FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 41 (2016) No. 1
DE GRUYTER OPEN
DOI: 10.1515/fcds-2016-0004
ISSN 0867-6356
e-ISSN 2300-3405

A CODE REVIEWER ASSIGNMENT MODEL INCORPORATING THE COMPETENCE DIFFERENCES AND PARTICIPANT PREFERENCES

Yanqing WANG*, Xiaolei WANG[1], Yu JIANG, Yaowen LIANG, Ying LIU[2]

Abstract. A good assignment of code reviewers can effectively utilize the intellectual resources, assure code quality and improve programmers' skills in software development. However, little research on reviewer assignment of code review has been found. In this study, a code reviewer assignment model is created based on participants' preference to reviewing assignment. With a constraint of the smallest size of a review group, the model is optimized to maximize review outcomes and avoid the negative impact of "mutual admiration society". This study shows that the reviewer assignment strategies incorporating either the reviewers' preferences or the authors' preferences get much improvement than a random assignment. The strategy incorporating authors' preference makes higher improvement than that incorporating reviewers' preference. However, when the reviewers' and authors' preference matrixes are merged, the improvement becomes moderate. The study indicates that the majority of the participants have a strong wish to work with reviewers and authors having highest competence. If we want to satisfy the preference of both reviewers and authors at the same time, the overall improvement of learning outcomes may be not the best.

Keywords: peer code review, reviewer assignment problem (RAP), code review assignment model, preference matrix, subtour-size constraint, mutual admiration society

1. Introduction

Reviewer assignment problem (RAP) is a common task to people such as software developers, conference organizers, journal editors, grant administrators and educators [2,

* Correspondence: Yanqing Wang, School of Management, Harbin Institute of Technology No.13, Fayuan St., Nangang Dist., Harbin 150001, P. R. China.
E-mail: yanqing@hit.edu.cn. Phone: +86-15124500998. FAX: +86-451-86414022
[1] School of Management, Harbin Institute of Technology, Harbin 150001, P. R. China
[2] Department of Information Systems, College of Business Administration, California State University Long Beach, CA 90840, USA

24] because qualified reviewers, especially highly competent ones, are usually scarce intellectual resources and play important roles in their fields. Sun, Ma, Fan, and Wang proposed that a good match between a reviewer's knowledge and a project would make useful and professional judgments on the research project to be funded [14]. Chen and Fan built up a model to measure the match degree between a reviewer's research domain knowledge with a proposal [1]. The match fitness between reviewers with assigned proposal (or paper manuscript) determines the review quality greatly [24] and the fact of having a good match helps improve the overall productivity [6]. Therefore, the review assignment problem is an important research topic [17].

Many contributions on RAP can be found in literature. Tayal, Saxena, Sharma, Khanna, and Gupta put forward a new method for solving reviewer assignment problem in government funding agencies [15]. Li and Watanabe proposed an automatic paper-to-reviewer assignment approach based on the matching degree of the reviewers [9]. Long, Wong, Peng, and Ye solved a conference paper assignment problem by maximizing the topic coverage of the paper-reviewer assignment [10]. Wang, Shi, and Chen did a comprehensive survey of the reviewer assignment problem [19]. However, the current researches on RAP mainly concentrate on the selection of conference paper, journal paper, or R&D projects. Not much work has been done in the area of code reviewer assignment.

In the software industry, the concept of code review is developed from code inspection proposed by Fagan of IBM [5]. Many researchers have tested the reliability and effectiveness of code review in the field of software industry [3, 12] and programming education [7, 8, 18]. In addition, some contributions to code review were made in areas such as quality assurance [23], participants' behavior analysis [22], learning outcomes analysis [21], and assessment approach [20]. The RAP in peer code review has drawn attention of some researchers. Topping showed that peers may be matched in a variety of ways for a variety of reasons. They may be matched in groups or pairs by capability, friendships, or randomly [16]. Different configurations support different goals. Random assignment of reviewers for each assignment may expose students to a wider range of ideas. Matching students in pairs over a semester allows for a longer term social interaction. Li conducted two yearlong experiments in which two reviewers were assigned randomly for each student. The study found that an anonymous review can significantly reduce the "mutual admiration society" effect [8]. In a mutual admiration society, everyone respects each other though everyone does not necessarily agree with each other in everything. The reason that a mutual admiration society in code review practice is harmful is that participants in such a society avoid giving negative comments on others' code. Wang et al. used an assessment method based on peer code review and measured the learning outcomes of the method. The study still used a random reviewer assignment strategy though they proposed a ranking-based reviewer assignment strategy as a future research topic [20].

Matching between reviewers' competence and review subjects such as research proposals and manuscripts significantly affects the review quality and productivity [24, 6]. Despite the similarities between a paper review and a code review, it is not easy to investigate the similar phenomena in a code review because a code review often happens within a small software development team and the level of competence of a reviewer or an author is hard to measure. Nonetheless, a programming class has some desired features to study the code review assignment strategies incorporating the participants' levels of competence and their preferences. In a typical programming class, students have different

levels of competence and their assignment scores are good proxies of levels of competence. Students are asked to complete multiple programming projects and as in a typical real world project, a code review is a standard step in software development in a classroom setting. Because of the special requirements of the educational environment, we developed a code review system named EduPCR to manage the code review process. The EduPCR system follows the code review standards and best practices. The system records assignment scores and code review activity data during the process. Figure 1 is the activity diagram of EduPCR system.

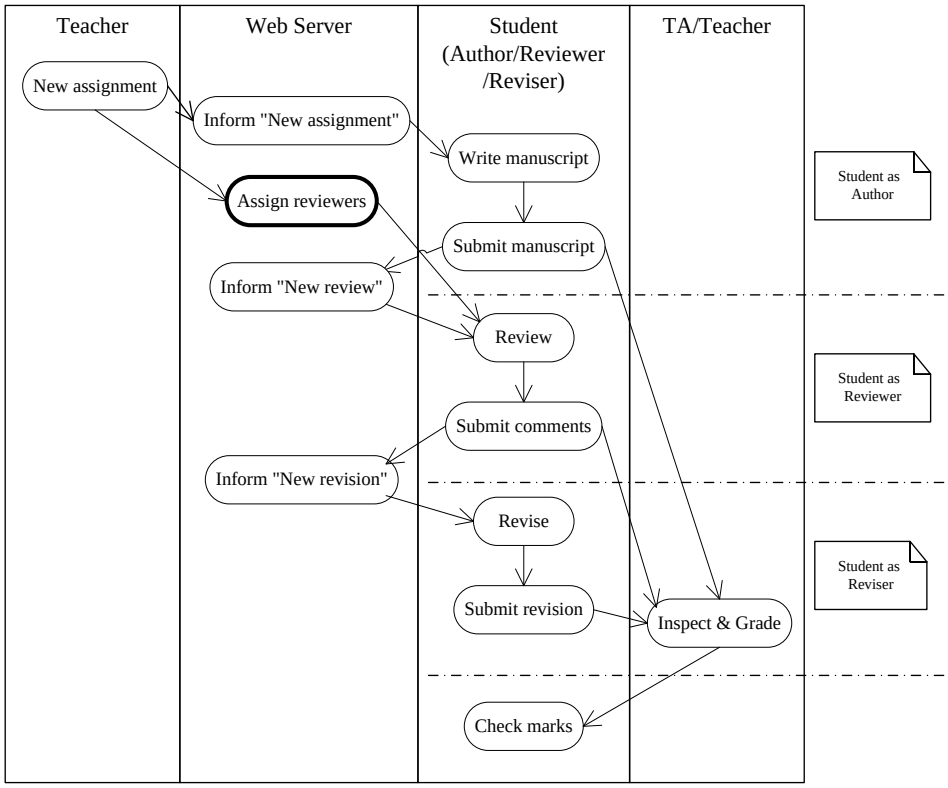

Figure 1. Activity diagram of peer code review process in EduPCR

In Figure 1, *author*, *reviewer* and *reviser* are three roles that every student plays in different stages; *manuscript* stands for the first version of a program written by an author according to the task arranged by the teacher; *comments* indicate the suggestions to an author proposed by a reviewer; *revision* presents the revised version of an author's program. In a special case, a student without submitting the manuscript code is not eligible to be a reviewer so as to facilitate the course management. The code review process is driven by a web server that informs students of notifications such as "You have a new assignment", "You have an incoming review job" and so on by sending them short messages. The EduPCR system automates many tasks such as reviewer assignment, event notification and data collection. From the perspective of software development, the coding tasks and code

review activities are the same as typical real world projects. The grading is conducted manually and scores are typed into the system as measures of students' levels of competence.

EduPCR uses a random reviewer assignment strategy that treats all participants equally. However, as in a typical development team, students' programming competences are not at the same level. The assignments between two students with different competence levels may produce different results. Additionally, programmers care about who will review their code and whose code to be reviewed. We like to know whether high-level participants are willing to help low-level ones and the influences of such reviewer assignments.

2. Methodology

Integer linear programming is a very successful and popular method being used to solve the RAP [2, 14, 6]. Based on these work, we recommend a new approach to study the code reviewer assignment problem with integer linear programming. The key issue is how to define the preference of each review pair in a format of {reviewer→author}. Without loss of generality, assuming that all students can be categorized into three levels of *high*, *middle* and *low* according to their programming competency, we believe that the review quality will be affected by a reviewer assignment strategy, i.e., how the assignments are made between the students of different competence levels.

2.1. Ranking students' competence

An assumption or precondition of solving the RAP in this study is to measure all students' programming competences. To measure the level of a student's competence more objectively and effectively, we take the cumulative average scores of the students as the measurement value. In our system, the score of a student in a task includes both a quality score (assessing students' programming ability) and a review score (assessing students' review ability). The latter is acquired by an extra step that an author needs to grade his/her reviewer's review. After all students complete a programming task, they get scores according to the quality of their work and their average scores are updated. Because grading is conducted by the same teacher manually, the scores are good indicators of their programming competences. It is reliable because we only care about the relative competence levels (*high*, *middle*, *low*) in this research. The value of a student's competence is a cumulative one. As a result, the more tasks a student has finished, the more accurate this value is. The detailed algorithm is described as follows:

At the beginning, we initialize every student's competence with a *middle* level. When each program task is done, a student's average score is recalculated. Then we rank all students according to their programming competence using k-means clustering approach. For the convenience of this study, a student's programming competence is ranked as one of three levels of *high*, *middle*, and *low*.

2.2. Defining preference matrix

After ranking students' competences, we need to create a data structure representing the student's reviewer assignment preference between each pair of levels. For the three levels of competences, we define a square matrix $P_{3\times3}$ and name it as *preference matrix*, as depicted in Equation (1).

$$\mathrm{P}=\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \tag{1}$$

In Equation (1), p_{ij} is the entry in row i, column j of P, which denotes a preference value by a reviewer level to an author level. The indexes i and j are in the range of *1* to *3*, which represent the three levels *high*, *middle*, and *low* of reviewer and author respectively. A preference matrix depicts all combinations of different competence levels of students. For different assignment preferences, there are different element values for a preference matrix. The values in a preference matrix and a reviewer assignment strategy determine the result of an integer programming optimization problem.

2.3. Forming match matrix

In a RAP model, a match matrix is an instantiation or an expansion of a preference matrix. In other words, we can get the match value between two specific students as long as we know their competence levels. With knowledge of each student's competence, it is easy to construct a match matrix by checking the preference matrix just like consulting a dictionary.

Let $N = \{1, ..., n\}$ where n is the number of authors to be reviewed. $M_{m\times n}$ is used as a notation for a matrix with m rows (reviewers) and n columns (authors). Because in our research each student plays the roles of both reviewer and author exactly once in a coding assignment, the match matrix is a square matrix $M_{n\times n}$. The element m_{ij} in M stands for the degree of match between a reviewer i and an author j ($i, j \in N$).

It is easy to obtain the match matrix M by expanding the preference matrix P with knowledge of each student's competence level. For example, if the ranking of reviewer i is a *middle* level and that of author j is a *high* level, then the data element m_{ij} in M is equal to the element value of p_{21} (see E.1 for details). To avoid an author as a reviewer of his/her code, the elements on the diagonal line m_{ii} are all set to a very large negative value.

2.4. A RAP model with a subtour-size constraint

Because every student acts as an author and a reviewer exactly once in a task, the RAP in this study becomes a balanced assignment problem [11], in which the number of reviewers is equal to that of authors. Theoretically, the result of RAP forms one or more closed cycles. Similar to the term of closed cycle in a traditional asymmetric travelling salesman problem and in the literature by Drexl and Irnich [4], we name a closed cycle in our RAP study as a *subtour*. Actually, a subtour is a connected sub-graph with in-degree and out-degree of *1* at each node.

However, if we apply the model of traditional balanced assignment problem, we cannot control the size (number of edges) of a closed cycle. In some RAPs such as a code reviewer assignment problem, the control of the size of the shortest closed cycle is often required [7, 8, 20]. For example, Li found that the "mutual admiration society" phenomenon was one of the typical pitfalls in code review process [7]. In our experience, though the reviewer assignment was performed by a web server automatically and anonymously, a student in a small-size review group may guess the names of other students more easily than in a big-size review group. To reduce the negative impact of a "mutual admiration society", we improve the traditional balanced RAP model by introducing a size constraint of the smallest review group, i.e. a subtour-size constraint. As for the optimization objective, a reviewer assignment strategy should respect participants' preferences and tries to maximize the total preference value. Thus, the reviewer assignment problem is formally defined as the following:

$$\max \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij} \cdot a_{ij}$$

$$s.t. \begin{cases} \sum_{j=1}^{n} a_{ij} = 1, \text{for all } i \in N \\ \sum_{i=1}^{n} a_{ij} = 1, \text{for all } j \in N \\ \sum_{i \in S}\sum_{j \in S} a_{ij} \le |S| - 1, \text{ for all } S \subset N \text{ with } 1 \le |S| \le c-1 \\ a_{ij} \in \{0,1\} \end{cases} \tag{2}$$

In the model defined in (2), the definitions of n, N, m_{ij} and M have the same meanings as in Section 2.3. The decision variable $A_{n\times n}$ is called an assignment matrix, in which a_{ij} will be *1* if a reviewer i is assigned to an author j or it will be *0* if not. The subtour-size constraint in the third line of the model is used to control the cycle size of the smallest review group c. S stands for any nonempty proper subset of N and $|S|$ is the size of set S.

Theoretically, as the minimum value of subtour size, the c could be an integer in the range of *2* to n according to the real application requirement. The smaller the given c is, the easier the negative impact of the "mutual admiration society" could be enhanced. In an extreme case when the c is *2*, a student *A* reviews a student *B*'s work and the student *B* reviews that of *A*. In this case, the chance is high for the two to build up a "mutual admiration society".

3. Preference matrix calculation

According to the optimization model in Equation (2), we have to get the match matrix M before we utilize this optimization model. The preference matrix P is the abstract format of match matrix M, so the preference matrix P is our next focus. As mentioned in (1), given that all students are ranked into three levels, a preference matrix P is a 3x3 matrix, in which

the element p_{ij} denotes the preference degree of reviewers at level i to review the code of authors at level j.

Unlike analytic hierarchy process [13], in which weights are obtained by pairwise comparison matrixes, we acquire the preference matrix P according to students' preferences to reviewer assignments. Our survey implied that the more satisfied the students are with the reviewer assignment strategy, the better review quality and the higher learning outcomes could be achieved.

In order to discover each participant's preference to a reviewer assignment, a questionnaire was handed out to all participants. They are undergraduate students from Year 1 to Year 4 majoring in *Information System* in a public university in China. All respondents have experience of using EduPCR. In total, 104 copies were given out and 94 valid copies were collected. Each student was asked to answer the following two questions:

Q1. As a reviewer, reviewing which degree's work can help you more with your learning outcome? (Please sort them in DESCENDING order):

A. a student two levels superior to me
B. a student one level superior to me
C. a student of the same level as me
D. a student one level inferior to me
E. a student two levels inferior to me
*Order:*_____

Q2. As an author, reviewer of which degree can help you more with your learning outcome? (Please sort them in DESCENDING order):

A. a student two levels superior to me
B. a student one level superior to me
C. a student of the same level as me
D. a student one level inferior to me
E. a student two levels inferior to me
*Order:*_____

The following steps constitute the algorithm of generating preference matrixes and applying changes to preference matrixes.

(1) Initializing. Every value in the preference matrix is set to 0.

(2) Determining weights for options. Different preference options affect different number of items in a preference matrix. For example, an option "*A*" to *Q2* will affect item p_{13} alone while an option "*C*" to *Q2* will affect p_{11}, p_{22} and p_{33}. To eliminate the influence of repeated computation, the weights to option "*A*" through "*E*" are set to 1, 1/2, 1/3, 1/2 and 1 respectively.

(3) Determining weights for preference position. The position of an option determines its preference level because options are sorted in descending order of preference. We adopt a common approach in statistics, the reciprocal of e, to set the weights for options in the first through the fifth position with $e^0, e^{-1}, e^{-2}, e^{-3}, e^{-4}$ respectively.

(4) Computing initial preference matrixes. We multiply option weight by position weight and sum the values in the preference matrix for all respondents. For example, if a student chooses "*B*" as the first option, two items are affected, i.e. $p_{21}=p_{21}+(1/2)e^0$, $p_{32}=p_{32}+(1/2)e^0$. Two preference matrixes P_1 and P_2 are obtained from the computation of

two questions independently, of which the P_1 is from the preference view of reviewer (part (a) in Table 1) and the P_2 is from the preference view of author (part (b) in Table 1).

Table 1. Initial matrixes of students' preference to reviewer assignment

		author		
		high	middle	low
reviewer	high	8.75	5.26	2.10
	middle	24.66	8.75	5.26
	low	35.92	24.66	8.75

(a) P_1 by reviewer's preference

		reviewer		
		high	middle	low
author	high	6.75	2.42	2.28
	middle	22.47	6.75	2.42
	low	50.25	22.47	6.75

(b) P_2 by author's preference

From P_1 and P_2, we can find that the reviewer assignment preference by both reviewers and authors has similar pattern. For example, the values in the first line of P_1 shows that high-level reviewers prefer to review the work of high-level authors rather than that of low-level ones. Also, from the values in the first line of P_2, it is obvious that high-level authors wish their work to be reviewed by high-level reviewers rather than low-level ones.

(5) Normalizing and transposing. Normalization is achieved by dividing every item in a preference matrix by the smallest value in the matrix. Besides, P_1 and P_2 come from different points of view by reviewers and authors. So we transpose one of them to make them have the same meaning in the following optimization. The normalized matrix of P_1 and the transposed and normalized matrix of P_2 are P_3 and P_4 correspondingly in Table 2.

Table 2. Two adjusted matrixes of students' preference to reviewer assignment

		author		
		high	middle	low
reviewer	high	4.17	2.51	1.00
	middle	11.74	4.17	2.51
	low	17.11	11.74	4.17

(a) P_3 by reviewer's preference

		author		
		high	middle	low
reviewer	high	2.96	9.86	22.04
	middle	1.06	2.96	9.86
	low	1.00	1.06	2.96

(b) P_4 by author's preference

(6) Merging. P_1 and P_2 denote the different preference of authors and reviewers. It would be interesting to see the effect of a merged preference after we study them separately. In Equation (3), P_5 is the normalization of the merging of the initial preference matrix by reviewer (P_1) and the transposed initial preference matrix by author (P_2^T). M is the smallest value in the merged matrix.

$$P_5 = (P_1 + P_2^T)/M = \begin{bmatrix} 1.00 & 1.79 & 3.38 \\ 1.75 & 1.00 & 1.79 \\ 2.46 & 1.75 & 1.00 \end{bmatrix} \quad (3)$$

From the merged preference matrix P_5, it is found that the students' preference pairs of reviewer assignment are listed in descending order as follows:

- assigning high-level reviewer to low-level author
- assigning low-level reviewer to high-level author
- assigning adjacently higher-level reviewer to author
- assigning adjacently lower-level reviewer to author
- assigning same-level reviewer to author

4. Simulations and analysis

We used the EduPCR system and random reviewer assignment strategy in two introductory *C Programming* classes in 2010 and 2011. The first class has 10 programming assignments and 86 students. The second class has 12 programming assignments and 23 students.

To investigate the effects of competence levels and review preferences, we used the adjusted preference matrix to optimize the reviewer assignment problem and compare it with the random reviewer assignment strategy. Both classes used the EduPCR system to collect baseline data of the random strategy. With the real score data, we did two simulations to optimize the RAP with the preference matrixes P_3, P_4, and P_5.

4.1. Random reviewer assignment algorithm

We implemented a random code reviewer assignment algorithm with a minimum subtour size constraint in our current EduPCR system. The detailed algorithm is described as follows. In the following description, L is an element collection of all students' IDs, ranging from 0 to the total number of students minus one.

$L \leftarrow \{0..num\text{-}1\}$ // Initialize arraylist L with 0..num-1 in ascending order
$i \leftarrow 0$ // i is the index of subtours
while L is not empty **do**
| new a subtour s_i
| $k \leftarrow$random(|L|) // get a random position in the range of 0 to $|L|-1$
| $s_{i,0} \leftarrow L_k$ // determine the head of a subtour
| $j \leftarrow 1$ // j is the index of elements in one subtour
| **do**
| | $k \leftarrow$random(|L|)
| | **if** $k \neq s_{i,0}$ **then** // there does not form a circle yet
| | $s_{i,j} \leftarrow L_k$
| | $j \leftarrow j+1$
| | remove L_k from L
| | **end**
| | **else if** $|L|>2$ **then** // when the next element is the head of a subtour, close it
| | remove L_k from L
| | $i \leftarrow i+1$ // start the next subtour
| | break // go to the beginning of outer loop
| | **end**
| | **else if** $|L|=2$ **then** // do not leave two elements alone.

```
|  |  s_{i,j} ← L_0
|  |  s_{i,j+1} ← L_1
|  |  remove L_0, L_1 and L_k from L
|  | end
|  | else if |L|=1 then // do not leave one element alone.
|  |  s_{i,j} ← L_0
|  |  remove L_0 and L_k from L
|  | end
| while L is not empty
end
```

In the above random algorithm, we set the minimal size of a subtour to *3*. When there are only one or two elements in a subtour, they will be merged into another subtour to meet the minimal subtour size constraint. For example, the 23 students in the second class are assigned randomly to four subtours as follows. The arrow mark "→" stands for the review relationship of a reviewer to an author.

subtour 1 : 13→0→8→14→2→19→11→5→12→3→15→13
subtour 2 : 18→1→21→20→9→4→6→18
subtour 3 : 22→7→10→22
subtour 4 : 16→17→16

The *subtour 4* {16, 17} only has two elements therefore they are merged to the *subtour 3* {22, 7, 10} to build up a new *subtour 3'* consisting of five elements as follows.

subtour 3' : 22→7→10→16→17→22

4.2. Code reviewer assignment procedure with optimization model

Using the reviewer assignment optimization algorithm, the code reviewer assignment procedure has the following steps:

(1) Ranking students' competence. The students' cumulative average scores were taken as input, and their programming competences were classified into three levels of *high*, *middle* and *low* using k-means clustering approach. The level of one particular student is dynamic because his/her average score may vary when a new programming assignment is completed.

(2) Building up the match matrix. We built up the match matrix M by retrieving the values in preference matrix P.

(3) Optimizing. The program was written in *Java* language using a software package named ILOG CPLEX. In order to make the following comparative study more accurate, the size of the shortest subtour was determined as *3* because the same number was utilized in the random assignment algorithm just mentioned.

(4) Controlling the size of shortest subtour. During the integer programming process, the size of every subtour was checked after each iteration. When the size of each subtour was greater than or equal to three, the iteration was terminated and an optimal solution was obtained. Finally, the reviewer assignment result was acquired by parsing the decision variable matrix X_{86x86} and X_{23x23} in these two simulations.

4.3. Comparative study with random assignment

By searching the data in the database of EduPCR system, the data of an actual reviewer assignment applying random assignment strategy were extracted. The study process includes:

(1) Building up the random assignment matrix. Based on the actual random assignment data, we build up the random assignment matrix A_r (86x86 in simulation 1 and 23×23 in simulation 2), which has an identical structure and meaning with decision variable X but has different values in it.

(2) Summarizing the total preference values separately. Expanding three matrixes P_3, P_4 and P_5, three match matrixes M_1, M_2, M_3 were obtained. Using the match matrixes, we got six total preference values T_{r1}, T_{r2}, T_{r3}, T_{o1}, T_{o2} and T_{o3}. T_{r1}, T_{r2} and T_{r3} is the total preference values by random assignment while T_{o1}, T_{o2} and T_{o3} stand for the total preference values after the optimization (See (4)). In Equation (4), A_1, A_2 and A_3 are optimized assignment matrixes.

$$\begin{aligned} T_{r1} &= sumproduct(M_1, A_r), \quad T_{r2} = sumproduct(M_2, A_r), \quad T_{r3} = sumproduct(M_3, A_r) \\ T_{o1} &= sumproduct(M_1, A_1), \quad T_{o2} = sumproduct(M_2, A_2), \quad T_{o3} = sumproduct(M_3, A_3) \end{aligned} \tag{4}$$

(3) Computing the improvement. We use equation $C=(T_o-T_r)/T_r$ to evaluate the optimization performance. T_o and T_r are total preference values. The difference between them represents the degree of improvement. The improvement values of each task and average were acquired, as shown in Table 3 and Table 4.

Table 3 and Table 4 show that in both simulations, all ranking-based optimization assignment strategies improve total preference values from 17.6% to 28.1% compared with the random assignment outcomes. That is to say, as long as we conduct the optimization using preference matrix P_3, P_4 or P_5, we will achieve higher learning outcomes than we use random assignment strategy.

Table 3. Comparison of optimization assignment to random assignment (Simulation 1)

Task No.	T_{r1}	T_{o1}	C_1	T_{r2}	T_{o2}	C_2	T_{r3}	T_{o3}	C_3
1	480.53	588.87	0.225	361.40	475.92	0.317	118.30	149.88	0.267
2	491.06	598.64	0.219	406.06	493.04	0.214	126.29	153.72	0.217
3	528.76	558.84	0.057	433.00	503.64	0.163	135.34	149.82	0.107
4	515.00	533.39	0.036	457.48	464.40	0.015	137.10	140.60	0.026
5	467.55	603.83	0.291	396.98	577.12	0.454	121.75	166.72	0.369
6	529.21	613.60	0.159	448.84	594.24	0.324	137.74	170.56	0.238
7	549.38	615.65	0.121	498.22	587.12	0.178	147.73	169.80	0.149
8	529.15	652.92	0.234	453.64	636.36	0.403	138.42	182.10	0.316
9	525.51	652.92	0.242	475.46	636.36	0.338	141.15	182.10	0.290
10	507.75	666.12	0.312	444.10	636.36	0.433	134.15	182.34	0.359
avg.	512.39	608.48	***0.188***	437.52	560.46	***0.281***	133.80	164.76	***0.231***

Table 4. Comparison of optimization assignment to random assignment (Simulation 2)

Task No.	T_{r1}	T_{o1}	C_1	T_{r2}	T_{o2}	C_2	T_{r3}	T_{o3}	C_3
1	127.27	139.09	0.093	112.32	122.32	0.089	33.76	36.84	0.091
2	129.32	172.50	0.334	105.20	159.44	0.516	33.00	46.84	0.419
3	152.96	173.34	0.133	125.20	161.56	0.290	39.16	46.22	0.180
4	155.46	167.43	0.077	133.92	156.56	0.169	40.80	44.68	0.095
5	145.00	173.34	0.195	127.32	161.56	0.269	38.38	46.22	0.204
6	145.00	166.59	0.149	127.32	154.44	0.213	38.38	45.30	0.180
7	142.95	174.70	0.222	134.44	159.44	0.186	39.14	46.88	0.198
8	146.59	167.28	0.141	124.08	149.44	0.204	38.12	43.88	0.151
9	131.37	152.96	0.164	98.08	125.20	0.277	32.24	39.16	0.215
10	140.99	152.96	0.085	109.92	125.20	0.139	35.28	39.16	0.110
11	138.94	170.84	0.230	117.04	147.32	0.259	36.04	44.58	0.237
12	139.17	182.42	0.311	131.26	183.68	0.399	38.17	51.48	0.349
avg.	141.25	166.12	***0.176***	120.51	150.51	***0.249***	36.87	44.27	***0.201***

4.4. Result analysis

Even though all three optimized matrixes achieve better performance than random strategy, the simulations indicate two similar phenomena: (1) the optimization by author's preference gives more improvement than that by reviewer's preference; (2) the optimization by the merged preference gives the moderate improvement. Why do these phenomena happen?

Phenomenon 1: After reanalyzing the values in P_1 and P_2 and interviewing some students, the reason is discovered. On the one hand, the number of students who regard themselves to be a "low" level is bigger than that of students who regards themselves to be a "high" level. On the other hand, the low-level students are much more eager to cooperate with high-level students than the high-level students are. The maximum value in P_2 (50.25) is much bigger than the maximum value in P_1 (35.92). This major difference demonstrates that as low-level students, their desire to have a high-level reviewer is much stronger than their desire to have a high-level author. It is a rational behavior because one can learn more when a high-level reviewer gives useful comments to one's code. The optimization is to maximize the total preference value so that the strongest preference was met at the highest priority. Therefore, author's preference gives more improvement than reviewer' preference after optimization.

Phenomenon 2: In the merged preference matrix P_5, the top two preference pairs are {high→low} and {low→high}. It seems like a contradiction. Actually, with strong wish of "harmonious development", the low-level students are trying to get more help by persuading the teacher or the web server to assign them to the high-level authors and assign the high-level reviewers to them as well. Therefore, considering both author's preference and reviewer's preference, the merged preference matrix P_5 gives the moderate improvement.

5. Conclusion and future work

The RAP becomes a pressing concern because of the increasing trend of peer review practices by many people such as conference organizers, journal editors and grant administrators [6]. Similarly, in the field of peer code review, the research on reviewer assignment is needed urgently since it plays the roles of both assuring program's quality and enhancing learning outcomes [20]. A programming class provides a good context to study different assignment strategy because we know the skill levels of all students.

We used the peer code review system EduPCR to study the student's preferences in a code reviewer assignment. The contribution includes creating a preference matrix and constructing an improved RAP model with a subtour-size constraint. The subtour-size constraint is applied to minimize the negative impact of "mutual admiration society". The subsequent simulations show the practical value of the optimization approach and the performance of the assignment with students' preference matrix is better than that of the random assignment.

However, the RAP is a quite challenging issue [24]. There are more interesting future research topics. For example, (1) "mutual admiration society" is a neutral phenomenon itself, but the correlation of review group size (closed circle) with the negative impact of "mutual admiration society" should be an interesting topic; (2) since a preference matrix can manipulate optimization results, can we construct it to achieve the goal of specialized settings? That is to say, can we design a preference matrix for the benefit of low-level programmers alone or high-level programmers alone? (3) if the preference matrix can be designed, the orientation effect of different preference matrixes on the competence improvement of programmers will be a good research topic.

Acknowledgement

This work was partially supported by National Natural Science Foundation of China [grant number 71573065] and Online Education Research Foundation (QTone Education) of China's MOE Research Center for Online Education [grant number 2016YB130]

References

[1] Chen Y., Fan Z. P., A Method for proposal-reviewer assignment in proposal review based on the match degree of research discipline, *Chinese Journal of Management Science*, 19, 2, 2011, 169-173 (in Chinese).

[2] Cook W. D., Golany B., Kress M., Penn M., Raviv T., Optimal allocation of proposals to reviewers to facilitate effective ranking, *Management Science*, 51, 4, 2005, 655-661.

[3] Devito Da Cunha A., Greathead D., Does personality matter? An analysis of code-review ability, *Communications of the ACM*, 50, 5, 2007, 109-112.

[4] Drexl M., Irnich S., Solving elementary shortest-path problems as mixed-integer programs, *OR spectrum*, 36, 2, 2014, 281-296.

[5] Fagan E., Design and code inspections to reduce errors in program development, *IBM System Journal*, **3**, 1976, 182-211.
[6] Karimzadehgan M., Zhai C. X., Integer linear programming for constrained multi-aspect committee review assignment, *Information Processing and Management*, 48, 4, 2012, 725-740.
[7] Li X., Using peer review to assess coding standards–a case study, In *Frontiers in education conference, 36th annual*, San Diego, CA, USA, 2006, 9-14.
[8] Li X., Incorporating a code review process into the assessment, In *the 20th Annual Conference of the National Advisory Committee on Computing Qualifications*, Nelson, New Zealand, 2007, 125-131.
[9] Li X., Watanabe T., Automatic paper-to-reviewer assignment based on the matching degree of the reviewers, *Procedia Computer Science*, 22, 2013, 633-642.
[10] Long C., Wong R. C. W., Peng Y., Ye L., On good and fair paper-reviewer assignment, In *IEEE 13th International Conference on Data Mining* (ICDM'2013), 2013, 1145-1150.
[11] Martello S., Pulleyblank W. R., Toth P., de Werra D., Balanced optimization problems, *Operations Research Letters*, 3, 5, 1984, 275-278.
[12] Meyer B., Design and code reviews in the age of the internet, *Communications of the ACM*, 51, 9, 2008, 66-71.
[13] Saaty T. L., How to make a decision: the analytic hierarchy process, *European journal of operational research*, 48, 1, 1990, 9-26.
[14] Sun Y. H., Ma J., Fan Z. P., Wang J., A hybrid knowledge and model approach for reviewer assignment, *Expert Systems with Applications*, 34, 2, 2008, 817-824.
[15] Tayal D. K., Saxena P. C., Sharma A., Khanna G., Gupta S., New method for solving reviewer assignment problem using type-2 fuzzy sets and fuzzy functions, *Applied intelligence*, 40, 1, 2014, 54-73.
[16] Topping K., Peer Assessment Between Students in Colleges and Universities, *Review of Educational Research*, 68, 3, 1998, 249-276.
[17] Tsang E. W. K., Is this referee really my peer? A challenge to the peer-review process, *Journal of Management Inquiry*, 22, 2, 2013, 166-171.
[18] Turner S. A., Peer review in CS2: the effects on attitudes, engagement, and conceptual learning, Doctoral Dissertation of Virginia Polytechnic Institute and State University, 2009, Retrieved from http://scholar.lib.vt.edu/theses/available/etd-08272009-003738/unrestricted/Turner_SA_D_2009.pdf
[19] Wang F., Shi N., Chen B., A comprehensive survey of the reviewer assignment problem, *International Journal of Information Technology and Decision Making*, 9, 4, 2010, 645-668.
[20] Wang Y. Q., Li H., Feng Y. Q., Jiang Y., Liu Y., Assessment of programming language learning based on peer code review model: Implementation and experience report, *Computers & Education*, 59, 2, 2012, 412-422.
[21] Wang Y. Q., Li H., Sun Y. N., Jiang Y., Yu J., Learning outcomes of programming language courses based on peer code review model, In *the 6th International Conference on Computer Science & Education*, August 3-5, SuperStar Virgo, Singapore, ThC 5.47, 2011, 751-754.
[22] Wang Y. Q., Li Y. J., Collins M., Liu P. J., Process improvement of peer code review and behavior analysis of its participants, *ACM SIGCSE Bulletin*, 40, 1, 2008, 107-111.

[23] Wang Y. Q., Yang F., Liu P. J., Collins M., Quality assurance of peer code review process: A computer science based strategy, *Zhongshan Daxue Xuebao/Acta Scientiarum Natralium Universitatis Sunyatseni*, 46(suppl), 2007, 116-120.
[24] Xu Y. H., Ma J., Sun Y. H., Hao G., Xu W., Zhao D. T., A decision support approach for assigning reviewers to proposals, *Expert Systems with Applications*, 37, 10, 2010, 6948-6956.

Received 23.10.2014, accepted 20.10.2015