Netflix*, corresponding to the minimal R, are both equal to −0.85.

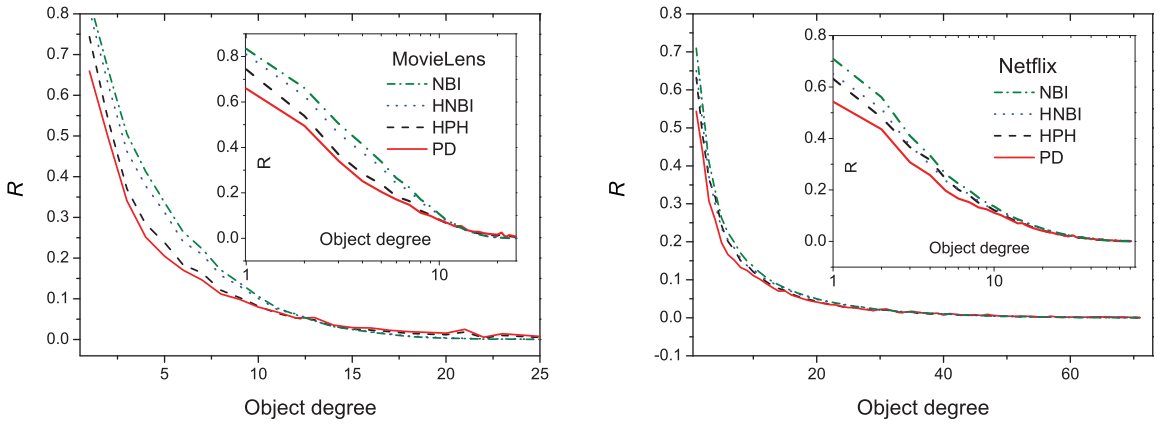

FIG. 3. (Color online) Dependence of ranking score $\langle R\rangle$ on the object degree. For a given x, its corresponding R is obtained by averaging over all the objects whose degrees are in the range of $[a(x^2-x),a(x^2+x)]$, where a is chosen as $\frac{1}{2}\log 5$ for a better illustration. Insets show R against logarithm of x.

and we do not allow the removal that leads to unconnected nodes.

Accuracy is the most important aspect in evaluating the recommendation algorithmic performance. A good algorithm is expected to give accurate recommendations, namely higher ability to find what the users like. Here we use *ranking score* (R) [16] to measure the ability of a recommendation algorithm to produce a good ordering of objects that matches the user's preference. For a target user, the recommender system will return a ranking list of all his uncollected objects to him. For each hidden user-object relation (i.e., the link in probe set), we measure the rank of this object in the recommendation list of this user. For example, if there are 1000 uncollected objects for user u_i, and object o_α is at 10th place, we say the position of this object is 10/1000, denoted by $R_{i\alpha}=0.01$. A good algorithm is expected to give high ranks to the hidden objects, and thus lead to small R. Averaging over all the hidden user-object relations, we obtain the mean value of ranking score R that can be used to evaluate the algorithm's accuracy, namely

$$R=\frac{1}{|E^P|}\sum_{i\alpha\in E^P}R_{i\alpha}, \tag{7}$$

where $i\alpha$ denotes the probe link connecting u_i and o_α. Clearly, the smaller the ranking score, the higher the algorithm's accuracy, and vice versa. Since real users usually consider only the top part of the recommendation list, a more practical measure may be to consider the number of a user's hidden links contained in the top-L places. Therefore, we adopt another accuracy metric called *precision*. For a target user u_i, the precision of recommendation, $P_i(L)$, is defined as

$$P_i(L)=\frac{d_i(L)}{L}, \tag{8}$$

where $d_i(L)$ indicates the number of relevant objects (namely the objects collected by u_i in the probe set) in the top-L places of a recommendation list. Averaging the individual precisions over all users with at least one hidden link, we obtain the mean precision $P(L)$ of the whole system.

Besides accuracy, diversity is taken into account as another important aspect to evaluate the recommendation algorithm. There are two kinds of diversity. One is called *interdiversity*, which considers the uniqueness of different users' recommendation lists. Given two users u_i and u_j, the difference between their recommendation lists can be measured by the Hamming distance [21],

$$H_{ij}(L)=1-\frac{C_{ij}(L)}{L}, \tag{9}$$

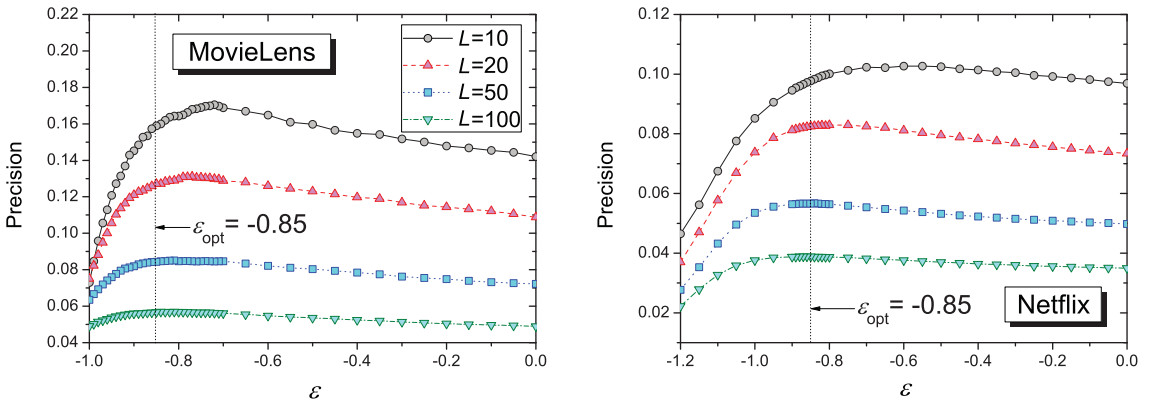

FIG. 4. (Color online) Dependence of precision on parameter ε. Each data point is obtained by averaging over five independent runs with data division identical to the case shown in Fig. 2. The vertical dotted line indicates the optimal parameter ε subject to the lowest ranking score.

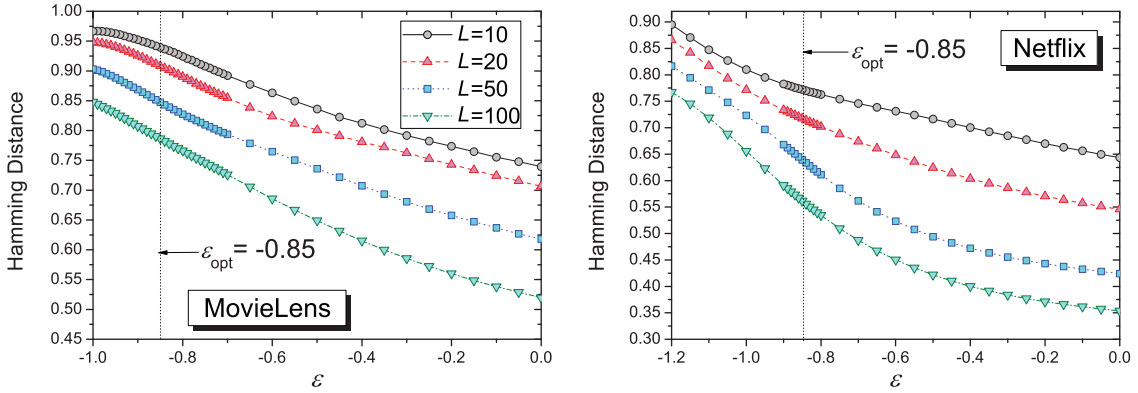

FIG. 5. (Color online) Hamming distance vs ε. Each data point is obtained by averaging over five independent runs with data division identical to the case shown in Fig. 2. The vertical dotted line indicates the optimal parameter ε subject to the lowest ranking score.

where $C_{ij}(L)$ is the number of common objects in the top-L places of both lists. Clearly, if u_i and u_j have the same list, $H_{ij}(L) = 0$, while if their lists are completely different, $H_{ij}(L) = 1$. Averaging $H_{ij}(L)$ over all pairs of users, we obtain the mean distance $H(L)$, for which greater or lesser values mean, respectively, greater or lesser personalization of users' recommendation lists. A good algorithm should not only give diverse recommendations among users (i.e., high interdiversity), but also provide diverse recommendations for a single user (i.e., high intradiversity) [22,25]. The latter can be measured by *intrasimilarity*. For a target user u_i, whose recommended objects are $\{o_1, o_2, \ldots, o_L\}$, the intrasimilarity of u_i's recommendation list is defined as [22]

$$I_i(L) = \frac{1}{L(L-1)} \sum_{\alpha \neq \beta} s^o_{\alpha\beta}, \tag{10}$$

where $s^o_{\alpha\beta}$ is the similarity between objects o_α and o_β in u_i's recommendation list. There are many similarity indices that can be used to quantify the similarity between objects [26]. Here we adopt the widely used cosine similarity to measure

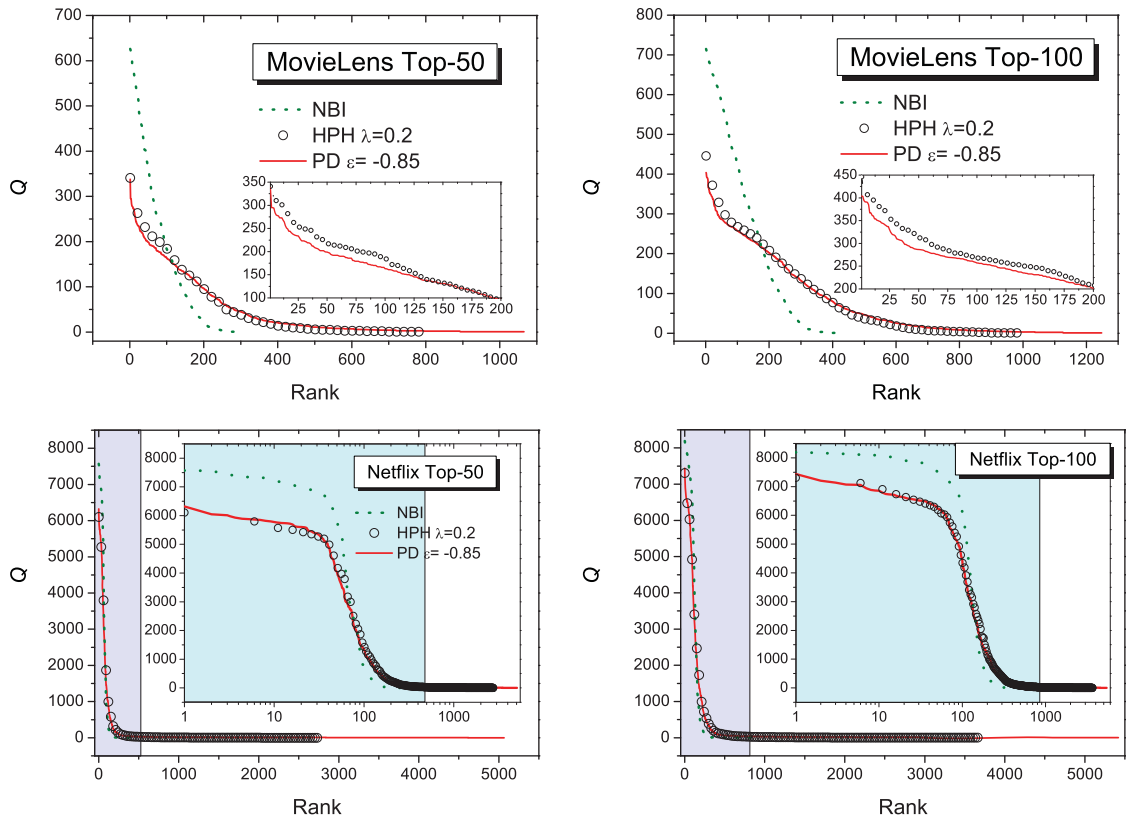

FIG. 6. (Color online) Relationship between the recommended times Q of objects and their ranks for two data sets. Insets of two *MovieLens* subfigures show the results of the top 200 frequently recommended objects. Insets of two *Netflix* subfigures show Q against the logarithm of x (i.e., rank). For NBI, only the objects inside the blue region have the chance to be recommended.

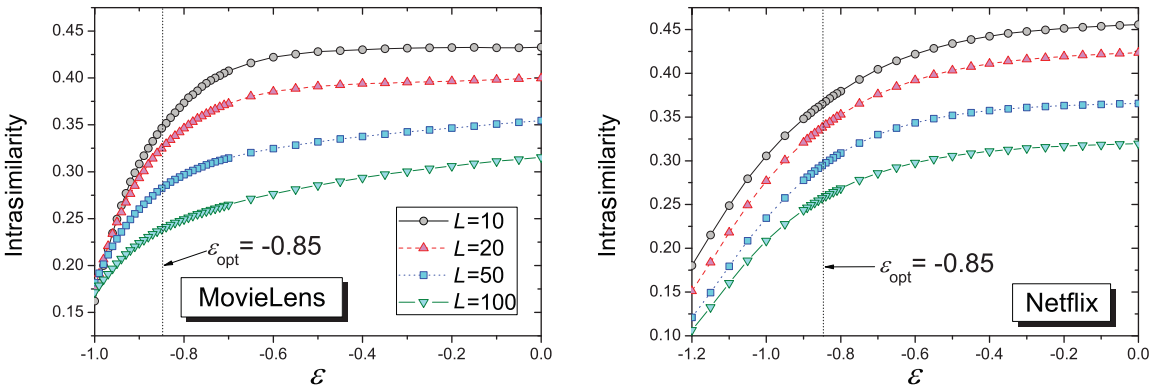

FIG. 7. (Color online) Intrasimilarity as a function of ε. Each data point is obtained by averaging over five independent runs with data division identical to the case shown in Fig. 2. The vertical dotted line indicates the optimal parameter ε subject to the lowest ranking score.

object similarity. For two objects o_α and o_β, their similarity is defined as

$$s^o_{\alpha\beta} = \frac{1}{\sqrt{k_{o_\alpha} k_{o_\beta}}} \sum_{l=1}^{m} a_{l\alpha} a_{l\beta}. \tag{11}$$

Averaging $I_i(L)$ over all users, we obtain the mean intrasimilarity $I(L)$ for the system. A good recommendation algorithm is expected to give fruitful recommendations and has the ability to guide or help the users to exploit their potential interest fields, and thus leads to a lower intrasimilarity (i.e., higher intradiversity).

Highly accurate recommendations might not be satisfied by the users. For example, recommending the popular film *Avatar* to a user on the *MovieLens* website may not always be the best recommendation, because he or she might have already seen this film at the cinema. A diverse recommender system is expected to find the niche or unpopular objects that cannot be easily known by other ways yet match users' preferences. The metric *Popularity* quantifies the capacity of an algorithm to generate novel and unexpected results, that is to say, to recommend less popular items unlikely to be already known. The simplest way to calculate popularity is to use the average collected times over all the recommended items, as

$$N_i(L) = \frac{1}{L} \sum_{o_\alpha \in O_R^i} k_{o_\alpha}, \tag{12}$$

where O_R^i is the recommendation list for user u_i. Clearly, lower popularity indicates higher novelty and surprisal. Averaging $N_i(L)$ over all users, we obtain the mean popularity $N(L)$ for the system.

IV. RESULTS

Summaries of the results for all algorithms and metrics on *MovieLens* and *Netflix* data sets are shown respectively in Tables I and II. The so-called *optimal parameters* are subject to the lowest ranking score. The other four metrics, namely precision, intrasimilarity, hamming distance, and popularity, are obtained at the optimal parameters. Clearly, PD outperforms HNBI over all five evaluation metrics. Among all four previous algorithms, RENBI gives the highest accuracy by considering the high-order correlations between objects,

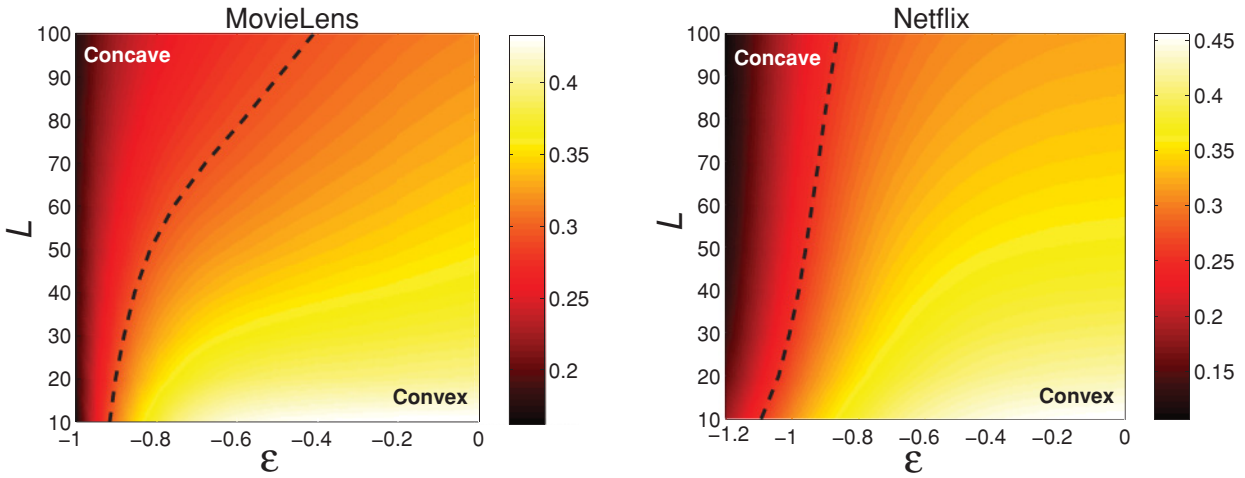

FIG. 8. (Color online) Intrasimilarity I in (ε, L) plane for two data sets. The numerical simulation run over the parameter L in the interval [10,100] with step length equal to 10, and the parameter ε in the interval $[-1,0]$ and $[-1.2,0]$ for *MovieLens* and *Netflix*, respectively, with step 0.05. All the results are obtained by averaging over five independent runs with data division identical to the case shown in Fig. 2. The dashed line indicates that, with the parameter combination (ε, L) on this line, the intrasimilarity equals the value of the system.

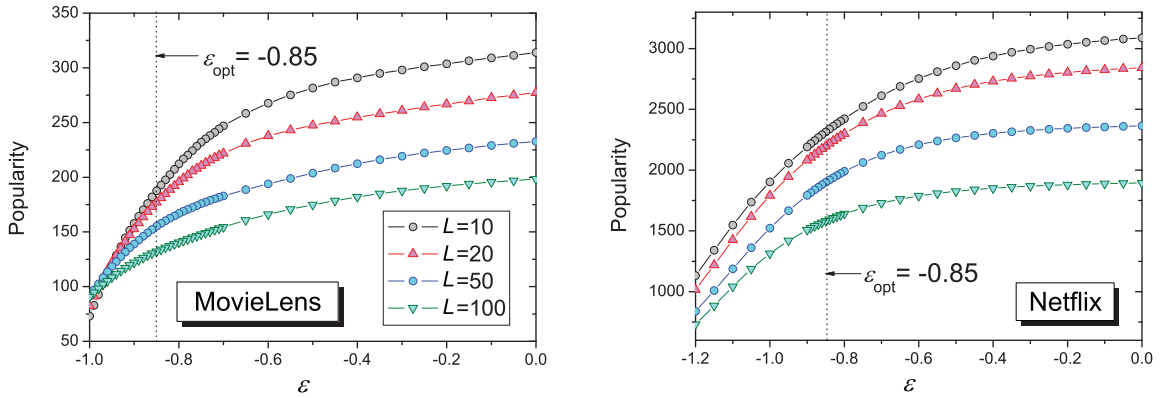

FIG. 9. (Color online) Dependence of popularity (i.e., average degree) on ε. Each data point is obtained by averaging over five independent runs with data division identical to the case shown in Fig. 2. The vertical dotted line indicates the optimal parameter ε subject to the lowest ranking score.

while HPH has the best performance on diversity and novelty. Comparing with these two outstanding algorithms, PD can reach or closely near the best accuracy without considering high-order correlation between objects, and provide much more diverse results. By considering the heterogenous initial resource distribution, the algorithmic performance can be further improved. For example, in *MovieLens*, HPD decreases the ranking score to 0.081 with the parameters $\theta = -0.25$ and $\varepsilon = -0.8$, which is the lowest among all the methods referred to in this paper. Although, with a heterogenous initial resource distribution, both accuracy and diversity can be improved, compared to a pure PD algorithm, such improvements are less remarkable. This indicates that PD actually plays the main role of improvements.

For PD algorithm, the dependence of parameter ε on accuracy measured by ranking score is shown in Fig. 2. The optimal values of parameter ε corresponding to the lowest ranking score on two data sets are both equal to 0.85. Compared to the standard case NBI, namely $\varepsilon = 0$, the ranking score can be reduced by 23% for *MovieLens* and 18% for *Netflix*. We further investigate the dependence of ranking score on the object degree of four methods, namely NBI, HNBI, HPH, and our method PD. The results are shown in Fig. 3. Notice that, for a given x, its corresponding R is obtained by averaging over all the objects whose degrees are in the range $(\frac{x^2-x}{2}\log 5, \frac{x^2+x}{2}\log 5]$. Insets show the R against the logarithm of x. It can be seen that the ranking score decreases with the increasing of the object degree for all these four algorithms. This indicates that, on average, popular objects can be more accurately recommended than the unpopular objects. The significant differences of these four algorithms are embodied on their ability to accurately recommend unpopular objects. Clearly, PD works best for this task, and is followed by HPH. Moreover, comparing the results of HNBI with PD, we can see that, although they both consider the preferential diffusion from user to object, considering at the first step (i.e., HNBI) has much less effect on the unpopular objects than directly acting on the final step (i.e., PD).

Figure 4 shows how the precision changes with the parameter ε for four typical lengths of recommendation list. Given L, there exists an optimal parameter ε leading to the highest precision. Although this optimal parameter ε_1 is different from that subject to the lowest ranking score ε_2, the precision obtained with ε_2 is also considerably higher than that obtained by NBI. For example, when $L = 50$, with the optimal parameter corresponding to the lowest ranking score, the precision is prominently improved by 18% and 14% for *MovieLens* and *Netflix*, respectively.

Hamming distance actually measures the ability of an algorithm to give personalized recommendations. How the parameter ε affects the Hamming distance is shown in Fig. 5. Clearly, a smaller ε leads to a higher Hamming distance (i.e., higher interdiversity) and thus a more personalized recommendation. Compared to the standard case NBI, given $L = 50$, Hamming distance can be enhanced by 37% for *MovieLens* and 56% for *Netflix* with optimal parameters corresponding to their respective lowest ranking scores, even higher than the HPH algorithm. As a result, our method has a higher ability to find the niche (unpopular) objects that may be liked by users, and thus give a more personalized recommendation to the target user. To give more evidence, for a given algorithm we collect the top-L recommended objects for each user. We denote by d the number of distinct objects among all the recommended objects. Then we rank the d objects according to their recommended times, denoting by Q_i $(i = 1, \ldots, d)$, in decreasing order. The relationships between the objects' recommended times Q and their ranks are shown in Fig. 6. We have tested for many different L, and here take $L = 50$ and $L = 100$ as typical examples. Two important phenomena can be obtained from Fig. 6. First, comparing three algorithms, NBI, HPH with $\lambda = 0.2$, and PD with $\varepsilon = -0.85$, we have $d_{\mathrm{PD}} > d_{\mathrm{HPH}} > d_{\mathrm{NBI}}$. That is to say that PD provides a larger number of distinct objects to users than NBI and

TABLE III. Basic properties of the two data sets (the training set) and the average computing time for one recommendation $\langle T \rangle$ with PD algorithm. Sparsity is defined as $\frac{|E|}{N_u N_o}$.

Network	$\lvert E\rvert$	N_u	N_o	$\langle K_u\rangle$	$\langle K_o\rangle$	Sparsity	$\langle T\rangle$
MovieLens	74249	943	1561	79	48	5.04×10^{-2}	0.24 ms
Netflix	634 588	10 000	5586	63	113	1.13×10^{-2}	2.6 ms

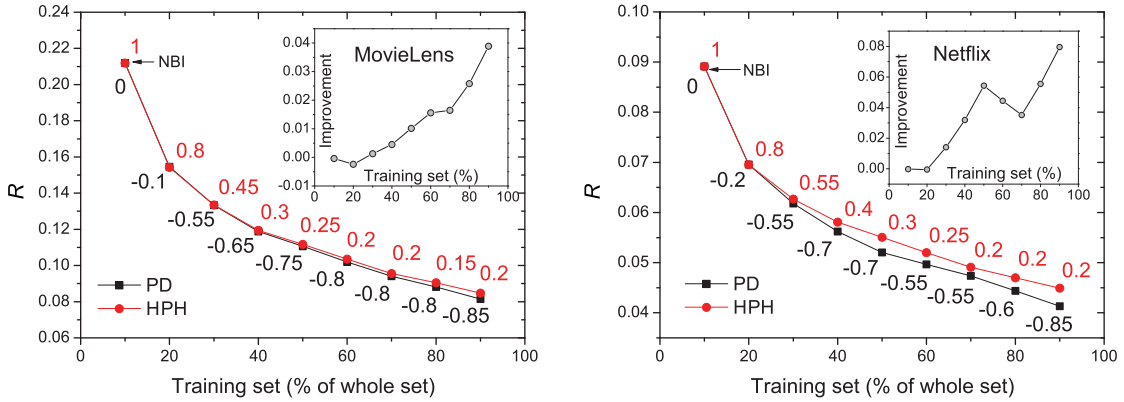

FIG. 10. (Color online) Ranking score changes with the size of the training set measured by the percentage of the whole data set. That is to say, we change the size of the training set from 10% to 90% to respectively predict the remaining 90% to 10%. Each data point is obtained with the parameter ($\varepsilon \in [-1,0]$ for PD and $\lambda \in [0,1]$ for HPH with step 0.05) subject to the lowest R. The optimal parameters are labeled in black for PD and red for HPH. Insets show the R improvement of PD compared to HPH against the size of the training set.

HPH. For example, when the length of recommendation list is 50, in *MovieLens* data, NBI can only recommend 293 distinct objects, HPH can recommend 787 distinct objects, while PD increases this number to more than 1000. In *Netflix* data, for the case $L = 50$, more than 5000 distinct objects can be recommended through PD algorithm, namely almost every object has the chance to be recommended. Secondly, the curves for NBI are remarkably steeper than those from HPH and PD. Taking the *MovieLens* data, for example (the case $L = 50$), with NBI algorithm, six movies are recommended over 600 times. Since there are only 943 users in this data set, it means that each of these movies is recommended to more than two-thirds of the users. The result with HPH is much better: the no. 1 object is recommended 341 times. However, compared to HPH (see the insets of Fig. 6), PD performs better, which indicates that with the PD algorithm users are more likely to be recommended different objects, namely, PD can provide more personalized recommendations.

Another metric to measure the algorithm's diversity is intrasimilarity. Different from Hamming distance, intrasimilarity measures the ability of an algorithm to provide diverse recommendations for a single user. The dependence of intrasimilarity on parameter ε is shown in Fig. 7. It shows that the parameter ε is positively correlated with intrasimilarity, namely the smaller ε, the lower the intrasimilarity (i.e., higher the intradiversity). Compared to NBI, when $L = 50$, intrasimilarity can be decreased by 21% for *MovieLens* and 23% for *Netflix* with optimal parameters corresponding to their respective lowest ranking scores. Even compared to the HPH algorithm, the improvement can reach up to 5% for both data sets. This claims that our method is effective to generate more fruitful recommendations. Furthermore, we investigate how the two parameters (ε, L) affect intrasimilarity. The intrasimilarity I in the (ε, L) plane for two data sets is shown in Fig. 8. The dashed line indicates the intrasimilarity of the system, which is obtained by averaging $s^o_{\alpha\beta}$ over all the object pairs. Thus the intrasimilarity as obtained from (ε, L) on the dashed line is equal to that of L randomly chosen objects from the system. The left region has lower intrasimilarity, while the right region has higher intrasimilarity. As a metaphor, one

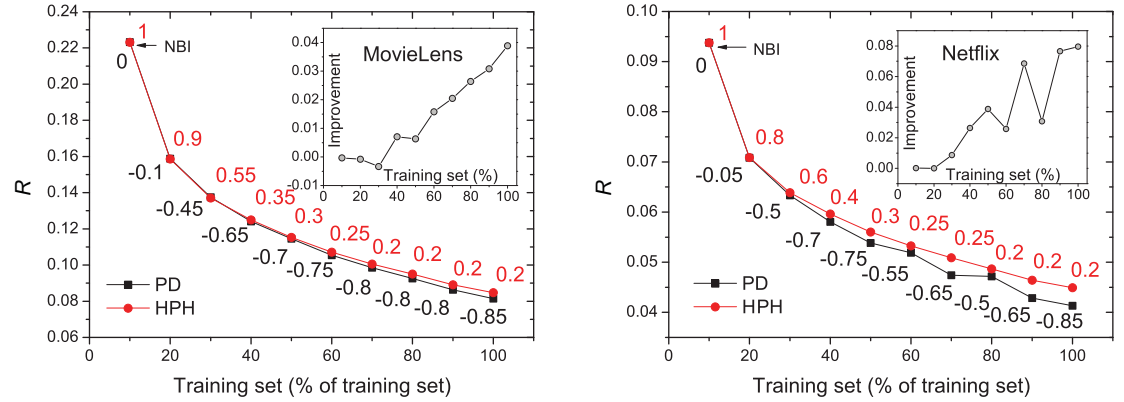

FIG. 11. (Color online) Ranking score changes with the size of the training set measured by the percentage of the 90% training set. That is to say, given a 90%–10% division of the training set and probe set, we randomly choose p% of the known links in the training set to predict the links in the unchanged probe set. Each data point is obtained with the parameter ($\varepsilon \in [-1,0]$ for PD and $\lambda \in [0,1]$ for HPH with step 0.05) subject to the lowest R. The optimal parameters are labeled in black for PD and red for HPH. Insets show the R improvement of PD compared to HPH against the size of the training set.

can think of the dashed line as a plane lens keeping the same size of the user's vision. In the left region, especially the area corresponding to smaller ε and larger L, the algorithm is like a concave lens that broadens the user's vision, while in the right region, corresponding to larger ε and smaller L, the algorithm is like a convex lens that narrows the user's vision. The focal length is determined by parameter ε. A smaller ε in the left region indicates a smaller focal length for the concave lens and hence a broader view, while in the right region it indicates a larger focal length for the convex lens and hence a narrow view.

In Fig. 9, we report the dependence of popularity on parameter ε. Similar to intrasimilarity, a smaller ε yields a smaller popularity P, and thus a more novel recommendation. Compared to the NBI, popularity can be remarkably improved by 33% and 23% for *MovieLens* and *Netflix* data sets. Even compared to the HPH algorithm, the improvement can reach 7% and 5%, respectively.

In real application, the computation complexity and memory space are crucial factors. It is obvious that any highly accurate recommendation algorithm will become meaningless if the consuming time or memory is unacceptable. After investigation, we find that our proposed algorithm is very efficient, especially for the large yet sparse network. On average, for one recommendation the time complexity of the PD algorithm is about $O[\min(N_u, \langle K_u\rangle^2\langle K_o\rangle)]$, where N_u is the number of users, and $\langle K_u\rangle$ and $\langle K_o\rangle$ respectively indicate the average degree of users and objects. Since the maximum value of $\min(N_u, \langle K_u\rangle\langle K_o\rangle)$ is N_u, in the worst case the complexity is approximate to $O(N_u\langle K_u\rangle) \approx O(|E|)$, where $|E|$ is the number of links (i.e., user-object pairs). The basic properties of the two data sets (consider the training sets) and the average computing time for one recommendation $\langle T\rangle$ with PD algorithm are shown in Table III. The computations were carried out on a desktop computer with an Intel Core 2 Duo 3.0 GHz CPU. From Table III, we can see that on *MovieLens* the real computing time is about 10 times faster than *Netflix*, which is approximately equal to the theoretical number, namely the ratio of their number of links ($634\,588/74\,249 = 8.5$). Moreover, for a network like *Netflix*, which is large yet sparse, the recommendation for a user only takes 2.6 ms. That is to say, in one second we can do recommendations for about 384 users. Besides the time complexity, the memory space is another limitation for algorithmic implementation for huge-size networks. For PD algorithm, the memory required is of the order $O(|E|)$, which is the minimum space needed to store the network topology.

V. EFFECTS OF DATA SPARSITY

In this section, we investigate the effects of data sparsity on the algorithmic performance. Since HPH is the most similar algorithm to our method, we choose it for comparison (although RENBI is more accurate, it considers the high-order correlations between objects). We investigate the effects of data sparsity on the algorithmic performance in two ways. (i) For the whole data set, we select $p\%$ (ranging from 10% to 90% with step 10%) links as the training set; the remaining $(100 - p)\%$ links constitute the probe set. Clearly, lower p indicates sparser data (i.e., less information). (ii) Given a 90%–10% division of the training set and probe set, we randomly choose $p\%$ of the known links in the prepared training set to predict the links in probe set. To do this, the probe links stay unchanged. For example, $p = 10$ means that we actually use 9% of the whole data set to predict the links in the probe set, which contains 10% links of the whole data set. Lower p indicates sparser data. The numerical results on two data sets are shown in Fig. 10 for method (i) and Fig. 11 for method (ii). Each point is obtained with the optimal parameter subject to the lowest ranking score. From Fig. 10, it can be seen that the ranking score decreases with the increasing size of the training set, which agrees with the intuition that we can obtain a better recommendation with more information. Furthermore, the optimal parameters of both methods decrease with increasing p for both methods. This shows that when the training set contains 10% links, the optimal parameters are $\lambda = 1$ for HPH and $\varepsilon = 0$ for PD, which are all corresponding to the standard case NBI. Insets show the R improvement (I^R) of PD compared to HPH, which is defined as

$$I^R = \frac{R^*_{\mathrm{HPH}} - R^*_{\mathrm{PD}}}{R^*_{\mathrm{HPH}}}, \tag{13}$$

where R^* indicates the lowest ranking score for a given training and probe set division. Generally speaking, the R improvement increases with the increasing size of the training set. That is to say, PD performs much better than HPH for denser data sets. The qualitative behaviors in Fig. 11 are the same as what we obtained in Fig. 10, which further demonstrates that PD can give much better predictions than HPH for denser data sets.

VI. CONCLUSION AND DISCUSSION

The preferential diffusion proposed in this paper is a kind of biased random walk, taking into account the heterogeneity of users' degrees. The present process indeed defines a new local index of similarity in bipartite networks (like the original NBI algorithm is corresponding to the so-called resource-allocation similarity index [26,27]) and thus it has potential applications in similarity-based link prediction [28,29], community detection [30], node classification [31], and so on. The biased random walk itself has already found extensive application in many branches of science and engineering, including detecting the navigation rules on a complex network [32], the design of routing strategy in transportation networks [33], quantifying the centrality of vertex and edge [34], modeling animal movements [35], and information discovery in wireless sensor networks [36]. Here we applied the biased random walk in dealing with the information filtering process, which may also broaden the understanding of the applicability of the biased random walk.

Accuracy metrics have been widely used to evaluate the performance of recommendation algorithms and considered to be the most important factor. For example, the Netflix Prize [37] focuses only on accuracy. However, user satisfaction does not always correlate with high recommendation accuracy [25,38]. The recommendations on popular objects (those are more easily found in other channels) are less likely to excite users. On the contrary, the unexpected and fortuitous recommendations, which are usually related with cold objects, are more favorable. Such a serendipitous recommendation will

improve user experience and thus enhance their loyalty to the system. In order to provide accurate as well as diverse and novel recommendations, in this paper, motivated by the perspective of physics, we proposed an algorithm, named PD, based on the preferential diffusion process on bipartite networks. We tested our algorithm on two benchmark data sets, *MovieLens* and *Netflix*, and applied five metrics, from the aspects of accuracy, diversity, and novelty, to evaluate the algorithmic performance. Compared to the standard algorithm NBI, the accuracy measured by ranking score can be further improved by 23% for *MovieLens* and 18% for *Netflix*. Even compared to the state-of-the-art algorithm, HPH, the improvement can reach 4% for *MovieLens* and 9% for *Netflix*. Moreover, the performance of PD can be further improved by considering a heterogenous initial resource configuration.

Furthermore, the statistical result on the ranking score of individual objects shows that our method has a much higher ability to accurately recommend the low-degree objects. That is to say, such prominent improvement on accuracy comes mainly from the highly accurate recommendation on unpopular objects, and thus it indeed enhances the recommendation diversity and novelty. For example, if we recommend 50 objects to each user, in *MovieLens*, NBI can only recommend 293 distinct objects to all users, HPH can recommend 787 distinct objects, while PD increases this number to more than 1000. In *Netflix* data, more than 5000 distinct objects can be recommended through PD algorithm; namely, almost every object has the chance to be recommended. Specially, we found that the recommender system may play different roles from the aspect of intrasimilarity—the similarity within a user's recommendation list, which is determined by the algorithm's parameter ε and the length of recommendation list L. Given (ε, L), if the intrasimilarity generated by the algorithm is higher than that of L randomly selected objects (i.e., average intrasimilarity of the whole system), the recommender system plays the role as a convex that narrows users' vision, whereas if intrasimilarity generated by the algorithm is lower than that of the system, the recommender system plays the role as a concave that broadens users' vision. Besides, we investigated the dependence of algorithm performance on data density. The results show that compared to HPH, PD algorithm gives more significant improvement for denser data.

A good recommendation algorithm can guide the system for a better development. You can think that the system itself and the recommendation algorithm constitute a symbiotic system. Generally speaking, there is no best recommendation algorithm, but the most suitable algorithm for a given system or a user. Just like the marriage game [39]: choose the right but not the best. In this sense, the most equitable evaluation on the recommendation algorithm should be based on the user experience, which is difficult to capture in metric. Notice that the optimal algorithm (or parameter) for the whole system is usually different from the optimal algorithm (or parameter) for an individual user. Thus an applicable and feasible way is building an open recommender system, where users can help themselves to find their best experienced algorithm (or parameter). For example, we can set a bar controlling the parameter of the algorithm on the website. Take the PD algorithm as an example; the user may set a large value of ε to obtain recommendations of popular and hot items, and set a small value of ε to obtain recommendations of niche and novel items. Here we argue that the design of user-centric recommender systems will become one of the challenges of the next generation information filtering techniques. Finally, we believe that this paper may shed some light on this interesting and exciting direction.

ACKNOWLEDGMENTS

We acknowledge the *GroupLens Research Group* for *MovieLens* data and *Netflix, Inc.* for *Netflix* data. We thank Yi-Cheng Zhang for providing the proper metaphor of *concave* and *convex* when referring to the user intrasimilarity, and Tao Zhou for a critical reading of the manuscript. This work is partially supported by the Swiss National Science Foundation under Grant No. 200020-132253 and the National Natural Science Foundation of China under Grants No. 60973069, No. 90924011, and No. 11075031.

[1] S. Brin and L. Page, Comput. Netw. ISDN Syst. **30**, 107 (1998).
[2] J. Kleinberg, J. ACM **46**, 604 (1999).
[3] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, Commun. ACM **35**, 61 (1992).
[4] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, Lect. Notes Comput. Sci. **4321**, 291 (2007).
[5] M.-S. Shang, L. Lü, W. Zeng, Y.-C. Zhang, and T. Zhou, Europhys. Lett. **88**, 68008 (2009).
[6] M. J. Pazzani and D. Billsus, Lect. Notes Comput. Sci. **4321**, 325 (2007).
[7] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, Inf. Retr. **4**, 133 (2001).
[8] S. Maslov and Y.-C. Zhang, Phys. Rev. Lett. **87**, 248701 (2001).
[9] P. Laureti, L. Moret, Y.-C. Zhang, and Y.-K. Yu, Europhys. Lett. **75**, 1006 (2006).
[10] J. Ren, T. Zhou, and Y.-C. Zhang, Europhys. Lett. **82**, 58007 (2008).
[11] S. M. McNee, J. Riedl, and J. A. Konstan, *Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems* (ACM Press, New York, 2005), p. 1097.
[12] C. Cattuto, V. Loreto, and L. Pietronero, Proc. Natl. Acad. Sci. U.S.A. **104**, 1461 (2007).
[13] Z.-K. Zhang, T. Zhou, and Y.-C. Zhang, Physica A **389**, 179 (2010).
[14] Z.-K. Zhang, C. Liu, Y.-C. Zhang, and T. Zhou, Europhys. Lett. **92**, 28002 (2010).
[15] W. Zeng, M.-S. Shang, Q.-M. Zhang, L. Lü, and T. Zhou, Int. J. Mod. Phys. C **21**, 1217 (2010).
[16] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang, Phys. Rev. E **76**, 046115 (2007).
[17] Y.-C. Zhang, M. Medo, J. Ren, T. Zhou, T. Li, and F. Yang, Europhys. Lett. **80**, 68003 (2007).
[18] Y.-C. Zhang, M. Blattner, and Y.-K. Yu, Phys. Rev. Lett. **99**, 154301 (2007).

[19] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang, Proc. Natl. Acad. Sci. U.S.A. **107**, 4511 (2010).
[20] W. Liu and L. Lü, Europhys. Lett. **89**, 58007 (2010).
[21] T. Zhou, L.-L. Jiang, R.-Q. Su, and Y.-C. Zhang, Europhys. Lett. **81**, 58004 (2008).
[22] T. Zhou, R.-Q. Su, R.-R. Liu, L.-L. Jiang, B.-H. Wang, and Y.-C. Zhang, New J. Phys. **11**, 123008 (2009).
[23] J. L. Herlocker, J. A. Konstan, K. Terveen, and J. T. Riedl, ACM Trans. Inform. Syst. **22**, 5 (2004).
[24] M.-S. Shang, L. Lü, Y.-C. Zhang, and T. Zhou, Europhys. Lett. **90**, 48006 (2010).
[25] C. N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, *Proceedings of the 14th International World Wide Web Conference (WWW2005)* (ACM Press, New York, 2005), p. 22.
[26] T. Zhou, L. Lü, and Y.-C. Zhang, Eur. Phys. J. B **71**, 623 (2009).
[27] L. Lü, C.-H. Jin, and T. Zhou, Phys. Rev. E **80**, 046122 (2009).
[28] D. Liben-Nowell and J. Kleinberg, J. Am. Soc. Inform. Sci. Technol. **58**, 1019 (2007).
[29] L. Lü and T. Zhou, Physica A **390**, 1150 (2011).
[30] Y. Pan, D.-H. Li, J.-G. Liu, and J.-Z. Liang, Physica A **389**, 2849 (2010).
[31] Q.-M. Zhang, M.-S. Shang, and L. Lü, Int. J. Mod. Phys. C **21**, 813 (2010).
[32] A. Fronczak and P. Fronczak, Phys. Rev. E **80**, 016107 (2009).
[33] W.-X. Wang, B.-H. Wang, C.-Y. Yin, Y.-B. Xie, and T. Zhou, Phys. Rev. E **73**, 026111 (2006).
[34] S. Lee, S. H. Yook, and Y. Kim, Eur. Phys. J. B **68**, 277 (2009).
[35] E. A. Codling, R. N. Bearon, and G. J. Thorn, Ecology **91**, 3106 (2010).
[36] K. K. Rachuri and C. S. R. Murthy, *Proceedings of the 2009 IEEE International Conference on Communications* (IEEE Press, Piscataway, NJ, 2009), p. 5035.
[37] J. Bennett and S. Lanning, *Proceedings of KDD Cup and Workshop 2007* (ACM Press, New York, 2005), p. 3.
[38] S. M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl, *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work* (ACM Press, New York, 2002), p. 116.
[39] M. J. Oméro, M. Dzierzawa, M. Marsili, and Y.-C. Zhang, J. Phys. I **7**, 1723 (1997).