Personalized Teammate Recommendation for Crowdsourced Software Developers

Luting Ye
yeluting@act.buaa.edu.cn
SKLSDE Lab, School of Computer Science and Engineering, Beihang University
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
Beijing, China

Hailong Sun
sunhl@buaa.edu.cn
SKLSDE Lab, School of Computer Science and Engineering, Beihang University
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
Beijing, China

Xu Wang
wangxu@act.buaa.edu.cn
SKLSDE Lab, School of Computer Science and Engineering, Beihang University
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
Beijing, China

Jiaruijue Wang
whynot_s@outlook.com
SKLSDE Lab, School of Computer Science and Engineering, Beihang University
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
Beijing, China

ABSTRACT
Most crowdsourced software development platforms adopt contest paradigm to solicit contributions from the community. To attain competitiveness in complex tasks, crowdsourced software developers often choose to work with others collaboratively. However, existing crowdsourcing platforms generally assume independent contributions from developers and do not provide effective support for team formation. Prior studies on team recommendation aim at optimizing task outcomes by recommending the most suitable team for a task instead of finding appropriate collaborators for a specific person. In this work, we are concerned with teammate recommendation for crowdsourcing developers. First, we present the results of an empirical study of Kaggle, which shows that developers’ personal teammate preferences are mainly affected by three factors. Second, we give a collaboration willingness model to characterize developers’ teammate preferences and formulate the teammate recommendation problem as an optimization problem. Then we design an approximation algorithm to find suitable teammates for a developer. Finally, we have conducted a set of experiments on a Kaggle dataset to evaluate the effectiveness of our approach.

KEYWORDS
Crowdsourcing, collaboration willingness, teammate recommendation

CCS CONCEPTS
• Software and its engineering → Programming teams; Collaboration in software development; Software creation and management;

1 INTRODUCTION
As a prevalent distributed problem-solving model, crowdsourcing has been widely applied to software development [1]. In recent years, a number of crowdsourced software development(CSD) platforms (e.g. Kaggle [2] and Topcoder [3]) have been built to solicit solutions to software development tasks. In CSD platforms, task requestors publish tasks with detailed descriptions and a certain amount of monetary rewards. Then developers can choose to participate individually or collaboratively and only a few top ranked developers can obtain rewards according to the pre-set rules.

Different from micro-tasks in crowdsourcing, e.g. data labeling, software development tasks are usually too complex for an individual, thus developers prefer to join a team so as to ensure the productivity and quality of software development [4]. Our empirical study on Kaggle reveals the benefits of team collaboration. However, today’s CSD platforms like Kaggle provide no effective support for team formation. As far as we know, few efforts have been made to recommend teammates for crowdsourcing developers.

Team recommendation problem has been extensively studied in various fields. In collaboration learning, a large number of studies concentrate on assigning students into the suitable teams and get their skills promoted based on expertise-dissimilarity objective and expertise-gain objective [5][6]. In project development, some studies aim to optimize task outcomes by recommending suitable teams for tasks [7][8][9][10]. In employee training, some works try to assign employees to various projects with goals for employee
To verify the significance of our study, we conducted an empirical study on Kaggle. Our results show that developers can benefit from teamwork and they have great demand for teammate recommendation support in CSD platforms. Our analysis also shows that developers’ teammate preferences is largely affected by three factors including the closeness with teammates, expertise difference with teammates, expertise gain through collaboration. Second, we define the collaboration willingness of a developer as the weighted sum of the above three factors, where the weights define the developer’s personal teammate preferences. Then the personalized teammate recommendation problem is formulated as a problem of optimizing the collaboration willingness of a team. Third, we further provide methods to quantify the three factors, then present a linear programming based method to compute developers’ teammate preferences with their collaboration history. Finally, based on developers’ personal teammate preferences, we design a teammate recommendation approach with an approximation algorithm to maximize the collaboration willingness of candidate team.

The major contributions of this work are as follows:

- We identify the personalized teammate recommendation problem for crowdsourced software development by empirical study of Kaggle. As far as we know, few efforts have been made to study this issue in CSD.
- We propose a metric called collaboration willingness of a team to measure the possibility of successful team formation by considering developers’ personal teammate preferences, and provide methods to obtain the preferences and compute the collaboration willingness.
- We further give an approach to personalized teammate recommendation for crowdsourcing developers based on their personal teammate preferences, and conduct a set of experiments to evaluate the performance of our approach in comparison with other methods.

2 AN EMPIRICAL STUDY OF KAGGLE

To verify the significance of our study, we conducted an empirical study on Kaggle. We aim to answer the following three research questions (RQs):

- RQ1: Does team collaboration matter for developers in CSD?
- RQ2: Is it common to see developers’ fail in teammate-seeking without teammate recommendation?
- RQ3: Which factors affect developers’ willingness of collaboration? And do developers show personal preferences in choosing teammates in terms of those factors?

Details are available in Google Docs:

2.1 Data Preparation

Beyond the basic supports for competitions, Kaggle provides online communities including Discussion Community and Kernel Community, where developers can discuss issues of various topics and share ideas.

We crawled a dataset from Kaggle involving 275 competitions and 74,354 developers who have made 191,300 submissions from April 2010 to January 2018. We also crawled developers’ social data from kaggle’s communities.

2.2 Benefits of Team Collaboration

We performed a statistical analysis of developers’ competition rankings under different work modes based on the development history of the selected developers who have participated in more than \( K \) competitions. The numbers of developers selected are 988 and 327 when \( K \) is set to be 10 and 20. According to developers’ work mode, we divided their development history into individual-mode and teamwork-mode. We calculated the proportion of each developer’s completed tasks ranking their own Top-K development history under specified work mode, denoted as \( \text{Prop} \). Results are shown in Figure 1. The x-axis demonstrates Prop while the y-axis represents the accumulated ratio of developers. Figure 1 exposes that developers can obtain better rankings through teamwork. For instance, the left two points in curves of \( k = 10 \) indicate that, for about 60% of 988 developers, the competitions they participated in under teamwork-mode all rank top 10 in their own development history, while the proportion is only 14.88% when competitions are finished individually.

![Figure 1: Developers’ performance under two modes](http://dlw.cn/1IvdnhAj)
2.3 Phenomenon of Failing to Find Teammates
We crawled through the discussion table in our dataset to select discussions with term team(case insensitive) in their titles. 500 discussions considered to be posted for teammate-seeking purpose were selected. From discussion posters and commenters, we identified 1987 developers who have intention to team up. Surprisingly, only 825 developers participate in the competitions in the end, among them merely 415 developers work in teams. We can infer that only 20.9% of developers team up successfully, and 58.5% of developers give up participating which may be due to lack of teammates.

It is common to see developers’ fail in teammate-seeking, thus it is critical to provide teammate recommendation for developers.

2.4 Affecting Factors and Personal Teammate Preferences
We expended a further analysis into those 500 discussions. It turns out that 45% of the posters clearly state that they hope to gain expertise improvement through team collaboration and 77% of them call for teamates whose skill proficiency is similar to theirs. We can draw the conclusion that expertise gain from collaboration and expertise difference between teammates can affect developers’ collaboration willingness.

We sorted the 275 competitions in the dataset according to start date. The first 245 competitions, about 90% of the dataset, are used as history data. We selected 585 developers who have collaborated both in history data and the remaining 30 competitions to analyze their collaboration relations and found that 75% of these developers collaborate with their former teammates. The closeness between developers can be reflected by collaboration relations, so it is also an important factor affecting developers’ collaboration willingness.

From above, it can also be seen that developers put forward different requirements for teammates. Some discussion posters call for someone with similar skill proficiency to team up, some want to get their expertise promoted through teamwork, when some else want to collaborate with their former teammates. Therefore, developers have their personal teammate preferences, which our teammate recommendation approach should take into consideration.

3 PROBLEM FORMULATION
Inspired by the above empirical study, we propose to provide personalized teammate recommendation for developers. In this section, we first introduce several important concepts and then provide the formulation of our problem.

In CSD platforms, developers usually have expertise at different proficiency levels. Developers with more experiences will be more proficient. Let \( U = \{u_1, u_2, \ldots, u_n\} \) be the set of developers, and the whole skills required in tasks can be represented as a set \( S = \{s_1, s_2, \ldots, s_m\} \). We introduce the concept of Skill Proficiency to denote developers’ skill-level.

**Definition 1** (Skill Proficiency) Formally, the set of skills that developer \( u_i \in U \) knows is represented as a set \( Su(u_i) \subset S \). For each skill \( s_k \) that \( u_i \) masters (i.e., \( s_k \in Su(u_i) \)), we define the corresponding skill proficiency as \( Prof(u_i, s_k) = \psi_i^k \in [0, 1] \), to present how proficiently \( u_i \) master \( s_k \).

Task requestors usually pre-specify the required skills for tasks. Thus we elicit the concept of Task’s Required Skill Set.

**Definition 2** (Task’s Required Skill Set) For a task \( v \), a set of skills is needed to guarantee its accomplishment, which can be represented as set \( S(v) \). Tasks can be divided into multi-task and single-task tasks according to the amount of required skills. Specially, our study is only focus on single-task tasks currently, thus we regard \( S(v) = s_k \) in this paper.

As Section 2.4 shows, closeness with teammates, expertise difference and expertise gain through collaboration affect developers’ collaboration willingness and developers have their personal teammate preferences on these three factors. Personality, geography, language, etc. also influence developers’ collaboration willingness, but this paper does not take them into consideration. Given a developer team \( T = \{u_1, u_2, \ldots, u_l\} \) which is formed for \( s_k \)-required task \( v \), for any developer \( u_i \in T \), let \( C(u_i, T) \in [0, 1] \) denote the \( u_i \)’s closeness with members in \( T \), \( ED(u_i, T, s_k) \in [0, 1] \) denote \( u_i \)’s expertise difference in \( s_k \) with members, and \( EG(u_i, T, s_k) \in [0, 1] \) represent \( u_i \)’s expertise gain in \( s_k \) through teamwork in team \( T \). Developers’ collaboration willingness goes up with the increase of \( C(u_i, T) \) and \( ED(u_i, T, s_k) \), declines with the increase of \( ED(u_i, T, s_k) \).

\( PreC = \{(a_i, \beta_i, \gamma_i) | i \in [1,n]\} \) is used to represent the teammate preferences of developers in \( U \). The formula of \( u_i \)’s collaboration willingness in team \( T \) can be formalized as:

\[
W(u_i, T) = a_i + C(u_i, T) + \beta_i + \gamma_i(1 - ED(u_i, T, s_k)) + \gamma_i*EG(u_i, T, s_k),
\]

\[
s.t. u_i \in T, a_i + \beta_i + \gamma_i = 1, \text{and}
\]

\[
a_i, \beta_i, \gamma_i, C(u_i, T), ED(u_i, T, s_k), EG(u_i, T, s_k) \in [0, 1],
\]

where \( a_i, \beta_i, \gamma_i \) indicate the influence degree of corresponding factors on developers’ collaboration willingness. The method to obtain the value of \( PreC \) and representations of \( C(u_i, T), ED(u_i, T, s_k) \), \( EG(u_i, T, s_k) \) will be introduced in the following section.

Based on the definitions introduced above, we define the personalized teammate recommendation problem as follows:

**Definition 3** (Personalized Teammate Recommendation Problem) Assuming that there exists a developer set \( U = \{u_1, u_2, \ldots, u_n\} \) and task \( v \)’s required skill set \( \{s_k\} \), given a developer \( u_i \) who seeks for \( N \) teammates for task \( v \), personalized teammate recommendation...
problem studies how to recommend $u_i$ the most suitable $N$ teammates with high collaboration willingness in consideration of each developer’s personal teammate preferences.

**Definition 4 (Collaboration Willingness of A Team)** Whether the team can be formed is based on each member’s willingness. We should not only take consideration of $u_i$’s collaboration willingness, but also the willingness of all the other members. Therefore, we should ensure that the collaboration willingness of each member is no less than a threshold $threshold$, which means the minimum collaboration willingness that each member in a team should satisfy. $threshold$ can be obtained from developers’ collaboration history and will be introduced in Section 4.2. Meanwhile, we need to maximize the average value of all members’ collaboration willingness, which is defined as the collaboration willingness of team $T$:

$$W(T) = \frac{\sum_{u_j \in T} W(u_j, T)}{|T|}.$$  

Formally, the personalized teammate recommendation for developer $u_i$ aims at obtaining the optimal team $T^*$:

$$T^* = \arg \max_T W(T) = \arg \max_T \frac{\sum_{u_j \in T} W(u_j, T)}{|T|} \quad \text{s.t.} \quad T \subseteq \mathcal{U}, |T| = N + 1, u_i \in T, W(u_i, T) \geq threshold$$

and $T^* \setminus \{u_i\}$ is the set of recommended teammates for $u_i$.

## 4 PROPOSED METHOD

We give our recommendation framework in Figure 3. In order to quantify the collaboration willingness of team, we model the three factors which dominate developers’ collaboration willingness, then characterize developers’ personal teammate preferences and get their minimum collaboration willingness by analyzing their collaboration history. Finally, based on collaboration willingness, we propose an approximation algorithm to provide teammate recommendation for developers.

![Figure 3: Recommendation Framework](image)

### 4.1 Factor Modeling

In this part, we introduce concrete representations of the three factors which dominate developers’ collaboration willingness.

**Factor 1 (Closeness with Teammates $C(u_i, T)$)** In CSD platforms, developers usually have collaboration relations and social interactions with each other, based on which we can measure the integrated closeness between developers.

**Definition 5 (Collaboration-Relation based Closeness $C_1(u_i, u_j)$)** Developers usually collaborate for tasks, and frequent collaboration makes developers closer. We introduce Collaboration-Relation based Closeness, i.e., $C_1(u_i, u_j) \in [0, 1]$, to represent how close $u_i$ is to $u_j$ under collaboration relations. It’s worth knowing that $C_1(u_i, u_j) \neq C_1(u_j, u_i)$. Let weighted undirected graph $G^R = (\mathcal{U}, R^R)$ denote collaboration relations between developers, in which $U$ is the set of developers and $R^R$ represents collaboration relations between them. $\forall (u_i, u_j)^R \in R^R$, the weight of this edge, $w((u_i, u_j)^R)$, represents the collaboration times between $u_i$ and $u_j$. How close $u_i$ is to $u_j$ depends on the proportion of collaboration relations between $u_i$ and $u_j$ in $u_i$’s total collaboration relations:

$$C_1(u_i, u_j) = \frac{\sum_{(u_i, u_j)^R \in R^R} w((u_i, u_j)^R)}{\sum_{u_k \in \mathcal{U}} w((u_i, u_k)^R)}$$

**Definition 6 (Social-Relation based Closeness $C_2(u_i, u_j)$)** Many CSD platforms provide online communities where developers can discuss and share ideas. The more frequent the social interactions are, the more closer developers are. Social-Relation based Closeness, $C_2(u_i, u_j) \in [0, 1]$, is used to represent how close $u_i$ is to $u_j$ under social relations. It’s also worthy to know that $C_2(u_i, u_j) \neq C_2(u_j, u_i)$. Let weighted undirected graph $G^b = (\mathcal{U}, R^b)$ denote social relations between developers, in which $U$ is the set of developers and $R^b$ represents social relations between them. $\forall (u_i, u_j)^b \in R^b$, the weight of this edge, $w((u_i, u_j)^b)$, represents the social relation times between $u_i$ and $u_j$. How close $u_i$ is to $u_j$ depends on the proportion of social relations between $u_i$ and $u_j$ in $u_i$’s total social relations, and can be represented as follows:

$$C_2(u_i, u_j) = \frac{\sum_{(u_i, u_j)^b \in R^b} w((u_i, u_j)^b)}{\sum_{u_k \in \mathcal{U}} w((u_i, u_k)^b)}$$

We denote the integrated closeness between $u_i$ and $u_j$ as:

$$C(u_i, u_j) = \omega \times C_1(u_i, u_j) + (1 - \omega) \times C_2(u_i, u_j),$$

where $\omega \in [0, 1]$ represents the weight of term $C_1(u_i, u_j)$. It is worthy to note that $C(u_i, u_j) \neq C(u_j, u_i)$.

$C(u_i, T)$ is defined as the average value of $u_i$’s integrated closeness with other members in $T$ and represented as follows:

$$C(u_i, T) = \frac{\sum_{u_j \in T \setminus \{u_i\}} C(u_i, u_j)}{|T| - 1}.$$

**Factor 2 (Expertise Difference with Teammates $ED(u_i, T, s_k)$)** Expertise difference weakens developers’ willingness to team up. Without loss of generality, the expertise difference in skill $s_k$ between $u_i$ and $u_j$ is defined as $D^k(u_i, u_j) = |s_k^i - s_k^j|$. $ED(u_i, T, s_k)$ is the average value of $u_i$’s expertise difference in $s_k$ with other members in $T$ as follows:

$$ED(u_i, T, s_k) = \frac{\sum_{u_j \in T \setminus \{u_i\}} D^k(u_i, u_j)}{|T| - 1}.$$
Factor 3 (Expertise Gain Through Collaboration \(EG(u_i, T, s_k)\)) Many developers want to promote their expertise through teamwork. Inspired by Jiawei Zhang et al.\cite{11}, we define a gain function to measure the improvements of developers’ skill proficiency. First, we introduce the skill proficiency of team \(T\), represented as \(Prof(T, s_k)\), and then give representation of developers’ expertise gain from team.

Definition 7 (Proficiency of Team (\(Prof(T, s_k)\))) Given a team of developers \(T = \{u_1, u_2, ..., u_l\}\) with proficiency levels \(s_k\) of task \(u_i\)’s required skill \(s_k\). Team’s proficiency of \(s_k\) can be defined as follows:

\[
Prof(T, s_k) = 1 - \prod_{u_i \in T} (1 - \frac{\alpha}{\gamma})
\]

The overall team proficiency is an important characteristic of teams, which denotes both the team’s competitiveness and the space for members’ expertise gain. Formally, when collaborating with team \(T\), the expertise gain for \(u_i\) in \(s_k\) can be represented as

\[
EG(u_i, T, s_k) = Prof(T, s_k) - Prof(u_i, s_k).
\]

4.2 Parameter Estimation

According to Section 3, we need to obtain the minimum collaboration willingness threshold and developers’ teammate preferences \(PreC = [(\alpha, \beta, \gamma)]\).

Supposing in developers’ collaboration history, each team member has a high willingness to collaborate because they team up on a voluntary basis. Let \(R_i = \{T_1, T_2, ..., T_k\}\) represent \(u_i\)’s collaboration history, and \(s_{T_j}\) represent the skill required by the \(T_j\)’s corresponding single-skill task, where \(T_j \in R_i\). We learn threshold and \(PreC\) at the same time as Algorithm 1 describes. As threshold and \(PreC\) are correlative, the learning process is iterative. With a given threshold, triple \( PreC = [(\alpha, \beta, \gamma)]\) is obtained by maximizing the sum of \(u_i\)’s collaboration willingness of all his collaboration history and guaranteeing that \(u_i\)’s collaboration willingness in each team is no less than threshold, as Equation (2) demonstrates. With a given threshold, Equation (2) is a typical linear programming problem, which obtains \(PreC\) based on which we obtain the value of \(PreC\).

\[
\begin{align*}
(\alpha, \beta, \gamma) & = \arg \max_{(\alpha, \beta, \gamma)} \sum_{T \in R_i} W(u_i, T) \\
\text{s.t.} & \forall T \in R_i, W(u_i, T) \geq \text{threshold, and} \\
& \alpha + \beta + \gamma = 1, \alpha, \beta, \gamma \in [0, 1]
\end{align*}
\]

4.3 Objective Function and Algorithm

After getting \(PreC\), developers’ teammate preferences are characterized. Combined with Factor 1-3, for any given team \(T\), \(W(u_i, T)\), we can calculate \(u_i\)’s collaboration willingness \(W(u_i, T)\). Then, we can get the concrete representation of our objective function denoted as Equation (1). To recommend the most suitable teammates for developer, a straightforward solution is traversing all candidates and find the required \(N\) teammates, which is not feasible as the computational complexity is \(O(nN)\) where \(n\) stands for the number of candidates and \(N\) stands for the number of needed teammates.

We propose an approximation algorithm \(PTR\) to solve the problem, and the pseudo code is presented in Algorithm 2. \(PTR\) is a greedy algorithm: given a teammate seeker \(u_i\), \(PTR\) traverses the candidate set \(U \setminus \{u_i\}\) to find a teammate \(u_j\) where \(\exists u_j \in U \setminus \{u_i\}, W'(\{u_i, u_j\}) > W'(\{u_i, u_j\})\); then on team \(\{u_i, u_j\}\), \(PTR\) continues traversing the candidate set \(U \setminus \{u_i, u_j\}\) to find a teammate \(u_k\) where \(\exists u_k \in U \setminus \{u_i, u_j\}, W'(\{u_i, u_j, u_k\}) > W'(\{u_i, u_j, u_k\})\); The process continues until the number of recommended teammates reaches \(N\). The computational complexity of \(PTR\) is only \(O(n + N)\).

Algorithm 1 Obtaining Teammate Preferences \& threshold

Require: \(U = \{u_1, u_2, ..., u_n\}\), \(R = \{R_1, R_2, ..., R_n\}\)
Ensure: \(PreC = [(\alpha, \beta, \gamma)] \in [1, n]\)
1: threshold = 0.02
2: while True do
3: \(\forall i \in [1, n]\), obtain \((\alpha, \beta, \gamma)\) by Equation (2)
4: if \(\exists i \in [1, n] \cap \alpha, \beta, \gamma\) by Equation (2)
5: Break
6: threshold += step
7: \(PreC = [(\alpha, \beta, \gamma)] \in [1, n]\]
8: end while
9: threshold -= step
10: \(PreC = [(\alpha, \beta, \gamma)] \in [1, n]\)
11: \(PreC = [(\alpha, \beta, \gamma)] \in [1, n]\)
12: return \(PreC\), threshold

Algorithm 2 Personalized Teammate Recommendation

Require: Teammate seeker \(u_i\) who requires \(N\) teammates
Ensure: Teammate candidates \(U = \{u_1, ..., u_k\} \setminus \{u_i\}\), Task \(v\)’s required skill \(s_k\)
Require: threshold, \(PreC = [(\alpha, \beta, \gamma)] \in [1, n]\)
Ensure: Recommended teammates set \(\{u'_1, ..., u'_n\}\)
1: \(T=[u_i]\), teamsize=1
2: while teamsize < \(N+1\) do
3: \(u' \in U \setminus \{u_i\}\), \(W'(\{u', u_i\}) > W'(\{u', u_i\})\)
4: \(u'_k \in U \setminus \{u_i, u_j\}\), \(W'(\{u_i, u_j, u_k\}) > W'(\{u_i, u_j, u_k\})\)
5: teamsize += 1
6: end while
7: return \(T \setminus \{u_i\}\)

5 EXPERIMENTS

In this section, we introduce the dataset and experiment settings. Then experiment results are presented and analyzed.

5.1 Dataset Description

We used a dataset collected from Kaggle, which involves 275 competitions and 74,354 developers from April 2010 to January 2018. This dataset contains 19,300 competition records of developers, from which we can obtain developers’ skill proficiency and collaboration relations. We also crawled their social data from Kaggle’s Discussion Community and Kernel Community to mine their social
relations. 488,722 pairs of developers are linked by collaboration or social connections. The dataset used in our experiments is available in GitHub\(^1\).

5.2 Experiment Settings

Tasks in Kaggle are all data science competitions, and the skill set involved is \( S = \{ s_i \} \) where \( s_i \) means Data Science. We leveraged Kaggle’s score system to get developers’ scores based on their development history. Through linear normalization, we obtained their skill proficiency \( \text{Prof}(u_i, s_j) \in [0, 1] \). Based on the collaboration and social relations, we got the integrated closeness \( C(u_i, u_j) \) between developers \((\omega, \text{weight of Collaboration-Relation based Closeness}, \text{is set to be 0.7})\).

3,808 developers whose collaboration times are no less than 2 were selected in our experiments. We first obtained their teammate preferences \( \text{PreC} \) and threshold as introduced in Section 4.2. Afterwards, we compared our work with other methods. The comparison methods include:

- Closeness First(CF): CF takes the closeness between developers into consideration during recommendation process, and the method is widely used in previous works [8]. Skill proficiency is not considered in CF.
- Expertise Gain First(EGF): EGF aims at maximizing team members’ expertise improvement, and the method is adopted in the area of collaboration learning [6].
- Expertise Diff First(EDF): EDF aims to minimize members’ expertise difference without considering the closeness between developers and expertise gain.
- Optimal: This method traverses the developer set to find the theoretically optimal solution.

All the experiments were performed on a desktop computer with Core i7-6700 3.4GHZ and 8GB memory.

5.3 Experiment Results

We got the recommended results of our method and the comparison methods respectively for \( N = 1, 2, 3 \), and calculated the average value of recommended teams’ collaboration willingness for each method. Table ?? demonstrates the results.

As shown in the table, our approach can recommend teams with higher collaboration willingness and outperforms the existing methods (except for Optimal), because our method takes three factors and personal teammate preferences into consideration, while CF, EGF and EDF only consider one factor that affects collaboration willingness respectively. The results of Optimal demonstrate the theoretically maximum collaboration willingness. However, Optimal has the highest computational complexity \( O(n^N) \), where \( n \) is the candidates number and \( N \) is the number of required teammates) to get the recommendation results. It takes hours to get all recommendation results for 3808 developers when \( N=2 \), while other methods only require several minutes. For all 3,808 developers and \( N=3 \), Optimal has run for 30 days till now, but has not finished. When \( N=1 \), the result of Optimal is the same with our method because our method traverse all candidates to recommend one teammate for a developer. It is noteworthy that our method can obtain similar results as Optimal, and the computational complexity is only \( O(n + N) \).

6 CONCLUSION

In this paper, we identify the personalized teammate recommendation problem for crowdsourced software development by an empirical study of Kaggle. Then, we propose an approach to solve the problem by maximizing the collaboration willingness of a team based on developers’ personal teammate preferences. Experiments conducted on Kaggle dataset verifies the effectiveness of our approach.

In the future, we will study team recommendation involving multiple development skills and try to solve the cold start problem when providing recommendation for novices.

ACKNOWLEDGMENT

This work was supported partly by National Key Research and Development Program of China under Grant No.2016YFB1000804, partly by National Natural Science Foundation under Grant No.(61421003, 61702024) and partly by SKLSDE Lab under Grant No. SKLSDE-2017ZX-14. Hailong Sun is the corresponding author of this paper.

REFERENCES