Future Generation Computer Systems 88 (2018) 262–270

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A collaborative filtering recommendation method based on discrete quantum-inspired shuffled frog leaping algorithms in social networks[☆]

Wenjuan Li [a,b,c,*], Jian Cao [b], Jiyi Wu [d], Changqin Huang [e], Rajkumar Buyya [c]

[a] Qianjiang College, Hangzhou Normal University, Hangzhou 310036, China
[b] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[c] Cloud Computing and Distributed Systems (CLOUDS) Laboratory at University of Melbourne, Australia
[d] Key Lab of E-Business, Hangzhou Normal University, Hangzhou 310036, China
[e] School of Information Technology in Education, South China Normal University, Guangzhou 510631, China

HIGHLIGHTS

- A Discrete Quantum-inspired Shuffled Frog Leaping algorithm (DQSFL) is proposed.
- The shuffled frog leaping algorithm and quantum information theory are combined.
- CF recommendation method based on DQSFL improves accuracy of rating score prediction.

ARTICLE INFO

Article history:
Received 4 January 2018
Received in revised form 29 April 2018
Accepted 27 May 2018
Available online 6 June 2018

Keywords:
Social network
Collaborative filtering recommendation
Shuffled frog leaping algorithm
Quantum information theory
Rating score prediction accuracy

ABSTRACT

In social network recommendation systems, the rating score prediction accuracy of the collaborative filtering (CF) method depends on both the extraction of the nearest neighbors and the calculation of user/project similarity. Based on a similar principle to user/project behavior, this paper uses the maximum intersection method to extract the optimal neighbor candidate set, and presents a weighted adjusted cosine similarity method to compute user/project similarity. Furthermore, to optimize the weights of the method, a novel optimization method called the discrete quantum-inspired shuffled frog leaping (DQSFL) algorithm is proposed, which is based on the shuffled frog leaping algorithm and quantum information theory. The DQSFL algorithm uses quantum movement equations to search for the optimal location according to the co-evolution of the quantum frog colony. The experiments demonstrate that the CF recommendation method based on DQSFL can effectively solve the rating data sparseness problem in the similarity computation process to improve the accuracy of the rating score prediction, and provide a better recommended result than traditional CF algorithms.

1. Introduction

With the continuous development of Internet applications, data resources on the Web grow explosively every day, which causes a serious contradiction between user requirements and the huge amount of complex data resources. In order to solve these contradictions, a personalized recommendation system is proposed to extract useful information from vast amounts of information for users, which has been used for a wide range of applications. Personalized recommendation technology recommends various resources to users by studying user behaviors and interests, and was initially applied in e-commerce personalized service. With the fast development of social networks, recommendation systems are also widely used in social networks [1]. Different from traditional recommendation systems that use content filtering and then analyze the results directly for recommendation, collaborative filtering (CF) algorithms first analyze the user's interests and behaviors, and search for users who are similar to the current user, and then recommend projects to the current user by analyzing the rating scores of similar users on different projects and predict the rating

☆ This work was supported by National Natural Science Foundation of China under Grant 61370229, 61472253 and 61702151, the funding for visiting scholar from China Scholarship Council No. 201709645006 and the Research Project from Department of Education of Zhejiang Province No. Y201635438, the S&T Projects of Guangdong Province No. 2015A030401087.

* Corresponding author at: Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China.
E-mail addresses: liellie@163.com (W. Li), cao-jian@sjtu.edu.cn (J. Cao), CloudLab@aliyun.com (J. Wu), cqhuang@zju.edu.cn (C. Huang), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.future.2018.05.070
0167-739X/© 2018 Elsevier B.V. All rights reserved.

scores for these projects. CF is currently one of the most successful technologies and has been widely used in commercial personalized recommendation systems [2].

Goldberg et al. [3] first proposed a recommendation system based on the basic idea of the CF algorithm, the principle on which it is based being as follows: compare the current user's behaviors or preferences with those of other users, and find users whose behaviors and preferences are similar or identical to the current user, and then recommend products with high predicted scores to the current user based on the scores from similar users. In a certain sense, CF recommendation systems are attributed to a rating score prediction process. There are two key issues in the rating score prediction process, the extraction of a nearest neighbor candidate set and the calculation of user similarity. The extraction of the nearest neighbor candidate set is an important factor which directly affects the accuracy of the rating score prediction. If the extraction of the nearest neighbor candidate set is improper, it will result in the candidate set comprising users who are not the current user's neighbors which reduces the accuracy of the similarity calculation and leads to lower efficiency rating score prediction. The latter is a common problem existing in CF algorithms, which is called the rating data sparseness problem [4]. The maximum intersection method allows the selected collection of the nearest neighbors to have the most common user rating with the current project, thus avoiding the rating data sparseness problems in the direct computation of user similarity. Hence, this paper uses the method to extract neighbors.

User similarity can be calculated by different metrics, such as relevant similarity, cosine similarity, adjusted cosine similarity and Euclidean distance [5]. Different calculation methods directly affect the accuracy of rating score prediction. In order to improve prediction accuracy, this paper proposes a new similarity calculation method called weighted correction cosine similarity (WCCS), and designs a novel modified optimization algorithm to optimize the weights in WCCS.

In recent years, many intelligent optimization algorithms, such as particle swarm optimization (PSO) [6], genetic optimization [7] and artificial immune network [8], have been introduced to optimize the CF algorithms used for recommendation systems. However, all the optimization algorithms have the dimension disaster problem when used to solve multiple-dimensional discrete optimization problems, which lead to a slow convergence rate and poor performance. Compared to traditional optimization algorithms, the shuffled frog leaping (SFL) algorithm, which has increased in popularity in recent years, has a faster convergence rate in solving continuous optimization problems. Therefore, in this paper, we propose a novel modified SFL algorithm called the discrete quantum-inspired shuffled frog leaping (DQSFL) algorithm, which introduces quantum computing into the SFL algorithm for quantization and discretization in order to optimize the weights of CF algorithms effectively. The experimental results demonstrate that the proposed algorithm can effectively improve prediction accuracy.

The contributions of this paper are as follows: (1) it combines the shuffled frog leaping algorithm and quantum information theory with the optimal location; and (2) it improves the rating score prediction accuracy of the CF method.

The rest of the paper is organized as follows. Section 2 describes the related work, such as the CF recommendation method, the SFL algorithm and quantum computing. The DQSFL algorithm is proposed and detailed in Section 3. Section 4 provides details on the proposed CF algorithm. The experiment and analysis of the proposed CF algorithm are provided in Section 5. Finally, we conclude the paper in Section 6.

2. Related work

2.1. CF recommendation method

The Collaborative Filtering (CF) method, originated in the 1990s and has undergone 20 years of research and application. Depending on the different objects used for the CF method, Sarwar et al. divide CF systems into user-based CF systems and project-based CF systems [2]. The user-based CF method makes recommendations for users according to user similarity on the assumption that associated relationships exists between users with similar behaviors or preferences, while the project-based CF method provides recommendations according to the correlation by assuming that there must be correlation relationship between projects. Breese et al. divides CF systems into model-based CF systems and memory-based CF systems by considering the different algorithms used for CF systems [9]. A model-based CF system learns and trains related complicated data collaborative models using data mining algorithms or statistical methods which will be applied in practical data prediction, whereas a memory-based CF system calculates the similarity between two users/projects or weights for computing scores, and then selects the best users/projects with the highest scores. In addition, for a memory-based CF system, similarity calculation is important, where the main similarity metrics have a Pearson correlation coefficient, cosine similarity and adjusted cosine similarity [10–12]. The rating score prediction of a CF recommendation system has two purposes as follows: first, to determine the recommended object by predicting the scores; and second, to ease score matrix sparsity. The most common method for rating score prediction is the weighted average method, which predicts scores by calculating the similarities of multiple neighbor users/projects. However, the weighted average method suffers from the rating data sparseness problem in the process of determining nearest neighbors.

Zhang et al. use BP neural networks to predict the score of the user/project to reduce the data sparsity of the nearest neighbor candidate set [13]. This algorithm avoids the disadvantages existing in the dimensional reduction method and the Intelligent Agent method, however, it needs a long time to train the model, and the status of the model obtained from the algorithm is not always stable, which has a negative impact on the accuracy of the prediction score results.

Generally speaking, the most common method for solving the rating data sparseness problem includes clustering analysis and matrix decomposition. Cluster analysis restricts the search area of the nearest neighbor candidate set in the nearest cluster and extracts the recommended result by using a cluster centroid. The latter method improves the recommendation speed, but reduces the recommendation quality, which does not fundamentally solve the problem. Some matrix decomposition methods were proposed in [14,15], whereas Kim solved the rating data sparseness problem by using the multi-level association rule mining method in [16].

In this paper, we use the maximum intersection method to extract the nearest neighbors based on the project-based CF algorithm when using a user-project score matrix for rating score prediction, which can partially solve the rating data sparseness problem.

2.2. Shuffled frog leaping algorithm and quantum computing

In 2003, Eusuff et al. first proposed a novel bionic intelligent optimization algorithm called SFLA in [17], which combines a memetic algorithm based on meme evolution and particle swarm optimization based on population behaviors. This algorithm is based on a simple concept, less parameters, higher calculation speed and stronger global search optimization capability. Luo et al. proved the global convergence of SFLA by solving the differential equation to analyze the trajectory of SFLA [18]. Elbeltagi et al. [19] made a comparison of SFLA with a genetic algorithm (GA), particle swarm algorithm, and ant colony algorithm, and then analyzed their optimization performance. SFLA is mainly used to solve the continuous multi-objective optimization problem, such as water resource allocation, piers maintenance, workshop operation process arrangements and other practical application problems. Hence, we use SFLA discretization for optimizing the weights of CF algorithms in recommendation systems.

Quantum computing with its unique computational performance has caused widespread interest in the scientific world. In 1996, Narayanan proposed a quantum-inspired genetic algorithm (QGA), and was the first to introduce quantum computing into optimization algorithms, fusing quantum computing and evolutionary computing research [20]. Subsequently, the results of world-wide research include the quantum-inspired ant colony algorithm, the quantum-inspired simulated annealing algorithm, and the quantum-inspired immune algorithm and so on [21–23]. Taking the quantum-inspired genetic algorithm as an example, it employs a multi-state qubit to encode genes and uses the quantum rotation gate to realize chromosome evolution while introducing the dynamic adjustment mechanism used for rotation angle and quantum crossover into the optimization process, which achieves better performance than conventional genetic algorithms.

The quantum evolutionary algorithm breaks through the classic information system limitations with a strong global optimization ability, but poor convergence capability. With the advantages of being simple, fast and easy for realization, it is possible for SFLA to accelerate the quantum evolutionary algorithm convergence rate. In addition, the quantum evolutionary algorithm is likely to address the key issues in SFLA research, such as dividing the population structure, information sharing among populations, etc. If the two algorithms are combined, high efficiency can be maintained on the basis of obtaining better optimization results. At present, the combination of the two algorithms is still relatively rare. Zhang [24] proposed the use of the quantum SFLA in neural network optimization and Ding [25] introduced a rapid rough property reduction algorithm based on quantum leapfrog coevolution.

Because the optimization of the weights of the CF algorithm is a discrete optimization problem, we propose a novel optimization algorithm called quantum-inspired shuffled frog leaping (DQSFL) algorithm, which introduces quantum computing into the SFL algorithm for quantization and discretization in order to optimize the weights of CF algorithms effectively.

3. Discrete Quantum-inspired Shuffled Frog Leaping algorithm (DQSFL)

3.1. Quantum position of DQSFL

The quantum position of a quantum frog can be represented by a string of qubits, and the quantum position of quantum frog i is defined as

$$v_i = [v_{i1}, v_{i2}, \dots, v_{il}] = \begin{bmatrix} \alpha_{i1} & \alpha_{i2} & \dots & \alpha_{il} \\ \beta_{i1} & \beta_{i2} & \dots & \beta_{il} \end{bmatrix} \tag{1}$$

Therein (1), qubit is the minimum information unit, which satisfies

$$|\alpha_{ij}|^2 + |\beta_{ij}|^2 = 1, j = 1, 2, \dots, l. \tag{2}$$

Furthermore, to make the DQSFL algorithm more concise and efficient, we define the qubits α_{ij} and β_{ij} in the region $0 \le \alpha_{ij} \le 1$ and $0 \le \beta_{ij} \le 1$ respectively, and the evolution of quantum frog populations is assumed as the update process of the quantum position. Hence, we define the quantum rotation angle as $\Delta\theta_{ij}^{t+1}$, and quantum rotation gate $U\left(\Delta\theta_{ij}^{t+1}\right)$ can be defined as

$$U\left(\Delta\theta_{ij}^{t+1}\right) = \begin{bmatrix} \cos\Delta\theta_{ij}^{t+1} & -\sin\Delta\theta_{ij}^{t+1} \\ \sin\Delta\theta_{ij}^{t+1} & \cos\Delta\theta_{ij}^{t+1} \end{bmatrix} \tag{3}$$

The qubit j of quantum frog i is $v_{ij}^t = [\alpha_{ij}^t, \beta_{ij}^t]^T$, which is updated according to the quantum rotation gate $U(\theta_{ij}^{t+1})$. Therefore, there is an update process as follows:

$$\begin{aligned} v_{ij}^{t+1} &= abs\left(\mathrm{U}\left(\Delta\theta_{ij}^{t+1}\right) v_{ij}^t\right) \\ &= abs\left(\begin{bmatrix} \cos\Delta\theta_{ij}^{t+1} & -\sin\Delta\theta_{ij}^{t+1} \\ \sin\Delta\theta_{ij}^{t+1} & \cos\Delta\theta_{ij}^{t+1} \end{bmatrix} v_{ij}^t\right) \end{aligned} \tag{4}$$

If quantum rotation angle θ_{ij}^{t+1} is zero, we use quantum NOT gate W to update the qubit v_{ij}^t in some small probability, where

$$v_{ij}^{t+1} = W v_{ij}^t = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} v_{ij}^t \tag{5}$$

3.2. DQSFL algorithm

In the DQSFL algorithm, the quantum frog group first produces the quantum positions and common positions of h quantum frogs respectively, and both quantum and common positions have l dimensions. The position set of each quantum frog can be represented by $\mathrm{x} = \{x_1, x_2, \dots, x_h\}$. After t iterations, the quantum position of quantum frog i is $v_i^t = \left[v_{i1}^t, v_{i2}^t, \dots, v_{il}^t\right]$, and its common position can be obtained from the relevant quantum position, which has $x_i^t = \left[x_{i1}^t, x_{i2}^t, \dots, x_{il}^t\right], i = 1, 2, \dots, h$. Each quantum frog updates its quantum position by the guidelines of the global extreme value and subgroup extreme value, and then determines its own common position according to the measurement of the quantum states. The local optimal value of quantum frog i is noted as $p_i^t = \left[p_{i1}^t, p_{i2}^t, \dots, p_{il}^t\right], i = 1, 2, \dots, h$, which is the best position of the quantum frog. The global extreme value can be noted as $p_g^t = \left[p_{g1}^t, p_{g2}^t, \dots, p_{gl}^t\right]$, which is the optimal solution of all the local extreme values after t iterations. Let quantum frogs be in descending order by the fitness values related to their local extreme values, and be divided in to q $(q = \frac{h}{r})$ subgroups. In frog subgroup $k(k = 1, 2, .., q)$, the optimal value of the subgroup is $p_k^t = \left[p_{k1}^t, p_{k2}^t, \dots, p_{kl}^t\right]$. The updating functions of the quantum rotation angle and qubit position for the worst quantum frog i can be given by

$$\Delta\theta_{ij}^{t+1} = e_1\left(p_{kj}^t - x_{ij}^t\right) + e_2\left(p_{gj}^t - x_{ij}^t\right) \tag{6}$$

$$v_{ij}^{t+1} = \begin{cases} W v_{ij}^t, & p_{kj}^t = x_{ij}^t = p_{gj}^t \text{ and } \eta_{ij}^t < c_1 \\ abs\left(\mathrm{U}\left(\Delta\theta_{ij}^{t+1}\right) v_{ij}^t\right) & else \end{cases} \tag{7}$$

Therein (6) and (7), $j = 1, 2, \dots, l$, and e_1, e_2 denote leap step length, which represents the impact of the global optimal value and subgroup optimal value on the quantum rotation angle and $c_1 \le 1/l$ denotes the qubit mutation probability that the quantum rotation angle is zero. The leap length needs to be carefully designed. Because a large leap step length may result in premature convergence, while a small one may slow down the convergence speed. Therefore we did a number of experiments to compare and select a better leap step length. In addition, introducing a new leap equation into the iterative equation improves information exchange among populations in order to determine the quantum

Table 1
Example of standard project configuration file.

1	2	3	4	6	7
Rating	Age	Gender	Occupation	Date	Project genre
4	24	1	47	1990-1-2	10

rotation angle and position. According to quantum position tests, the common position of the quantum frog can be given by

$$x_{ij}^{t+1} = \begin{cases} 1 & \gamma_{ij}^{t+1} > (\alpha_{ij}^{t+1})^2 \\ 0 & \gamma_{ij}^{t+1} \le (\alpha_{ij}^{t+1})^2 \end{cases} \tag{8}$$

Therein (8), γ_{ij}^{t+1} is a random number that satisfies a uniform distribution between [0,1], and α_{ij}^{t+1} is the probability of the quantum state v_{ij}^{t+1} being zero. In this paper, the quantum position of the worst quantum frog in each subgroup is updated according to Eqs. (6) and (7). If the fitness of the worst frog is improved, replace the original extreme with the new position, otherwise it remains unchanged.

4. CF recommendation method based on DQSFL

To improve the quality of the recommendation system, this section presents one novel optimization method for recommendation systems based on DQSFL, which is at the basis of the related principle of the CF algorithm. The major procedures in the rating score prediction algorithm are described in the following subsection.

4.1. Establish project configuration

Project configuration files are the basis of calculating the distance and similarity between projects. After establishing the project configuration files, it needs the nearest neighbor set to calculate the similarity relationship of projects. Let *profile*(j) denote the configuration file of project j, and each file has K rows which represent K characteristics of the configuration files. Each project configuration contains two parts: (1) the variable part, which comprises score of user characteristics such as career, gender, age and each user score for this project; and (2) the fixed part, which comprises the name, date and types of this project. Let *profile*(j, i) denote the configuration file of user i for project j. An example of a standard project configuration file is illustrated in Table 1.

The first column in Table 1 shows the project score, and the second shows age, which can be standardized as

$$\frac{age(i) - \min(age)}{\max(age) - \min(age)} \tag{9}$$

The third and four column represent the gender and occupation of the user respectively. The last two column are the attributes of the project. In this paper, all the variables in the files have been normalized by using (9).

4.2. Selecting the nearest neighbor candidate set and computing neighbor similarity

In this subsection, selecting the nearest neighbor set involves the following: (1) select the nearest neighbor set by using the maximum intersection method; (2) determine the weighted adjusted cosine similarity by using the DQSFL optimization algorithm; (3) to maintain the efficiency and timeliness of the recommendation system, model the project module and user module by offline computing and store the related data results.

First, we select the project configuration file candidate set based on the principle of user behavior similarity, assuming that there are more evaluations from a certain common user for the same project, which demonstrates that these two users have more similarity in their behaviors. Hence, rank the neighbors who have common user scores with the current project in descending order, and select the former n neighbors as the candidate set, which is represented by the following equation:

$$S_{i,j} = R\,(i, :) * R(j, :)' \tag{10}$$

Here $S_{i,j}$ represents the size of the intersection of project i and j, $R\,(i, :)$ and $R\,(j, :)$ represent the score vectors of project i and j. Since the higher $S_{i,j}$, the more common user scoring items the two projects have, we can use it to select the most suitable nearest neighbors. After we sort the scores in descending order, the highest scored n projects are chosen to the candidate set.

After selecting the nearest neighbor candidate set, we compute the weighted adjusted cosine similarity. The similarity between two projects/users not only depends on user scores, but is also affected by user characteristics. Therefore, a similarity calculation needs to consider the combination of user information and project information. Taking into account the different influence degree of each characteristic factor on scores, we add a weight for each characteristic, and use the DQSFL algorithm to optimize this weight to obtain the optimal solution.

Since the probability magnitude of a qubit in a two-bit quantum system has the property $\left|\alpha_{ij}\right|^2 + \left|\beta_{ij}\right|^2 = 1$. This paper takes $\left|\alpha_{ij}\right|^2$ directly as the relative weight of the first attribute in the project configuration file, and $\left|\beta_{ij}\right|^2$ as the relative weight of another attributes. When each quantum position of the quantum frog is K-bit encoded (K is equal to the number of attributes). Through the relative values of the weights, the actual weight of each attribute can be obtained by the equation $w_1 + w_2 + .. + w_k = 1$. Substituting the actual weights into the similarity calculation formula, the final fitness value is obtained.

Assume $W(A)$ represents a potential solution for characteristic weights of the current project configuration files, and A denotes a weight set which contains K weight values, which implies the quantum positions of a quantum frog with K qubits. When the quantum frogs leap, these characteristics are continuously adjusted until the optimal weight set is found. Here, we use the modified cosine function to compare the similarity and distance between project configuration files. Let *similarity*(A, j) denote the similarity between the current project A and neighbor project j, and is defined as:

$$similarity\,(A, j) = \sum_{i=1}^{n} \frac{AW \cdot jW}{\|AW\|\,\|jW\|} \tag{11}$$

In (11), W is a $K \times K$ matrix, and each value of the matrix represents the weight value of each characteristic, and n denotes the number of users with common scores between the two projects.

After calculating *similarity* (A, j), we store the nearest cosine distance obtained from the project configuration file in the system. Here, we use the offline method of storing cosine distances for rating score prediction. We only use these distances when rating score prediction begins, and only update these similarity values for a certain time. However, if the recommendation system has very frequent behaviors of user scoring, we also can use the online method to store and employ the related similarity values.

4.3. Weight optimization based on DQSFL

In this subsection, we use the DQSFL algorithm to optimize the characteristic weights of the current project configuration file in order to find the optimal characteristics for the project.

(1) Quantum frog. As described previously, the quantum position of each quantum frog represents a potential solution set consisting of K qubits. The frogs search for the optimal solution

by leaping continuously for multiple iterations. When the position of one particular frog results in the fitness function having the optimal value, the quantum position of this frog will be the global optimal position for all frogs.

(2) Fitness function. In this experiment, assuming all the characteristic weights of each project are stable, a single fitness function exists as a result of only one optimal solution existing in the current project. We define this fitness function as the average prediction error between the rating score predictions for the two projects. Hence, the expression of the fitness function is given by

$$fitness = \frac{1}{n}\sum_{i=1}^{n}\left\{\left(mean_A + k\sum_{i=1}^{m} similar(A,j) \times \left(vote(j,i) - mean_i\right)\right) - vote(A,i)\right\} \quad (12)$$

Where n denotes the number of users with common scores between the current project A and the neighbor project j. Let $mean_A$ denote the average score for the current project A, and $similarity(A, j)$ denotes the similarity of the current project A and the neighbor project$j$. Let $vote(j, i)$ represent the score of project j from user i, $mean_i$ denotes the average score from user i, and $vote(A, i)$ represents the actual score of project A from user i.

(3) The major procedures of the DQSFL algorithm.

Step 1. Initialize the quantum frog population. Use equal-magnitude to initialize the quantum positions of h quantum frogs. The initial value of all probability magnitudes $(2*h*l)$ is $1/\sqrt{2}$ and v_i^1 is the initial quantum position of the ith frog. Here l represents the solution space dimension which is the number of attributes used to compute the similarity here. The measurement state of v_i^t is the normal position of the quantum frog x_i^t, which is also the possible solution to the optimization. Test the common positions of these frogs to derive the initial local extreme value p_i^1.

Step 2. Compute the fitness values corresponding to the positions of quantum frogs. Rank the quantum frogs in the descending order of fitness value, and then divide them into q subgroups. Each subgroup contains r quantum frogs that satisfies $h = q * r$. The division method is: the ith frog is divided into the kth subgroup that satisfies $k = i\%q$. The current optimal position found by the kth frog subgroup is denoted by p_k^t and the global optimal solution is p_g^t.

Step 3. Update the quantum position of the worst quantum frog in each subgroup according to Eqs. (6) and (7).

Step 4. Test the quantum position v_i^t of each quantum frog to obtain the position x_i^t according to Eq. (8).

Step 5. Calculate the fitness values of each frog. If the new position of a certain quantum frog x_i^{t+1} is better than p_i^t, then the local optimal value is replaced by x_i^{t+1}, that is $p_i^{t+1} = x_i^{t+1}$; otherwise $p_i^{t+1} = p_i^t$ remains unchanged. Each subgroup completes φ times meta-evolutions.

Step 6. Quantum frogs are regrouped according to the newly obtained fitness value. Update the optimal position of each subgroup p_k^t and the global optimal value p_g^t of the whole frog group. Repeat step 3∼5.

Step 7. Determine whether the convergence condition is satisfied or whether the maximum iterations has been reached. If not, increase the number of iterations by one and repeat step 6. If yes, terminate the algorithm and output the global optimal position of the quantum frogs.

This paper uses the square of the probability amplitude ($\left|\alpha_{ij}\right|^2$ or $\left|\beta_{ij}\right|^2$) as the relative weight of the attribute, therefore, it is only necessary to convert the relative weight into the absolute weight to calculate the similarity. The workload of initializing and updating the positions of all the quantum frogs is $h*l$ or $h*K$ (h denotes the number of the quantum frogs, l is the qubits of a quantum frog and K indicates the number of attributes, l and K are equal). However, in each iteration of a subgroup, only the l qubits of the worst frog needs to be updated, so the workload is l. The worst case is that the maximum number of meta-evolution iterations φ is reached, so the workload of updating subgroup is $\varphi * l$. The ranking of fitness needs $h*\log h$. Assume the maximum global iterations is set to F, the total time consumption of DQSFL is $h * l + F * h * \log h + F * q * (\varphi * l)$. By setting the proper value of the population of quantum frogs, the max global iterations F and meta-evolution iterations φ, the time overhead of the algorithm can be controlled.

4.4. Rating score prediction and recommend

After obtaining the optimal weights of the configuration files of the nearest neighbor project and the current project by using DQSFL algorithm, we start to predict the scores.

(1) Rating score prediction. The expression of rating score prediction is defined as Eq. (13). Let n be the number of selected neighbors, k is the standard parameter, and the sum of the cosine distances is 1. The other major parameters in this expression are illustrated in Table 2.

$$P\_vote(A,i) = mean_A + k\sum_{j=1}^{n}(similarity(A,j) \times (vote(j,i) - mean_i)) \quad (13)$$

Because the scores of the current project have already known, the accuracy of the prediction result can be represented by Mean Absolute Error (MAE), which can be calculated as

$$MAE = \frac{1}{n}\sum_{m=1}^{n}\left|Y_{im} - S_{im}\right| \quad (14)$$

Where Y_{im} denotes the predicted score and S_{im} denotes the actual score.

(2) Recommend. The recommendation methods of recommendation systems are different due to the use of different recommendation algorithms. There are two main methods as follows: first, recommend interesting projects to the current user when extracting the nearest neighbors with similar interests; second, extract the nearest neighbors, and make a rating score prediction for the current project and user, and then rank the predicted scores in order to recommend the ones with higher scores.

5. Performance evaluation

In this section, we firstly test and analyze the performance of the DQSFL algorithm and compare it with other traditional optimization algorithms such as GA, QGA and PSO. Furthermore, we conduct experiments to measure the performance of the recommendation system using the CF algorithm based on DQSFL, and compare it with other traditional algorithms.

5.1. Testing and analyzing DQSFL

(1) Test Design. Function optimization is a classic application of artificial intelligence algorithms and is also one of the commonly used evaluation methods. In order to test convergence speed and capacity, we select two classic benchmark functions to test the minimum extreme value for the DQSFL algorithm. The potential solution is encoded with binary information, the bit length of which is 20. The number of simulations is set to 200, and the population in all the intelligent optimization algorithms is set to 25. The simulation result is computed by the statistical average method. The parameter settings of the other traditional optimization algorithms, such as GA, QGA and PSO, can be found in [6,20]. In

Table 2
The Notations of major parameters in (13).

Symbol	Explanation
$P\_vote(A, i)$	the predicted score for the current project
$mean_A$	the average score for current project A
$vote(A, i)$	the configuration file of user i scoring for project A
$similar(A, j)$	the nearest cosine similarity value of the configuration files of project A and j
$vote(j, i)$	the score of project j from user i
$mean_i$	the average score of all the projects from user i

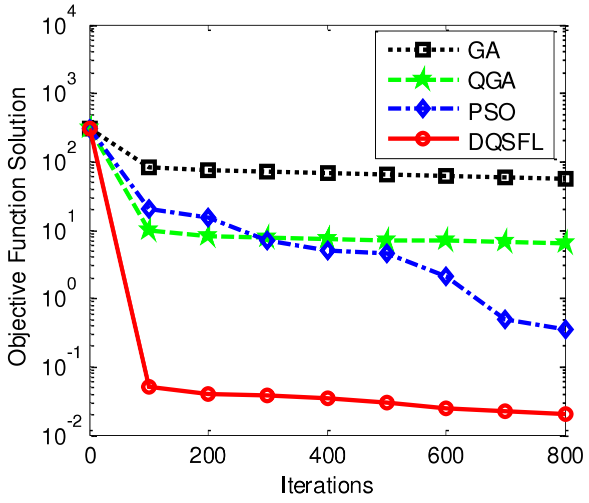


Fig. 1. Convergence curve of Schwefel function.

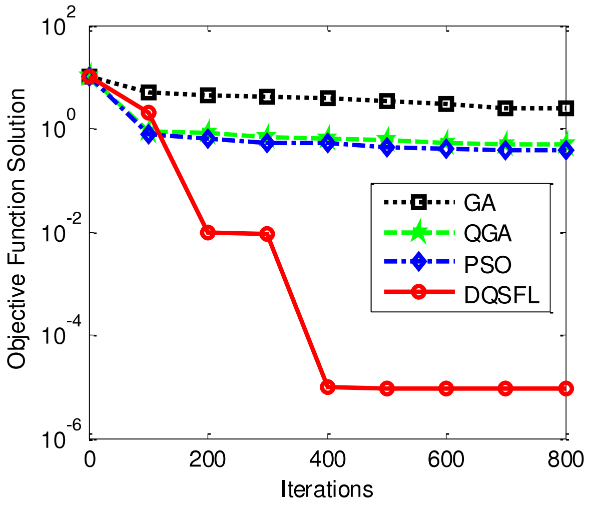


Fig. 2. Convergence curve of Rastrigin function.

DQSFL, we define the fitness function as the optimization function, and the position which corresponds with the minimum extreme value of the fitness function is the optimal solution. The major parameter settings are set as follows: there are 4 frog subgroups in the frog population: two subgroups have $e_1 = 0.05$ and $e_2 = 0.04$; and the other two subgroups have $e_1 = 0.05\ \pi$ and $e_2 = 0.04\ \pi$; and all the subgroups have$c_1 = c_2 = 0.1/l$, where l is the dimension number of the solution space.

Table 3
Comparison of Schwefel function test.

Algorithm	Optimal value	Execution time (s)
DQSFL	1.806×10^{-2}	2.96
PSO	3.951×10^{-1}	3.87
QGA	8.669×10^{0}	3.05
GA	4.595×10^{1}	4.27

Table 4
Comparison of Rastrigin function test.

Algorithm	Optimal value	Execution time (s)
DQSFL	8.157×10^{-6}	3.15
PSO	5.557×10^{-1}	3.34
QGA	7.932×10^{-1}	2.98
GA	2.486×10^{1}	3.52

An example of the classic Schwefel function is:

$$F_1(y) = 2 \times 418.9828 + \sum_{i=1}^{2} -y_i \sin\left(\sqrt{|y_i|}\right), \quad -500 \le y_i \le 500, i = 1, 2. \tag{15}$$

An example of the classic Rastrigin function is:

$$F_2(y) = \sum_{i=1}^{2} (y_i^2 - 10\cos(2\pi y_i) + 10), \quad -5.12 \le y_i \le 5.12, i = 1, 2. \tag{16}$$

(2) Result Analysis. Both the classic Schwefel function and Rastrigin function are complex multimodal functions with a large number of local minimum extreme values, hence it is easy to make the search fall into the local optimal solution. Figs. 1 and 2 illustrate the convergence curves of these two functions using different optimization algorithms, respectively. As shown in Figs. 1 and 2, it is known that GA, QGA and PSO algorithms have similar convergence speed and accuracy, both of which are much lower than the DQSFL algorithm. Therefore, the proposed DQSFL algorithm should have great potential for application. We also know the test performance of the benchmark function for GA, QGA and PSO is not always stable, and is easy to fall into a local convergence. Compared to the traditional algorithms, DQSFL has a great advantage in terms of convergence speed and accuracy, the reason for this being that the combination of quantum computing and the SFL algorithm can preserve better population diversity to search for the optimal solution.

Tables 3 and 4 show the comparison of the average optimal values and execution time of the mentioned algorithms when performing the function optimization test (100 statistical results). From the results, we can see that they have little difference in the total execution time, but the optimization capability of DQSFL is obviously better than the others. By improving the quantum coding or finding the more suitable updating parameters, the DQSFL algorithm has a great potential for improving the time efficiency.

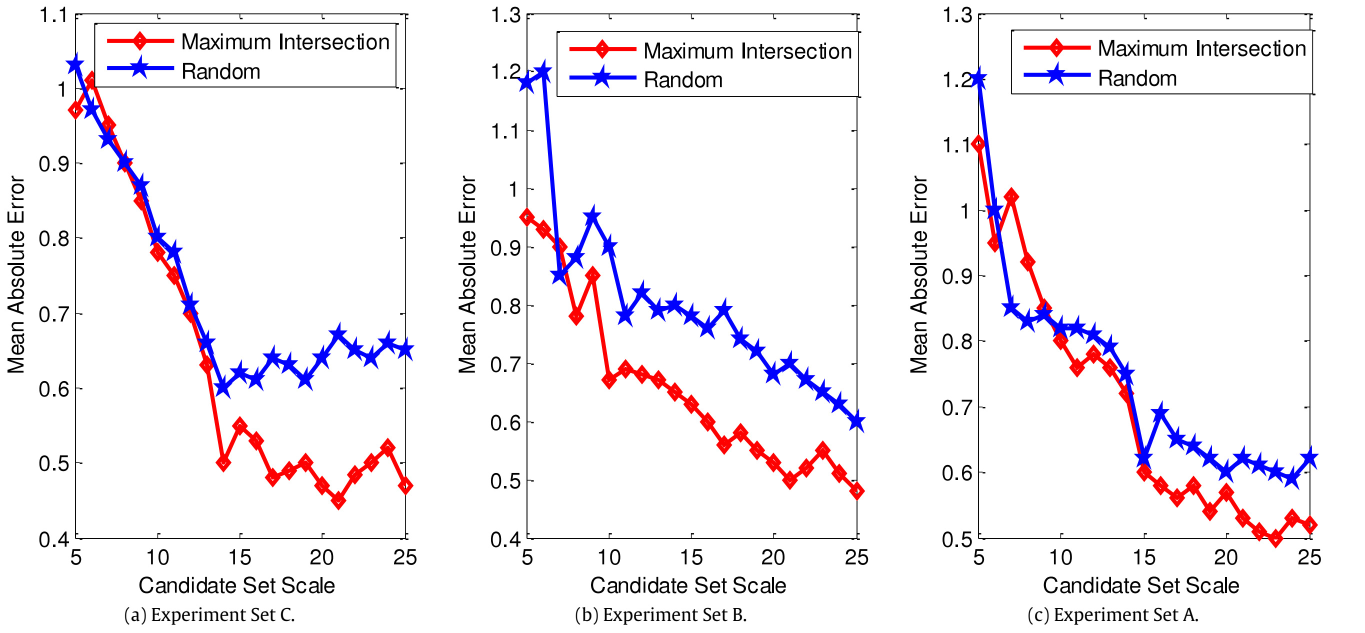


Fig. 3. The comparison results of the maximum intersection method and random method.

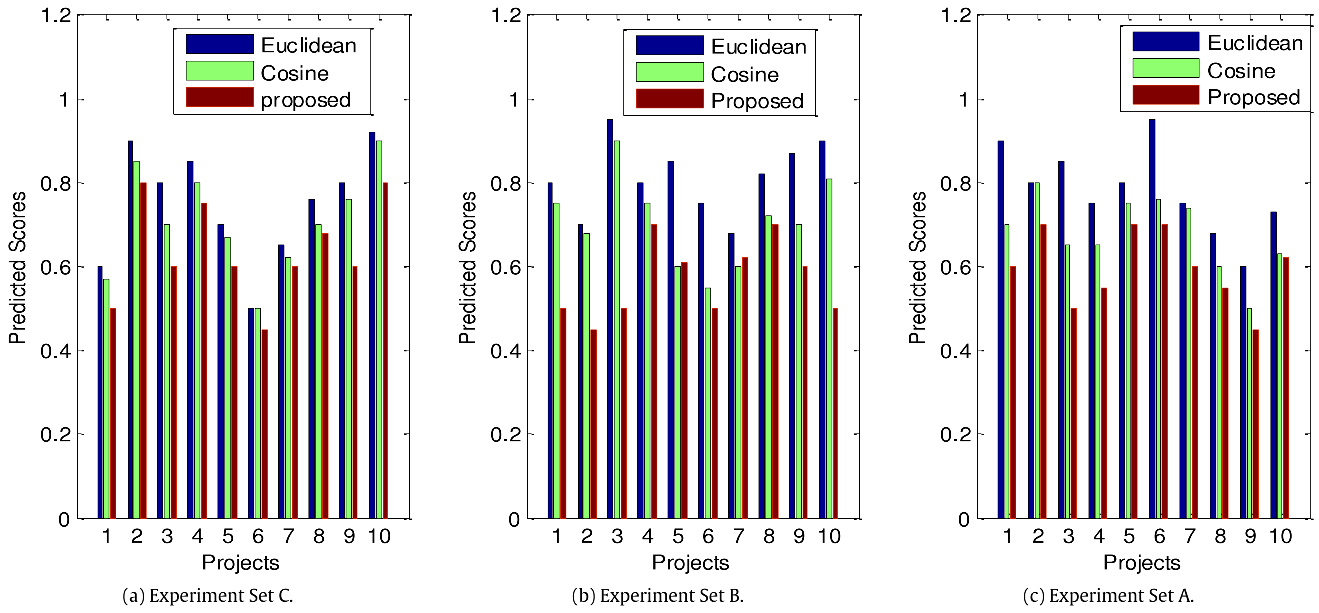


Fig. 4. Comparison results of weighted adjusted cosine similarity (proposed), Weighted Euclidean distance and cosine similarity.

5.2. Experiment using the CF algorithm based on DQSFL

(1) Experiment Design. The experiment employs the MovieLens [26] movie evaluation dataset for recommendation, which contains 943 users, 1682 projects (here this denotes movies) and ten thousand score records. In this experiment, we use the characteristics of this dataset namely: three user features of age, occupation and gender; and 18 project types such as action, adventure, animation and so on.

To ensure the accuracy of the rating score predictions, the whole dataset is divided into Part A, Part B and Part C to conduct three cross-experiments. For each experiment, we select any two parts as the experimental project training set, which is used to train weighted adjusted cosine similarity and modified Euclidean distance weights. The remaining part is used as the experimental set to verify the effectiveness of the optimization results. The three cross-experiment datasets are set as follows: (1) both A and B are training sets, and C is the experimental set; (2) both A and C are training sets, and B is the experimental set; (3) both B and C are training sets, and A is the experimental set. Moreover, each experiment can be divided into two subgroups as follows:

- Randomly select 10 projects as the current projects, and employ the maximum intersection method to extract the nearest neighbor set, where the number of projects is n ($n = 10, 11, \ldots, 30$).
- Randomly select 10 projects as the current projects, and employ the random method to extract the nearest neighbor set, where the number of projects is n ($n = 10, 11, \ldots, 30$).

For the training set, the parameters of the DQSFL algorithm are initialized as follows: the quantum frog number is $N = 40$, the other parameters are the same as those used in the previous experiment and the maximum iteration times are $M = 1000$.

The differences in extracting the nearest neighbor set in experiments 1 and 2 verify the effectiveness of the maximum intersection method for improving the rating score prediction accuracy.

Furthermore, we design two contrasting experiments to verify the effectiveness of weighted adjusted cosine similarity and use DQSFL to optimize the weights for the rating score prediction. Because researchers has verified that rating score prediction with weighted Euclidean distance outperforms one with relevant similarity, only modified Euclidean distance and traditional cosine similarity are used for contrasting experiments. The details are described as follows.

- Weighted modified Euclidean distance

The experiment process of rating score prediction with weighted modified Euclidean distance is similar to the rating score prediction experiment with weighted adjusted cosine similarity. When computing similarity, the Euclidean distance can be calculated as

$$euclidean(A, j) = \sqrt{\sum_{i=1}^{z}\sum_{f=1}^{22} w_f \times diff_{i,j}(A, j)^2} \tag{17}$$

where A denotes the current project, j denotes the project selected from the project configuration process, w_f denotes the weight of characteristic f of the current project, z represents the number of users who score the current project, and $diff_{i,j}(A, j)$ represents the difference between the characteristics of the current project A and the selected project j.

In this experiment, we keep the current project and nearest neighbor candidate set identical for the standard experiment, and do not change the parameter setting of the DQSFL algorithm. In the prediction process, the weights obtained from the training set optimization are used as parameters to establish the rating score prediction equation according to (11) by using weighted modified Euclidean distance instead of weighted modified cosine when computing similarity. Finally, the Mean Absolute Error of (14) is used to evaluate the results. Therefore, the lower the predicted score, the higher the accuracy of the recommendation method.

- Traditional cosine similarity

To verify the effectiveness of using DQSFL to optimize weighted adjusted cosine similarity, we make traditional cosine similarity as the contrasting experiment for rating score prediction. This experiment is also divided into two groups: in the training set, the current project and the nearest neighbor candidate set of the standard experiment is used to test the similarity value according to cosine similarity; in the experiment set, the related calculation equation is established to predict scores.

(2) Result Analysis. Fig. 3 shows the performance curves of the three cross-experiments as described in 5.1.1. In Fig. 3, the x axis shows that the scale of the nearest neighbor candidate set is from 5 to 25, and the y axis denotes the Mean Absolute Error of the rating score prediction of the 10 current projects. As shown in Fig. 3, the accuracy results of the maximum intersection method are overall higher than the random method results.

After the three cross-experiments, Fig. 4 illustrates the performance comparison of the three similarity calculation methods, weighted adjusted cosine similarity optimized by the DQSFL algorithm, weighted Euclidean distance and cosine similarity. In Fig. 4, the x axis represents the 10 current projects, and the y axis denotes the rating score predictions of these 10 current projects. We use the maximum intersection method to select the nearest neighbor candidate set, the scale of which is 16. As shown in Fig. 4, the rating score prediction accuracy performance when using the weighted cosine similarity optimized by DQSFL significantly outperforms the other two methods.

Hence, according to previous performance analysis, using the maximum intersection method and weighted adjusted cosine similarity based on the DQSFL optimization algorithm obviously improves the rating score prediction accuracy of the CF algorithm, thereby enhancing the quality of the recommendation system.

6. Conclusions and future work

In this paper, we proposed a novel collaborative filtering algorithm to improve the rating score prediction accuracy of the recommendation system in social networks. In the proposed collaborative filtering algorithm, we use the maximum intersection method to extract the nearest neighbor candidate set and compute the weighted adjusted cosine similarity to rate the score prediction, which can partially avoid the rating data sparseness problem. To optimize the weights of the CF algorithm, we proposed a novel optimization method, named DQSFL, which introduces quantum computing into the SFL optimization algorithm. Moreover, we tested the DQSFL performance against other optimization methods and experiment on the proposed CF algorithm with other traditional recommendation algorithms. A comparison of the performance results shows that the DQSFL optimization algorithm has a better capability of searching for optimal solutions and can optimize discrete optimization problems, and the rating score prediction accuracy of the CF based on DQSFL outperforms other traditional recommendation algorithms, which improves the quality of social network recommendation systems.

However, there remains a lot of work to do in the future. The special feature of quantum coding can improve the capability of storage and computation of the traditional algorithms. Quantum-inspired frog algorithm has powerful global optimization ability. However, the frog leaping algorithm itself has the problem of long optimization time. It is necessary to further study the design of the quantum coding methods and the related parameters (including the leap step length, frog population, meta-evolution and global communication iterations). In addition, MovieLens is just a small-scaled database. To deploy the DQSFL-based collaborative filtering algorithm on the actual social networks or recommendation platforms to detect its performance is also what we need to do.

References

[1] S.L. Lim, A. Finkelstein, Stakerare: Using social networks and collaborative filtering for large-scale requirement elicitation, IEEE Trans. Softw. Eng. 38 (3) (2012) 707–735.

[2] X. Yang, Y. Guo, Y. Li, A survey of collaborative filtering based social recommender systems, Comput. Commun. 41 (15) (2014) 1–10.

[3] D. Goldberg, D. Nichols, B.M. Oki, Using collaborative filtering to weave an information tapestry, Commun. ACM 35 (12) (1992) 61–70.

[4] M.A.C. Juan, Y. Yang, M.T. Carlos, A memory-based collaborative filtering algorithm for recommending semantic web services, IEEE Latin Amer. Trans. 11 (2) (2013) 795–801.

[5] J. Wu, L. Chen, Y. Feng, Z. Zheng, Z. Wu, Predicting quality of service for selection by neighborhood-based collaborative filtering, IEEE Trans. Syst. Man Cybern. 43 (2) (2013) 428–439.

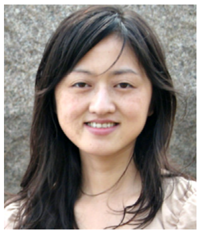
[6] S. Tyagi, K.K. Bharadwaj, Enhancing collaborative filtering recommendations by utilizing multi-objective particle swarm optimization embedded association rule mining, Swarm Evolut. Comput. 13 (2) (2013) 1423–1432.

[7] C. Huang, Y. Su, K. Tseng, Using genetic algorithms for personalized recommendation, Lecture Notes in Comput. Sci. 6422 (1) (2010) 104–112.

[8] F. Ortega, J. Sanchez, J. Bobadilla, A. Gutierrez, Improving collaborative filtering-based recommender systems results Pareto dominance, Inform. Sci. 239 (2013) 50–61 Complete.

[9] W. Agarwal, K.K. Bharadwaj, A collaborative filtering framework for friends recommendation in social networks based on interaction intensity and adaptive user similarity, Soc. Netw. Anal. Mining 3 (3) (2013) 359–379.

[10] L. Xiang, Recommendation System Applications, People Post Press, Beijing, 2012 (in Chinese).
[11] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y.S. Kim, Collaborative filtering for people to people recommendation in social networks, Lecture Notes in Comput. Sci. 6464 (1) (2011) 476–485.
[12] X. Su, T.M. Khoshgoftaa, A survey of collaborative filtering techniques, Adv. Artif. Intell. 23 (2) (2009) 123–129.
[13] F. Zhang, H. Chang, Using BP neural network to ease rating data sparseness problem of collaborative filtering recommendation algorithms, Comput. Res. Develop. 21 (4) (2006) 667–672 (in Chinese).
[14] J. Rennie, N. Srebro, Fast maximum margin matrix factorization for collaborative prediction, in: Proceedings of the 22nd International Conference on Machine learning, ACM Press, New York, 2012, pp. 713–719.
[15] D. Decoste, Collaborative prediction using ensembles of maximum margin matrix factorizations, in: Proceedings of the 23rd International Conference no Machine Learning, ACM Press, New York, 2013, pp. 249–256.
[16] P. Sprechmann, I. Ramirez, G. Sapiro, Y.C. Eldar, C-Hilasso: A collaborative hierarchical sparse modeling framework, IEEE Trans. Signal Process. 59 (9) (2011) 4183–4198.
[17] M. Eusuff, K. Lansey, Optimization of water distribution network design using the shuffled frog leaping algorithm, J. Water Resour. Plann. Manage. 129 (3) (2003) 210–225.
[18] J. Luo, M. Chen, Design and analysis of trajectory and convergence of shuffled frog leaping algorithm, Signal Process. 26 (9) (2010) 1428–1433.
[19] E. Elbeltagi, T. Hegazy, D. Grierson, Comparison among five evolutionary-based optimization algorithms, Adv. Eng. Inform. 19 (1) (2005) 43–53.
[20] S.L. Ho, S. Yang, J. Huang, A quantum inspired evolutionary algorithm for multi-objective design, IEEE Trans. Magn. 49 (5) (2013) 1609–1612.
[21] X. Chen, X. Xia, R. Yu, Quantum ant colony algorithm based on bloch coordinates, Lecture Notes in Comput. Sci. 7473 (1) (2012) 413–420.
[22] W. Shu, Quantum-inspired genetic algorithm based on simulated annealing for combinatorial optimization problem, Int. J. Distrib. Sens. Netw. 5 (1) (2012) 61–65.
[23] H. Li, S. Li, A quantum immune evolutionary algorithm and its application, J. Comput. Inf. Syst. 7 (8) (2011) 2972–2979.
[24] Q. Zhang, S. Xu, L. Liu, Application of mixed quantum leapfrog algorithm in the optimization of process neural network, J. Signal Process. 29 (8) (2013) 1003–1011.
[25] W.P. Ding, J.D. Wang, Z.J. Guan, Efficient rough attribute reduction based on quantum frog-leaping co-evolution, J. Acta Electron. Sin. 11 (11) (2011) 2597–2603.
[26] F. Maxwell Harper, A.K. Joseph, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. (TiiS) 5 (4) (2015) 19 19 pages, http://dx.doi.org/10.1145/2827872.


Wenjuan Li received a Ph.D. degree in 2012 from Zhejiang University, Hangzhou China, in computer science. She is now a post-doctor in Shanghai Tiao Tong University and also a visiting scholar in CLOUDS Lab at University of Melbourne. Her research interests include cloud computing, social network and trust.

Jian Cao is currently a Professor with Department of Computer Science and Engineering at Shanghai Jiao Tong University, China, and the deputy head of the department. He is also the leader of the Lab for Collaborative Intelligent Technology. His research interests include network computing, service computing and data analytics. He leads the research group of collaborative information system. He has authored or co-authored over 180 journal and conference papers in the above areas.

Jiyi Wu received a Ph.D. degree in 2011 from Zhejiang University, Hangzhou China, in computer science. He is now a director of the Key Lab of E-Business, Hangzhou Normal University. His research interests include service computing, trust and reputation.

Huang Changqin received a Ph.D. degree in 2005 from Zhejiang University, Hangzhou China, He is currently a professor at South China Normal University. His research interests include service computing, cloud computing and semantic web education.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft Pty Ltd. His research interests include cloud computing, fog computing, parallel and distributed systems.