Theoretical Computer Science 857 (2021) 43–58

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A unified task recommendation strategy for realistic mobile crowdsourcing system ☆

Zhiyao Li [a], Bosen Cheng [a], Xiaofeng Gao [a,*], Huai Chen [b], Guihai Chen [a]

[a] Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
[b] Tencent Inc. Shenzhen, China

ARTICLE INFO

Article history:
Received 14 April 2020
Received in revised form 22 August 2020
Accepted 15 December 2020
Available online 6 January 2021

Keywords:
Crowdsourcing
Recommendation system
K-medoids

ABSTRACT

A well-designed task recommendation framework aims to protect the data quality as well as increase the task execution results. However, current crowdsourcing systems ignore the fact that there are few duplicate task expectations because of the budget limitation in realistic conditions. Besides, a practical crowdsourcing system needs to recommend new tasks without previous knowledge about the concrete task content due to short task lifespan. Thus, most of the existing studies are not applicable due to the idealized assumptions. In this paper, we formally define the problem and prove it is NP-Hard. For the problem, we design a unified task recommendation system for realistic conditions to address the mentioned problems, Pioneer-Assisted Task RecommendatiON (PATRON) framework. The framework first selects a set of pioneer workers to collect initial knowledge of the new tasks. Then it adopts the *k*-medoids clustering algorithm to split the workers into subsets based on the worker similarity. Cluster selection and worker pruning provides accurate and efficient recommendations that satisfy the valid recommendation requirements from requesters. Finally, we conducted our experiments based on real datasets from a famous Chinese crowdsourcing platform, Tencent SOHO. The experimental results show the efficiency and accuracy of PATRON compared with three baseline methods from several perspectives, such as recommendation success rate and recommended worker quality.

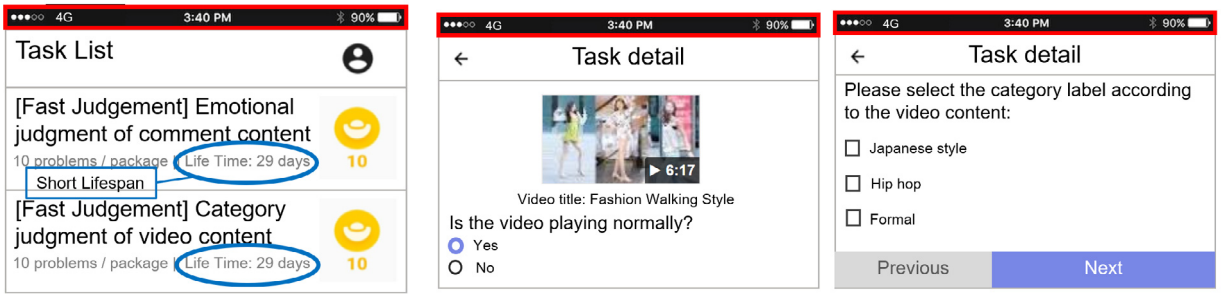
1. Introduction

Recently, crowdsourcing has been a hot topic. It refers to an efficient system that obtains services from a large, relatively open, and often rapidly evolving group of internet users. In general, a crowdsourcing platform has the capability to gather information that is hard for authorities to disseminate. There are different types of crowdsourcing models, focusing on real-time air quality [1], traffic crowdedness [2], and geographic information in disaster areas [3]. In practice, crowd workers in

☆ This work was supported by the National Key R&D Program of China [2020YFB1707903], the National Natural Science Foundation of China [61872238, 61972254], and the Huawei Cloud [TC20201127009]. The authors also would like to thank Yuchen Xia and Mo Chi for their contributions on the early versions of this paper. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the funding agencies or the government.

* Corresponding author.
E-mail addresses: lzy_budding@sjtu.edu.cn (Z. Li), unchain@sjtu.edu.cn (B. Cheng), gao-xf@cs.sjtu.edu.cn (X. Gao), spiritchen@tencent.com (H. Chen), gchen@cs.sjtu.edu.cn (G. Chen).

https://doi.org/10.1016/j.tcs.2020.12.034
0304-3975/



(a) Short Lifespan. (b) Crowdsourcing Task with Video.

Fig. 1. Limitations of realistic crowdsourcing platforms, with Tencent SOHO [8] as an example. The tasks' lifespan is relatively short (the task shown in Fig. 1(a) has a lifespan for about 1 month), and more video or audio based crowdsourcing tasks raise the difficulty of analyzing the tasks' contents. (the task shown in Fig. 1(b) requires the worker to choose the corresponding tags for the video in the link). The red frames imply that it is a mobile application. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

crowdsourcing platform are not controllable and predictable in terms of quality and the will of workers when comparing to expert workers, which impact the data quality and production rate. Correspondingly, effective task recommendation for suitable workers is a pressing problem to improve the data quality.

Numerous researchers have made attempts to improve the stability and efficiency of the crowdsourcing system [4–7]. However, the crowdsourcing systems in real world are quite different from those researches with the assumptions. Generally, there are three major challenges in practical crowdsourcing platforms.

1. The first challenge is that a real crowdsourcing platform might not allocate the same quests to multiple workers due to budget limitation, leading to **no duplicate** answers available for traditional truth inference approaches.
2. The second challenge is that most task categories requesting recommendations on an actual platform are **original tasks without prior knowledge** because of the **relatively short** lifespan of each task category (as shown in Fig. 1(a)). This means that when recommending a certain task to a worker, the worker's history data of the task category should be analyzed.
3. The third challenge is that the actual platform includes many **video-based and audio-based quests** (an example is shown in Fig. 1(b)), making it hard to analyze its specific content comparing with traditional text-based quests. But most previous works just focus on text-based quests.

Taking Tencent SOHO [8] as an example shown in Fig. 1, it is a famous Chinese crowdsourcing platform that provides massive tasks, including continuous voice transfer, picture annotation, text annotation, video annotation, etc. In Fig. 1, the red frame shows that this is a mobile application. Although there is also a web-based version, most users prefer the mobile version as it is more convenient and easier to work on. On this platform, there is no duplicate answer since one quest will be allocated to only one worker. Besides, Fig. 1(a) and Fig. 1(b) show that tasks on SOHO have a short lifespan and vary in types.

In this case, current crowdsourcing platforms cannot satisfy the actual requirements. Thus, we aim at designing a more general and practical crowdsourcing framework, which can be deployed economically without losing much accuracy. In this paper, we consider Tencent SOHO as the target crowdsourcing system and propose a unified, non-content-related task recommendation framework, Pioneer-Assisted Task RecommendatiON (PATRON). The PARTON can effectively address the existing issues of SOHO and enrich the crowdsourcing task recommendation theory. Moreover, our work might be promoted and applied to other crowdsourcing platforms and improve the effectiveness of task recommendation in the future.

The PATRON is an iterative clustering-based task allocation system that recommends new task categories to workers with maximum recommendation efficiency, meeting the required valid recommendation for a new task category. The objective of this framework is to increase the recommendation success rate as well as the worker's willingness to accept requests using as few recommendations as possible to save costs. In the framework, we first select a group of workers as the pioneers and allocate the new task categories for them to collect initial knowledge. Then we use a clustering algorithm to get similar workers as groups. We achieve the expected valid recommendation requirement by selecting clusters and pruning low-will workers. In extensive experiments, the proposed framework PATRON performs better than baseline methods in terms of accuracy and efficiency. Our main contributions are as follows:

1. We formally define the Accurate Recommendation Problem (ARP) to model the actual task recommendation problem on the crowdsourcing platform with realistic limitations. The worker experience is analyzed by collecting the execution history and apply different confidence on different types of quest packages. Then the willingness to accept task requests of workers is modeled by their working experience on the platform.

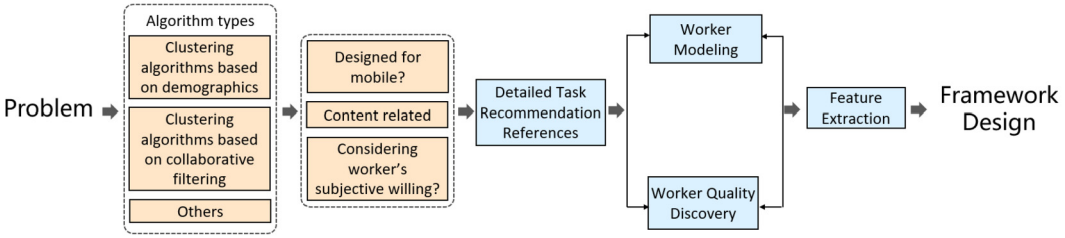


Fig. 2. Research Framework for Unified Crowdsourcing System.

2. We design the iterative clustering-based task allocation system, PATRON framework. We select a set of pioneer workers and let them execute new task categories' test packages to collect their initial knowledge. We use these workers as initial centers to split all workers into clusters with similar features. The worker pruning process then abandon inefficient recommendations. Besides, the estimated quality of workers is the additional information for doing recommendations in the iterations.
3. We conduct several experiments on a dataset from Tencent SOHO, an realistic crowdsourcing platform. Results show that PATRON returns an overall better result comparing to the other strategies in terms of recommendation success rate and recommended worker quality. We also find out the stability of the system on certain parameters.

The rest of this paper is organized as follows. Related work is discussed in Sec. 2. We formally define the Accurate Recommendation Problem (ARP) and prove the problem is NP-Hard in Sec. 3. In Sec. 4, we describe the Pioneer-Assisted Task Recommendation (PATRON) framework specifically with technical discussions and proofs. In Sec. 5, we conduct some evaluations based on real trace data from Tencent SOHO to show the efficiency of our proposed framework. In the end, we conclude our work in Sec. 6.

2. Related work

In this section, we review many recent studies about task recommendation. The overall workflow for reviewing the related works is shown in Fig. 2. Table 1 shows the model and algorithm designs for some of these works. As emphasized in Sec. 1, we value certain features for the realistic condition. Clustering algorithms are simply divided into 3 types: algorithms based on demographics, algorithms based on collaborative filtering and others. Since the actual usage of the system is on mobile devices, we care whether the crowdsourcing system is related to mobile. Since we aim to build a unified system that is independent of the specific task contents, we also check whether the crowdsourcing system is related to the concrete content of the tasks. Besides, the influence of the worker's willingness also matters for the design of our crowdsourcing system. Then, the task recommendations are reviewed in detail, and the worker modeling and worker quality discovery techniques are studied as well. In general, we extract useful information from those researches.

2.1. Task recommendation

The quality of workers (described by their work output) and correctness of the tasks are highly dependent on task recommendation algorithms. A proper task recommendation can stimulate the workers finishing the tasks, and attract them to participate in task execution.

[4] investigates a new condition about mobile crowdsourcing in opportunistic mobile social networks, and proposes a heuristic approximation algorithm distributing tasks using minimum forwarders. [9] offers a collaborative-task assignment algorithm (Collaborative-Task Assignment Algorithm) to minimize the idle time of workers. However, it does not include situations where workers are interrupted when completing tasks.

[10] investigates the gathering worker groups in a stream of complex tasks, and compares three strategies on different psychological and quality issues. [6] focuses on the spontaneously-formed mobile social networks. It also minimizes operating cost and completion time by solving two task scheduling problems. [11] proposes two Top-N recommendation algorithms for crowdsourcing systems by extending matrix factorization and kNN. Then it outputs the most suitable tasks to workers and identifying best workers for the requester.

[12] proposes a new pairwise crowdsourcing model to combine crowdsourcing and Top-K problems in uncertain databases to select the best pairing. However, it is necessary to apply this method to the relevance of other probabilistic problems such as the result of items. [5] designs a system that allows task publishers and workers to give feedback to each other, estimates the similarity between workers and the match between tasks. It better distributes tasks to workers, but it does not consider time for workers to perform tasks.

[13] proposes an online profile learning framework that considers whether the task and the user's common characteristics match for distribution. However, it doesn't consider the different behaviors of multiple tasks, and the framework also needs to improve the management of workers' profiles in crowdsourcing activities with different workers and task characteristics. [14] proposes a two-layer data representation scheme, which expands the minimum mean square error and

Table 1
Related Works: Algorithms types (3 types) - clustering algorithms based on demographics, clustering algorithms based on collaborative filtering and others. Mobile related or not - whether the crowdsourcing system is related to mobile. Content related or not - whether the crowdsourcing system is based on the concrete content of the tasks. Worker willing related or not - whether the crowdsourcing system considers the influence of worker's willingness. (The order of references is in the order in which they appear in the text.)

Related work	Algorithm type			Mobile related or not	Content related or not	Worker willing related or not	Key features
	Based on demographics	Based on collaborative filtering	Others				
ICNC 2018 [4]	√			√			Crowdsourcing in opportunistic mobile social network
AINA 2018 [5]		√				√	Feedback-based crowdsourcing
ICSOC 2018 [6]			√	√			Task scheduling in mobile social networks
ICSOC 2018 [7]	√					√	Quality inference based task assignment
IEEE ICC 2018 [9]		√		√			–
HCOMP 2018 [10]		√				√	Skill-and-stress-aware task assignment
ACM TOIS 2019 [11]			√				Efficient learning-based recommendation
ICDE 2018 [12]		√			√		Pairwise crowdsourcing,
SEBD 2018 [13]		√			√		–
TOIS 2019 [14]		√				√	Two-layer data representation scheme
CoRR 2018 [15]		√				√	Crowdsourcing based on learning preference and reliability
ICSOFT 2018 [16]		√				√	Social multi-agent cooperation system
TCS 2019 [17]			√				Quality-aware online task assignment
TCS 2019 [18]			√			√	Budget-limited task recommendation
AAAI 2014 [19]			√			√	–
PVLDB 2016 [20]			√		√		Domain-aware crowdsourcing system,
ICWWW 2016 [21]			√		√		Optimized task assignment in knowledge-intensive crowdsourcing
IEEE Access 2019 [22]			√			√	Worker quality management
IEEE TNNLS 2019 [23]		√				√	Bilayer collaborative clustering
CoRR 2018 [24]			√	√			Budget feasible peer graded mechanism
SIGMOD 2016 [25]			√			√	Uniform worker settings
IJRIS 2019 [26]			√			√	–
IEEE IoT-J 2019 [27]			√			√	Optimal selection balancing worker utility
SIGKDD 2015 [28]	√				√		Grained truth discovery for crowdsourcing
IEEE ICC 2016 [29]			√	√	√		–
IEEE TMC 2017 [30]		√		√			A randomized auction for smartphone crowdsourcing
IEEE TON 2016 [31]			√	√	√	√	Incentive mechanisms for smartphone crowdsourcing
TAAI 2015 [32]			√				–
EDBT 2017 [33]			√			√	Motivation-aware task assignment

Bayesian personalized ranking and combines them to find Top N Tasks (TNT) recommendations for a worker and find the Top N workers recommendations (TNW) for a certain task so that it can generate most suitable and personalized recommendations. The disadvantage is that this study relies heavily on the assumption that tasks have been classified. [15] proposes a framework called MAB (Multi-Armed Bandit) to learn the preferences and reliability of workers in crowdsourcing tasks, thereby recommending tasks more efficiently. This paper hypothesizes that workers' preferences and reliability have nothing to do with different kinds of tasks. [16] establishes an agent-based architecture to deal with different services and tasks, especially individual planning and distributed task allocation. And it proposes a method to solve the multi-task assignment problem in Multi-Agent Systems. The disadvantage is that it cannot cope with larger state-action spaces.

[17] considers dividing tasks into clusters (i.e., topics) based on workers' behaviors and propose two online task assignment mechanisms that dynamically assign each incoming worker a set of tasks where he can achieve the maximum expected gain or maximum expected and potential gain. [18] considers the problem in which it is allowed to spend a given

budget to create new links so as to suggest a bounded number of possible persons to whom become a friend in order to maximize the influence of a given set of nodes.

2.2. Worker modeling

In order to control the quality of data collected from workers and give good workers more tasks, we need to model and keep track of the workers' quality. For example, [19] regards the worker quality as a single value, indicating the probability the worker producing a qualified answer or data. To illustrate the workers' performance and skills from different angles, [20] further splits the worker quality into specified ratings upon different task types to track the performance more precisely, while [21] puts an effort on rating the workers' ability on different latent topics or skills, so as to give tasks to the workers satisfying their knowledge reserves as much as possible. [22] proposes a system based on statistical quality control to regulate workers' performance. The framework is applied within the scope of a contact center, where requester feedback is used as a crowd to assist in deduplication detection. [23] proposes a novel Bi-Layer Collaborative Clustering (BLCC) method for the label aggregation in crowdsourcing. It uses two clustering algorithms for each layer and then outputs the results by integrating two layers.

2.3. Worker quality discovery

Learning the crowd workers clearly and efficiently will be a great assistance in avoiding resource loss while satisfying and incentivizing workers. [5] allows task requesters and workers to provide feedbacks to task executors or co-workers, and proposes a heuristic task assignment mechanism to distribute skillful workers to all collaborative tasks averagely. [24] determines the unknown quality of IoT devices by peer grading, and crafts a truthful budget feasible task allocation mechanism attaining a threshold quality. [25] jointly estimates the tasks' true answers and workers' quality by filtering, designs an algorithm considering all mappings from tasks to true answers and finds the maximum likelihood. [26] proposes a general worker quality evaluation algorithm that can be applied to any critical tasks without wasting resources. Realizing the evaluation algorithm in the Hadoop platform using MapReduce parallel programming is also involved. [27] designs a workload distribution strategy that can balance the quality of work and the benefits of the platform, which turns the problem into a multi-objective nonlinear programming problem.

3. Problem definition

In this section, we will formally define the Accurate Recommendation Problem (ARP) on the realistic crowdsourcing platform. Firstly we will give the definition of crowdsourcing task category and packages in Definition 1 and Definition 2. The Table 2 is used to summarize the meaning of all variables mentioned in Section 3 and Section 4.

Definition 1 *(Crowdsourcing task category). A crowdsourcing task category* $T = (P, v)$ *consists of a number of task packages* $P = \{p_1, p_2, \cdots, p_{|P|}\}$ *under the same execution rule given by it. The workers accept task packages and work on them. When they complete a task package belonging to the category, the reward* v *will be given to them. The collection of all history crowdsourcing task categories is* $\mathbf{T} = \{T_1, T_2, \cdots, T_n\}$*, while the collection of new task categories is* $\mathbf{T}^N = \{T_1^N, T_2^N, \cdots, T_l^N\}$.

Definition 2 *(Task packages). Each Task package* p *consists of a number of tasks for workers to work on. There are three types of packages, which can be identified by the type function* $type(p)$. ***Test*** *packages are packages with known truth labels for all tasks. For* ***inspector*** *packages, a portion* $0 < \Gamma(p) < 1$ *of quests will be inspected manually to estimate the package's accuracy. For* ***normal*** *packages, no operations are applied, and the package accuracy is not measured.*

We define the workers in the realistic crowdsourcing platform in Definition 3.

Definition 3 *(Crowdsourcing workers). The set of crowdsourcing workers* $W = \{w_1, w_2, \cdots, w_m\}$*. Each worker* w_j *has an execution history set* $H_j = \{h_1^j, h_2^j, \cdots, h_{|H_j|}^j\}$*, listing his/her previous task execution records on the crowdsourcing platform. Each execution record* $h_k^j = \langle q_k^j, a_k^j \rangle$*, where* q_k^j *is the task package the worker worked on, and* a_k^j *is the accuracy when the package is a test or inspector package.*

It is straightforward that if a worker always produces high accuracy when executing tasks on a certain category, we think that the worker is familiar with it. Also, if a worker executes lots of task packages on a certain task category, we think that the worker will gradually gain experience and familiarity on it. Thus, we can define the worker familiarity to the tasks according to the observation above in Definition 4. Note that, as the inspector packages have only a portion of tasks checked for correctness, the measurement confidence needs to be discounted according to the inspected proportion $\Gamma(p)$. Moreover, as normal packages' accuracy are not measured, they can only be referred by stacking up to show the workers' execution frequency. Therefore, we set accuracy of normal packages to 50%, and confidence to 0.1.

Table 2
Summary of variables and abbreviations.

Variable and abbreviation	Description
$T = (P, v)$	Crowdsourcing task category
$\mathbf{T} = \{T_1, T_2, \cdots, T_n\}$	Task category set of size n
$P = \{p_1, p_2, \cdots, p_{\|P\|}\}$	Set of task packages
v	Reward when completing a task package
$type(p)$	Function that returns the type of the package
$\Gamma(p)$	Portion estimating the package's accuracy
$W = \{w_1, w_2, \cdots, w_m\}$	Crowdsourcing worker set of size m
$H_j = \{h_1^j, h_2^j, \cdots, h_{\|H_j\|}^j\}$	Execution history set of worker w_j
$h_k^j = < q_k^j, a_k^j >$	History execution record in set H_j with index k
q_k^j	Task package of history record h_k^j
a_k^j	Accuracy the package of history record h_k^j
$x \in \{x^{test}, x^{inspector}, x^{normal}\}$	Type indicator
$S(w_i, w_j)$	Similarity between worker w_i and w_j
$P_i = \{w_{i1}, w_{i2}, \cdots, w_{iz}\}$	Pioneer worker set of task T_i of size z
$R_i = \{(w_{i1}, a_{i1}), (w_{i2}, a_{12}), \cdots, (w_{iz}, a_{iz})\}$	The result collection of the pioneer worker set P_i
$D(w)$	Acceptance willingness of worker w
G_i	Valid recommendation requirement of task category T_i
C	Recommendation count upper bound
d_{ij}	Recommendation result of worker w_j in task category T_i
$S = \{e_1, e_2, \cdots, e_m\}$	Set of positive integers
OPT	The optimal solution
ARP	Accurate Recommendation Problem
PATRON	Pioneer-Assisted Task RecommendatiON

Definition 4 *(Worker familiarity vector). Give a task category T_i and a worker w_j. The familiarity of w_j on T_i, $f(T_i, w_j)$, is defined as Eqn. (1). x_i^j is the type indicator, if $type(q_i) = j$, then $x_i^j = 1$, otherwise $x_i^j = 0$,*

$$f(T_i, w_j) = \sum_{k=1}^{|H_j|} \sum_{q_k^j \in T_i} (a_k^j \cdot x_i^{test} + a_k^j \cdot x_i^{inspector} \cdot \Gamma(q_k^j) + 0.5 \cdot x_i^{normal} \cdot 0.1). \tag{1}$$

We collect the familiarity of worker w_j to each task category in $\mathbf{T}$, and form the worker familiarity vector $\mathbf{f}_j = \langle f_{1j}, f_{2j}, \cdots, f_{nj} \rangle$, where $f_{ij} = f(T_i, w_j)$ $(1 \le i \le n)$.

Worker familiarity vector contains the information of quality and quantity of history task execution, and can be regarded as the workers' preference and expertness in each task category. Thus we can define the similarity between different workers as shown in Definition 5.

Definition 5 *(Worker similarity). The similarity $S(w_i, w_j)$ between two workers w_i and w_j is defined as the cosine similarity between their familiarity vector shown in Eqn. (2),*

$$S(w_i, w_j) = \frac{\mathbf{f}_i \cdot \mathbf{f}_j}{\|\mathbf{f}_i\| \cdot \|\mathbf{f}_i\|} = \frac{\sum_{k=1}^{n} f_{ki} \cdot f_{kj}}{\sqrt{\sum_{k=1}^{n} f_{ki}^2} \cdot \sqrt{\sum_{k=1}^{n} f_{kj}^2}}. \tag{2}$$

We want to recommend workers in W to new task categories $\mathbf{T}^N$. In order to do make an appropriate recommendation to these new categories with no prior knowledge, we need to get some initial information by selecting a small portion of workers and see how they perform on the new tasks' test packages, and the performance will become the guide on estimating other workers' quality. This small set of workers is called pioneer workers, and the detailed definition of them is shown in Definition 6.

Definition 6 *(Pioneer worker set). For each new task T_i^N, we select a subset of workers with size z ($z \ll m$) as pioneer workers*

$$P_i = \{w_{i1}^P, w_{i2}^P, \cdots, w_{iz}^P\} \subseteq W \tag{3}$$

Each worker in P_i will be sent to complete a test package of task T_i^N, and the result collection is $R_i = \{(w_{i1}^P, a_{i1}^P), (w_{i2}^P, a_{i2}^P), \cdots, (w_{iz}^P, a_{iz}^P)\}$.

Now we are going to define the willingess of the worker accepting the recommendations to execute tasks by the crowdsourcing platform. According to the long tail effect mentioned in [34], in a recommendation system, new users are likely to rate more items than experienced users. Similarly, in a crowdsourcing system, new workers with a short execution history will be more willing to accept recommended tasks, as they are not certain which task categories are suitable for them. Oppositely, experienced workers will be relatively resistant to the recommendations and choose tasks according their own preferences, as they have gained enough experience to judge whether the task categories are efficient to execute. Also it is common sense that the willingness will decrease slower as the experience rises, which is similar to the pattern of long-tail effect. Thus we define the recommendation acceptance willingness as shown in Definition 7.

Definition 7 *(Acceptance willingness). When a worker w_j is recommended with a task category by the crowdsourcing platform, the acceptance willingness $D(w_j)$ of w_j is defined as Eqn. (4),*

$$D(w_j) = \frac{1}{\sqrt{\log(|H_j| + 2)}}. \tag{4}$$

Now we will define the Accurate Recommendation Problem (ARP). Our objective is to give recommendations to the non-pioneer workers, and try to minimize the number of recommendations sent, while maximizing the number of workers receiving our recommendations at the same time. By doing this, we can make our task recommendations effective to save the potential waste on recommending tasks to bad workers or workers with low willingness to accept them. Also, we want to spread the recommendations across the worker set, in order not to excessively consume the workers' willingness by giving them too many recommendations at once, correspondingly retaining the workers to work on the platform in the future. Note that we need to meet the expected valid recommendation requirements given by tasks' requesters. Also, there is an upper bound for each worker accepting recommendations at a time, as we want to keep the precision and authority of our framework. Taking all the above into consideration, we define the Accurate Recommendation Problem (ARP) in Definition 8.

Definition 8 *(Accurate Recommendation Problem (ARP)). Give the crowdsourcing platform's history task category set $\mathbf{T} = \{T_1, T_2, \cdots, T_n\}$, and worker set $W = \{w_1, w_2, \cdots, w_m\}$. Give the new task category set as targets $\mathbf{T}^N = \{T_1^N, T_2^N, \cdots, T_l^N\}$, and for each new task category T_i^N, give its valid recommendation requirement G_i^N, pioneer worker set P_i, and the corresponding test result R_i. Give a recommendation count upper bound C. For each new task category T_i^N, we need to find a worker recommendation set $W_i^R \subseteq W - P_i$, maximizing the recommendation effectiveness, with each worker accepting no more than C, and each new task T_i^N receives at least G_i^N valid recommendations.*

With the help of estimated accuracy a_{ij}^N of non-pioneer workers w_j executing tasks in new category T_i^N, we can formulate ARP as an Integer Programming shown in (5)-(9). For easier computation and analysis, we integrate the two objectives into one. d_{ij} indicates the recommendation result, if $w_j \in W_i^R$, $d_{ij} = 1$, otherwise $d_{ij} = 0$. $\mathbb{1}(\cdot)$ is the indicator function, which returns 1 if the input statement is true, and returns 0 otherwise.

$$\max \sum_{j=1}^{m} \mathbb{1}\left(\sum_{i=1}^{l} d_{ij} > 0\right) / \sum_{i=1}^{l}\sum_{j=1}^{m} d_{ij} \tag{5}$$

$$\text{s.t.} \sum_{i=1}^{l} d_{ij} \le C, \forall w_j \in W \tag{6}$$

$$\sum_{j=1}^{m} a_{ij}^N D(w_j) d_{ij} \ge G_i^N, i = 1, 2, \cdots, l \tag{7}$$

$$d_{ij} = 0, \forall w_j \in P_i \tag{8}$$

$$d_{ij} \in \{0, 1\}, \forall i = 1, \cdots, n, j = 1, \cdots, m \tag{9}$$

Based on a polynomial-time reduction from the partition problem, we can prove that ARP is NP-Hard in Theorem 1.

Definition 9 *(The partition problem). Given a multiset denoted by $S = \{e_1, e_2, \ldots, e_m\}$ of positive integers, the partition problem is to decide whether S can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2.*

Theorem 1. *ARP is NP-Hard.*

Proof of Theorem 1. Given a partition problem instance, we can construct a polynomial-time reduction to a special instance of ARP. In this special instance of ARP, there are only two new task category and no pioneer worker set (i.e., $l = 2$ and $n = 2$). For the multiset $S = \{e_1, e_2, \ldots, e_m\}$ in the partition problem instance, we construct m workers and for worker w_j, let

$$a_{1j}^N D(w_j) = a_{2j}^N D(w_j) = e_j \tag{10}$$

In addition, for $i = 1, 2$, let G_i^N satisfy

$$G_i^N = \frac{1}{2}\sum_{j=1}^{m} e_j \tag{11}$$

Finally, for $w_j \in W$, let $C = 1$. Therefore, in this special instance, the objective (5) and the corresponding constraints (6)-(9) in the general problem are transformed into the following:

$$\max \ \sum_{j=1}^{m} \mathbb{1}\left(\sum_{i=1}^{2} d_{ij} > 0\right) / \sum_{i=1}^{2}\sum_{j=1}^{m} d_{ij} \tag{12}$$

$$\text{s.t.} \ \sum_{i=1}^{2} d_{ij} \le 1, \forall w_j \in W \tag{13}$$

$$\sum_{j=1}^{m} e_j d_{ij} \ge \frac{1}{2}\sum_{j=1}^{m} e_j, i = 1, 2 \tag{14}$$

$$d_{ij} \in \{0, 1\}, \forall i = 1, 2, j = 1, \cdots, m \tag{15}$$

Constraint (13) requires each worker to be assigned at most one task. If a worker is not assigned a task, it will violate the Constraint (14), which can be proved by summing up both sides of the Constraint (14). Since $G_1^N + G_2^N = \sum_{j=1}^{m} e_j$ and $C = 1$, all the workers must be assigned with one and only one task category such that Constraint (14) can be satisfied. Therefore, the objective value in this special instance is

$$\sum_{j=1}^{m} \mathbb{1}\left(\sum_{i=1}^{2} d_{ij} > 0\right) / \sum_{i=1}^{2}\sum_{j=1}^{m} d_{ij} = m/m = 1. \tag{16}$$

In addition, Constraint (13)-(15) is exactly a formulation for the partition problem. Therefore, if the constructed ARP instance has a feasible solution, we know that S can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2; Otherwise, S can not be partitioned into two such subsets S_1 and S_2. In other words, we can get the solution of the partition problem based on the solution of this ARP instance. Therefore, ARP is NP-Hard. □

Based on the above proof, we know that only finding a feasible solution for ARP is exactly NP-Hard. Therefore, there is also no polynomial-time approximation algorithm for ARP. Notice that the objective (5) in ARP is nonlinear. After some simple modification, we can transform this nonlinear objective into an equivalent linear one. This result and the detailed proof can be found in Theorem 2.

Theorem 2. *The nonlinear integer program (5)-(9) for ARP is equivalent to the following linear integer program:*

$$\min \ \sum_{i=1}^{l}\sum_{j=1}^{m} d_{ij} \tag{17}$$

$$s.t. \ \sum_{i=1}^{l} d_{ij} \ge 1, \forall w_j \in W \tag{18}$$

$$\sum_{i=1}^{l} d_{ij} \leq C, \forall w_j \in W \tag{19}$$

$$\sum_{j=1}^{m} a_{ij}^N D(w_j) d_{ij} \geq G_i^N, i = 1, 2, \cdots, l \tag{20}$$

$$d_{ij} = 0, \forall w_j \in P_i \tag{21}$$

$$d_{ij} \in \{0, 1\}, \forall i = 1, \cdots, n, j = 1, \cdots, m \tag{22}$$

Proof of Theorem 2. In order to show that the above two different integer programs are actually equivalent, we can prove that the optimal solution in these two-integer program can produce the same objective value for ARP, i.e.,

$$\sum_{j=1}^{m} \mathbb{1}\left(\sum_{i=1}^{l} d_{ij} > 0\right) / \sum_{i=1}^{l}\sum_{j=1}^{m} d_{ij}. \tag{23}$$

Denote by OPT_{non} the optimal solution of the nonlinear integer program (5)-(9) and OPT_{lin} the optimal solution for the linear integer program (17)-(22). Denote by OPT_{non}^{obj} and OPT_{lin}^{obj} the corresponding objective value for ARP produced by OPT_{non} and OPT_{lin}, respectively. Therefore, we can have

$$OPT_{non}^{obj} = OPT_{non} \text{ and } OPT_{lin}^{obj} = \frac{\sum_{j=1}^{m} \mathbb{1}\left(\sum_{i=1}^{2} d_{ij} > 0\right)}{OPT_{lin}} = \frac{m}{OPT_{lin}} \tag{24}$$

Notice that the feasible region in the new linear program (17)-(22) is a subset of that in the original nonlinear program (5)-(9) because of Constraint (17). Therefore, we have

$$OPT_{non}^{obj} \geq OPT_{lin}^{obj} \tag{25}$$

Now consider the original nonlinear program. If there is no worker in OPT_{non} that is not assigned with any task category, i.e.,

$$\sum_{i=1}^{l} d_{ij} \geq 1, \forall w_j \in W, \tag{26}$$

we can have

$$\sum_{j=1}^{m} \mathbb{1}\left(\sum_{i=1}^{2} d_{ij} > 0\right) = m \tag{27}$$

and

$$OPT_{non}^{obj} = OPT_{non} \leq \frac{m}{OPT_{lin}} = OPT_{lin}^{obj}. \tag{28}$$

Otherwise, there is at least one worker j^* in OPT_{non} that is not assigned with any task category, i.e.,

$$d_{ij^*} = 0, \forall i = 1, 2, \ldots, l. \tag{29}$$

Then we can assign worker j^* with any task category i^* ($w_{j^*} \notin P_{i^*}$). Notice that it does not violate any constraint (6)-(9). Therefore, we will get another solution OPT_{non}^*, where there is no worker that is not assigned with any task category and which satisfies

$$OPT_{non}^* = \frac{\sum_{j=1}^{m} \mathbb{1}\left(\sum_{i=1}^{2} d_{ij} > 0\right) + 1}{\sum_{i=1}^{2}\sum_{j=1}^{m} d_{ij} + 1} \geq \frac{\sum_{j=1}^{m} \mathbb{1}\left(\sum_{i=1}^{2} d_{ij} > 0\right)}{\sum_{i=1}^{2}\sum_{j=1}^{m} d_{ij}} = OPT_{non}, \tag{30}$$

where the above inequality holds because

$$\sum_{j=1}^{m} \mathbb{1}\left(\sum_{i=1}^{l} d_{ij} > 0\right) / \sum_{i=1}^{l}\sum_{j=1}^{m} d_{ij} \leq 1 \tag{31}$$

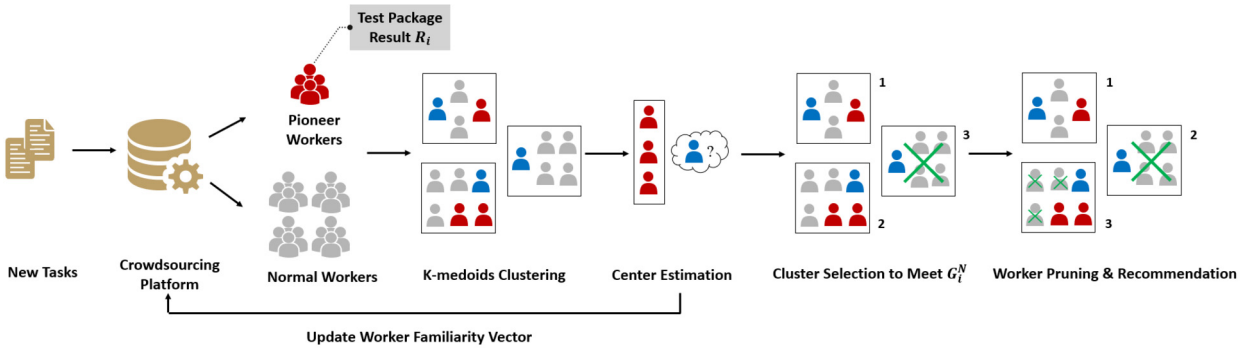


Fig. 3. PATRON framework workflow chart. We use k-medoids algorithm to cluster workers according to their familiarity, and estimate their quality by estimating the center worker of the cluster they belong to. Then we select the high quality clusters until we meet the valid recommendation requirements. In order to maximize the efficiency, we reversely prune some workers with low willingness to approximate the requirement.

Similarly, we can further get

$$OPT_{non}^{obj} = OPT_{non} \leq OPT_{non}^{*} \leq \frac{m}{OPT_{lin}} = OPT_{lin}^{obj}. \tag{32}$$

Combining Eq. (25) and Eq. (32), we have

$$OPT_{non}^{obj} = OPT_{lin}^{obj}. \tag{33}$$

Therefore, we finish the proof. □

4. PATRON: Pioneer-Assisted Task RecommendatiON

There are several critial challenges on designing a crowdsourcing task recommendation system under the realistic situation. First, comparing to previous works' assumptions, information is more limited for us to estimate the workers' quality and new tasks' features. Second, as more crowdsourcing tasks with complex contents are emerging, it becomes more difficult to analyze its content automatically, which means that we can only use the worker execution result to analyze the tasks' features. Because of the two reasons mentioned above, finding an efficient measurement method is challenging.

In this section, we will introduce our proposed Pioneer-Assisted Task RecommendatiON (PATRON) framework to recommend new tasks to workers on the crowdsourcing platform. The workflow is illustrated in Fig. 3. We design an iterative task-by-task recommendation and worker quality estimation strategy based on k-medoids [35], the clustering algorithm. When recommending workers to each new category, we estimate the woker quality of each cluster's center using the information collected from the pioneer workers, and use the estimation to represent the quality of the other workers in the corresponding cluster. Primal cluster selection and worker pruning approaches the maximization of recommendation efficiency.

To start with, we need to collect the initial knowledge of the task category to be recommended, we randomly select some workers, and send them invitations to execute the test packages of the new task T_i^N. As in realistic situations, the workers may not accept the invitations, so we need to supplement invitations, until we receive z pieces of results, which is collected in R_i. The workers participating in this execution are identified as pioneer workers P_i. This information will be an important reference when estimating the quality of normal workers.

Before seeking similarity between different workers, we abstract each worker w_j as a quality-willingness tuple $E_j = (D(w_j), \frac{\sum_{k=1}^{n} f_{kj}}{n})$. The former of the tuple is the acceptance willingness of w_j, while the latter is the average familiarity of w_j. We select the pioneer workers P_i as initial cluster centers, choose the Euclidean distance between the abstract tuples $dist(E_i, E_j)$ as the dissimilarities between workers w_i and w_j, and execute the k-medoids (CLARA [35] as implementation) to split the workers W into z clusters $X = \{X_1, X_2, \cdots, X_z\}$. As k-medoids algorithm always selects some data points that exist in the dataset as centers, we can select the cluster center $w_j^X \in X_j$ as the representation of quality and acceptance willingness of all workers in the corresponding cluster X_j. In order to avoid the situation that the clustering algorithm does not converge, we set up the round limit r to terminate the process.

Now we will estimate the normal worker's quality using the pioneer's test package execution result R_i. Note that although we only acquired the test package accuracy of each pioneer worker, it can also be regarded as the possibility of him/her producing the qualified answers, or to say, the quality of the him/her towards the specified task category. Also, according to the discussions about clusters above, we only need to estimate the quality of cluster centers, and use them to represent the quality of workers in their own cluster. To make the estimation more practical, we adopt the weighted average of the test accuracy of all pioneer workers P_i as the estimated quality a_{ij}^X of w_j^X, as shown in Eqn. (34).

Algorithm 1: PATRON Framework.

Input: $\mathbf{T}$, $\mathbf{W}$, $\mathbf{T}^N$, $\{G_i^N\}$, $\{P_i\}$, $\{R_i\}$, C
Output: Recommendation W_i^R for each T_i^N

for $i = 1$ **to** l **do**
  // Abstract Tuple Calculation
  for $j = 1$ **to** m **do** $E_j \leftarrow (D(w_j), \frac{\sum_{k=1}^{n} f_{kj}}{n})$;
  Run k-medoids on W, with P_i as initial centers, and Euclidean distance $dist(E_i, E_j)$ as dissimilarity between w_i, w_j, to get a result of clusters $X = \{X_1, \cdots, X_z\}$ with centers $w_j^X \in X_j$;
  // Center Quality Estimation
  for $j = 1$ **to** z **do**
    $a_{ij}^X = \frac{\sum_{k=1}^{z} S(w_j^X, w_{ik}^p) \cdot a_{ik}^p}{\sum_{k=1}^{z} S(w_j^X, w_{ik}^p)}$;
    foreach $w_k \in X_j$ **do** $a_{ik}^N \leftarrow a_{ij}^X$;
  // Primal Recommendation Set Construction
  $Valid \leftarrow 0$, $W_i^R \leftarrow \phi$, $j \leftarrow 1$;
  Rank clusters $\{X_j\}$ in descending order of a_{ij}^X;
  while $Valid < G_i^N$ **do**
    $W_i^R \leftarrow W_i^R \cup (X_j - P_i)$;
    $Valid \leftarrow Valid + a_{ij}^X D(w_j^X)|(X_j - P_i)|$;
    $j \leftarrow j + 1$;
  // Worker Pruning
  Rank $w_k \in X_j$ decreasingly by $D(w_j) \cdot \frac{C - A(w_j)}{C}$, result in $(w_1^j, w_2^j \cdots, w_{|X_j|}^j)$; $k \leftarrow |X_j|$;
  while $Valid > G_i^N$ **do**
    if $Valid - a_{ij}^X D(w_k^j) \geq G_i^N$ **then**
      if $w_k^j \notin P_i$ **then** $Valid \leftarrow Valid - a_{ij}^X D(w_k^j)$; $W_i^R \leftarrow W_i^R - \{w_k^j\}$;
      $k \leftarrow k - 1$;
    else Break;
  // Familiarity Vector Extension
  foreach $w_j \in W$ **do** $\mathbf{f}_j \leftarrow \langle \mathbf{f}_j; a_{ij}^N \rangle$;
Return $\{W_i^R\}$;

$$a_{ij}^X = \frac{\sum_{k=1}^{z} S(w_j^X, w_{ik}^p) \cdot a_{ik}^p}{\sum_{k=1}^{z} S(w_j^X, w_{ik}^p)} \tag{34}$$

After the quality estimation of normal workers, we select a primal set of recommendation targets to meet the valid recommendation requirement G_i^N of the new task category. We rank the estimated quality of cluster centers decendingly, and successively select the center w_j^X together with workers in his/her cluster as the primal targets while removing the pioneers in it. Each selected cluster produces $a_{ij}^X D(w_j^X)|X_j - P_i|$ expected valid recommendations. We keep on involving new clusters, until the total expected valid recommendation satisfies the requirement G_i^N given by the requester.

In order to maximize our recommendation efficiency, we need to discard the recommendations that would not be so efficient to reduce the recommendation excess from the primal solution. They might be low in quality with non-relevant experiences and return disqualified answers, or they have a low willingness on accepting platform recommendations. Therefore, for the last cluster we pick up X_j (with center estimated quality a_{ij}^X), we rank workers w_k in it in the descending sequence of modified willingness $D(w_k) \cdot \frac{C - A(w_k)}{C}$, where $A(w_k)$ is the recommendations already given to worker w_k in this process. The modified willingness reduces the possibility of recommending multiple task categories to the same worker, while trying to keep workers with high willingness to make the recommendation more likely to convert into a valid one. After the ranking, we prune the worker with the lowest modified will and subtract his/her generated expected valid recommendation, until discarding a worker makes G_i^N unsatisfiable.

After the worker pruning, we will recommend the new category T_i^N to all workers remaining in the primal set, and update the worker profile according to the clustering result. We label the normal worker's estimated quality as the same value of his/her cluster center's a_{ij}^N. Furthermore, in the recommendation of the following tasks, we extend the familiarity vector by one dimension using a_{ij}^N, and use the new vector to calculate the new worker similarity. We repeat the operations mentioned above, until all new categories are recommended with workers. We collect all the processes and describe it in the pseudo code in Algorithm 1.

Now we will analyze the performance of PATRON theoretically. We give the basic assumption that all clusters have a similar size, and $z \ll m$, $l < n$.

Theorem 3. *In each iteration of task recommendation, PATRON (Algorithm 1) has a time complexity of* $O(z(rm + zn) + \frac{m}{z} \log \frac{m}{z})$.

Proof of Theorem 3. Calculating the abstract tuples costs $O(\frac{mn}{l})$ (the first iteration $O(mn)$, the following iterations $O(m)$). k-medoids clustering costs $O((z(40 + z)^2 + z(m - z)) \cdot r)$ time [35] in maximum while executing for r rounds, and quality estimation costs $O(z^2 n)$ time, estimating z centers' quality by calculating z similarities each. In the primal set construction, the sorting costs $O(z \log z)$ time, and the construction costs $O(z)$ time. In the worker pruning, as all clusters have a similar size, we can regard them having the average size $\frac{m}{z}$. Therefore, the sorting costs $O(\frac{m}{z} \log \frac{m}{z})$ time, and the pruning costs $O(\frac{m}{z})$ time. According to the assumption $z \ll m$, the time complexity of primal set construction can be omitted, and the main complexity comes from the clustering process, center quality estimation and the worker pruning. Thus, collecting all the time cost together, we can get the time complexity of each iteration of PATRON is $O(z(rm + zn) + \frac{m}{z} \log \frac{m}{z})$. □

The traditional approach of task recommendation is: estimate every normal worker based on pioneer workers and the similarity relationship between them, then rank all workers by the expected valid recommendation value, and finally greedily select workers and recommend them to the target category, until the goal G_i^N is reached. It is easy to prove that the traditional approach has a time complexity of $O(mzn + m \log m)$ in each iteration of task recommendation. According to our assumptions, we have $mn > rm + zn$, thus PATRON costs less time comparing to the traditional approach, and we successfully achieve the goal of reducing the computation complexity.

5. Experiment

5.1. Experiment setup

We use the actual dataset to evaluate the PATRON framework. The dataset is collected from a famous Chinese crowdsourcing platform, Tencent SOHO. For the primary dataset, we gather the task execution history of each worker on the platform. Each history record contains worker ID, task category indicator, package type (test, inspector and normal), the number of total quests, the correctness of the checked quests and package status (terminated, abandoned, etc.). The raw dataset includes 108480 task execution records. The filtering procedure eliminates the test packages and inspector packages that have zero correctness and 100% correctness since they don't contribute to the recommendation framework. Also, to make the experiment more representative, we delete the task packages if the number of executions on the task package is less than 100 workers. Then we get 5476 available workers, 36749 valid records in 13 history task categories and 6670 valid records in 32 new task categories. Fig. 10 shows the package type distribution where the labels on the horizontal axis divide the history categories and new task categories. To demonstrate the superiority of our framework, we have selected the following algorithms as control groups.

- Random Recommendation Algorithm (RAND): Under the premise of not exceeding the maximum number of recommended people, we randomly select workers. They are then recommended to those task categories that have not yet met the requirements.
- Greedy Algorithm (GDY): First, workers are sorted according to their acceptance willingness of tasks in descending order. From the top to the bottom, workers are recommended to those task categories that have not met the requirements when they are available.
- Full Estimation Algorithm (FES): This is a traditional collaborative filtering user-based task recommendation method. First, the quality of all workers is estimated based on the quality of the pioneer workers. Then among the available workers, those who have the highest expected efficiency are continuously selected for recommendations until the task classification is reached.

We compare the efficiency of the PATRON framework simulating the actual situation. We measure the recommendation success rate and recommended worker quality in two conditions:

1. Varying the number of new requirements G when setting the maximum recommendation number $C = 3$ which is a general case on the Tencent SOHO platform.
2. Varying the maximum recommendation number C when setting the number of new requirements $G = 3$ which is also a common case on the platform.

Other parameters are set as: the percentage of training data set $P = 0.5$; the number of iterations $maxiter = 50$; the minimum tolerate change for one iteration of k-medoids $tol = 0.005$.

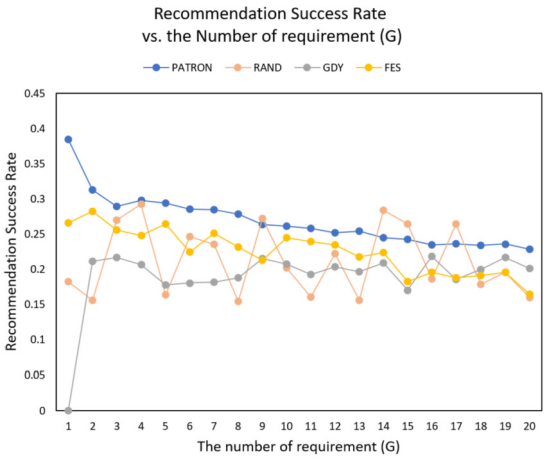


Fig. 4. Success Rate.

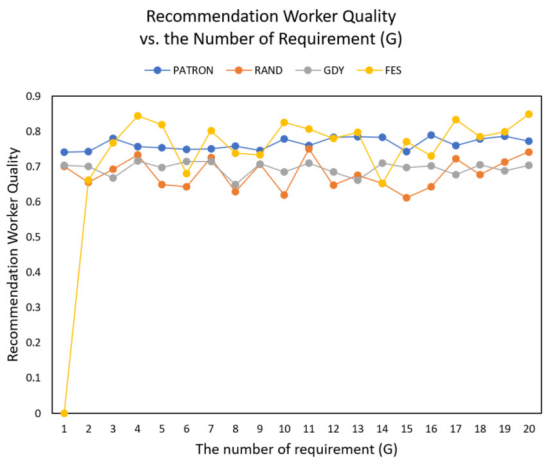


Fig. 5. Recommended Worker Quality.

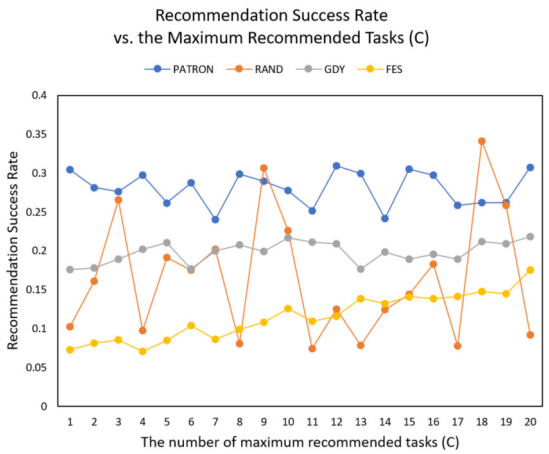


Fig. 6. Success Rate.

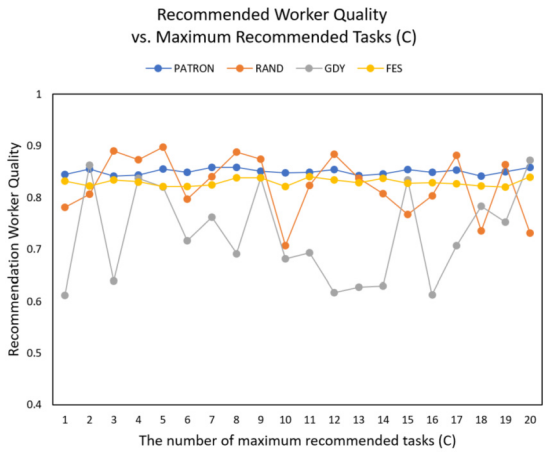


Fig. 7. Recommended Worker Quality.

5.2. Recommendation success rate

The recommendation success rate is calculated by the ratio between the workers who accept the recommended tasks and all recommended workers since the result implies the number of recommended workers who consequently accept a task form the recommended category on the platform. From Fig. 4 and Fig. 6, we can find that the PATRON performs generally better than other algorithms in both two conditions. When we change the maximum recommendation number C, the result of PATRON is more stable. When we change the number of new requirements, the PATRON is much better when the requirement G is small since the framework discards a few workers with low acceptance willingness who have a negative influence on recommendation efficiency. Besides, because workers tend to accept tasks in familiar categories, our recommendations are based on the experience and feature similarity of workers which indeed increase the success rate.

5.3. Recommended worker quality

The recommended worker quality is calculated by the ratio between the number of all passed quests and that of all examined quests among the new tasks, which implies the worker performance in the recommended new task categories. From Fig. 5 and Fig. 7, we can see that the workers' quality is more competitive and stable for the PATRON framework in both two conditions. The reason is that our estimation of recommended workers' quality is more sufficient with the quality of pioneer workers and center workers. The PATRON also recommends workers who are more familiar with the task categories based on the feature similarity and history records of workers. Besides, the PATRON protects the worker quality by discarding some workers with relatively low quality.

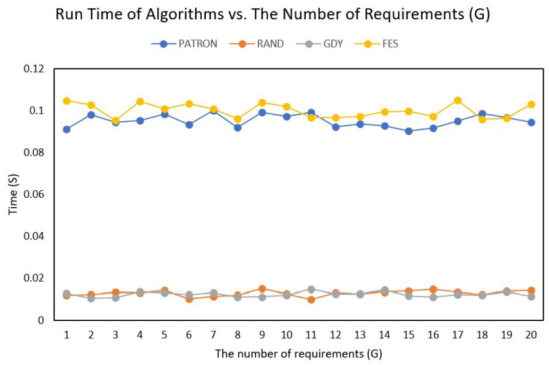


Fig. 8. Run Time vs. the Number of Requirements.

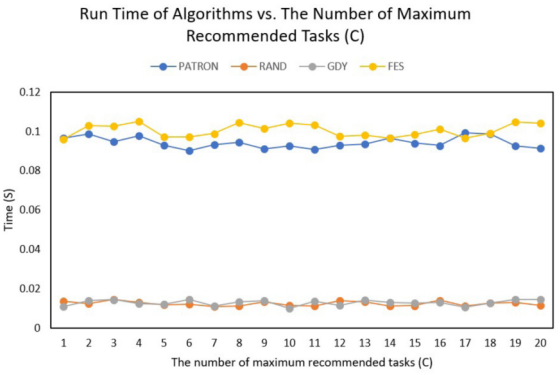


Fig. 9. Run Time vs. the Number of Maximum Recommended Tasks.

5.4. Run time of algorithms

We compare the run time of those algorithms. From Fig. 8 and Fig. 9, we can see that the run time of PATRON algorithm is faster than the FES algorithm in most cases. RAND and GDY algorithms achieve better performance in run time due to their simple search strategy. The run time of the PATRON algorithm will increase when the number of cluster heads increases, which becomes a trade-off between performance (accuracy) and speed. However, as we mentioned before, the proposed PATRON algorithm performs better on the recommendation success rate and the recommended worker quality, which are more important metrics in crowdsourcing systems in practice. Besides, we can deploy parallel computing or distributed computing on the server to improve the run time of PATRON.

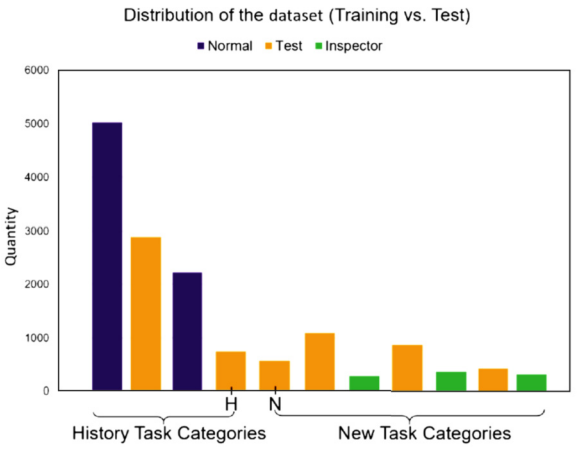


Fig. 10. Package Type Distribution.

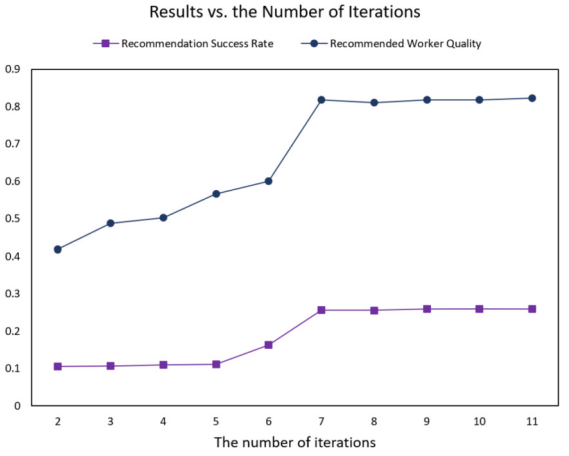


Fig. 11. Results vs. the Number of Iterations.

5.5. Algorithm parameters

Since the maximum recommended tasks C and the number of requirements G might change dependent on various realistic conditions, it is also necessary to find out how different internal parameters such as the number of iterations and the percentage of training data influence the results of the PATRON framework. First, we set the reasonable default settings according to the previous evaluations: the number of the maximum recommended tasks $C = 3$; the number of requirements $G = 3$; the percentage of training data set $P = 0.5$; the number of iterations $maxiter = 50$; the minimum tolerate change for one iteration of k-medoids $tol = 0.005$.

We conduct then evaluation by keeping the other parameters as the default and varying the investigated parameter. From Fig. 11, we can see that the results increase as the number of iterations increases. Owing to the relatively small scale dataset, the results quickly converge at the 8-th iteration and barely change after that. When it comes to a much larger dataset, we might need more iterations to get the clusters.

From Fig. 12, we can find that as the percentage of training dataset increases, the results also increase. Here notice that we have the range from 0.46 to 0.82. This is because the pioneer workers are not enough for the PATRON framework when the percentage of training dataset P is small. And the ARP problem has no solution when the percentage of training dataset

Fig. 12. Results vs. the Percentage of Training Data.

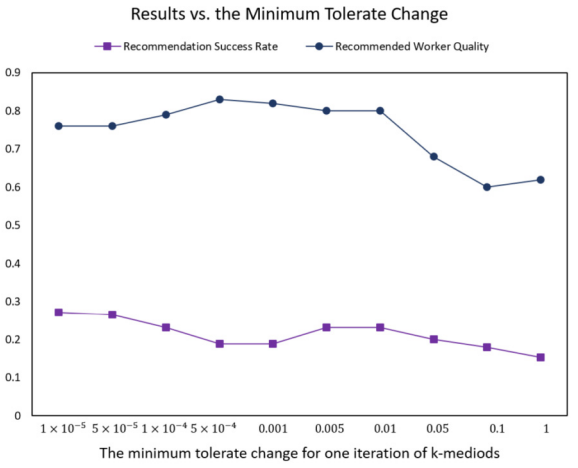


Fig. 13. Results vs. the Minimum Tolerate Change.

P is too large and less dataset is used to process the recommendation. In the actual conditions, the training dataset is supposed to be as smaller as possible, but our evaluation uses 0.5 of the dataset for training which seems not effective. Again, this is due to the limited evaluation data size. We believe that the framework requires less percentage of the training dataset in realistic conditions where a very large dataset is provided. From Fig. 13, we can see that the results fluctuate when the minimum tolerance for one iteration changes. However, the overall trend is that when the minimum tolerance gets larger, the results are worse.

6. Conclusion

Traditional crowdsourcing task recommendation systems are limited by the realistic condition where the tasks have a short lifespan, no duplicate execution and non-text tasks. In this paper, we formally define the Accurate Recommendation Problem (ARP) problem and prove it is NP-Hard. To solve the problem, we propose a unified Pioneer-Assisted Task RecommendatiON (PATRON) framework. The framework gets the initial knowledge of the new tasks from the selected set of pioneer workers. Accurate and efficient recommendations are offered with cluster selection and worker pruning. The experiments are based on real datasets from the Tencent SOHO crowdsourcing platform. The experiment shows the efficiency of our proposed framework on recommendation success rate and recommended worker quality and the relationship between parameters and the recommended results. A future direction is to apply the proposed framework to other crowdsourcing platforms, so as to solve the existing problems of these platforms and ultimately improve the performance of task recommendations.

Declaration of competing interest

The authors certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

[1] C. Zhang, Y. Wang, P. Liu, T. Lin, L. Luo, Z. Yu, X. Zhuo, Pmviewer: a crowdsourcing approach to fine-grained urban PM2.5 monitoring in China, in: IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS, 2017, pp. 323–327.
[2] H. Hu, G. Li, Z. Bao, Y. Cui, J. Feng, Crowdsourcing-based real-time urban traffic speed estimation: from trends to speeds, in: IEEE International Conference on Data Engineering (ICDE), 2016, pp. 883–894.
[3] M. Dittus, G. Quattrone, L. Capra, Mass participation during emergency response: event-centric crowdsourcing in humanitarian mapping, in: Conference on Computer Supported Cooperative Work (CSCW), 2017, pp. 1290–1303.
[4] X. Chen, B. Deng, Task allocation schemes for crowdsourcing in opportunistic mobile social networks, in: International Conference on Computing, Networking and Communications (ICNC), 2018, pp. 615–619.
[5] L. Qiao, F. Tang, J. Liu, Feedback based high-quality task assignment in collaborative crowdsourcing, in: IEEE International Conference on Advanced Information Networking and Applications (AINA), 2018, pp. 1139–1146.
[6] J. Fan, X. Zhou, X. Gao, G. Chen, Crowdsourcing task scheduling in mobile social networks, in: International Conference on Service Oriented Computing (ICSOC), 2018, pp. 317–331.
[7] C. Liu, X. Gao, F. Wu, G. Chen, QITA: quality inference based task assignment in mobile crowdsensing, in: International Conference on Service Oriented Computing (ICSOC), 2018, pp. 363–370.

[8] Allanyu, Tencent SOHO, https://soho.qq.com/tasks, 2020.
[9] R. Mizuhara, K. Sakai, S. Fukumoto, A collaborative-task assignment algorithm for mobile crowdsourcing in opportunistic networks, in: IEEE International Conference on Communications (ICC), 2018, pp. 1–6.
[10] K. Kumai, M. Matsubara, Y. Shiraishi, D. Wakatsuki, J. Zhang, T. Shionome, H. Kitagawa, A. Morishima, Skill-and-stress-aware assignment of crowdworker groups to task streams, in: AAAI Conference on Human Computation and Crowdsourcing (HCOMP), 2018, pp. 88–97.
[11] M.S. Safran, D. Che, Efficient learning-based recommendation algorithms for top-N tasks and top-N workers in large-scale crowdsourcing systems, ACM Trans. Inf. Syst. (TOIS) 37 (2019) 2.
[12] X. Lin, J. Xu, H. Hu, Z. Fan, Reducing uncertainty of probabilistic top-k ranking via pairwise crowdsourcing, in: International Council for Open and Distance Education (ICDE), vol. 14, 2018, pp. 1757–1758.
[13] S. Castano, A. Ferrara, S. Montanelli, Introducing online profile learning in crowdsourcing task routing, in: Italian Symposium on Database Systems (SEBD), 2018.
[14] M. Safran, D. Che, Efficient learning-based recommendation algorithms for top-n tasks and top-n workers in large-scale crowdsourcing systems, ACM Trans. Inf. Syst. (TOIS) 37 (2019) 2.
[15] Q. Kang, W.P. Tay, Task recommendation in crowdsourcing based on learning preferences and reliabilities, CoRR, arXiv:1807.10444 [abs], 2018.
[16] D.B. Noureddine, A. Gharbi, S.B. Ahmed, A social multi-agent cooperation system based on planning and distributed task allocation - real case study, in: International Conference on Software Technologies (ICSOFT), 2018, pp. 483–493.
[17] Y. Du, Y.-E. Sun, H. Huang, L. Huang, H. Xu, X. Wu, Quality-aware online task assignment mechanisms using latent topic model, Theor. Comput. Sci. (TCS) 803 (2019) 130–143.
[18] G. D'Angelo, L. Severini, Y. Velaj, Recommending links through influence maximization, Theor. Comput. Sci. (TCS) 764 (2019) 30–41.
[19] B.I. Aydin, Y.S. Yilmaz, Y. Li, Q. Li, J. Gao, M. Demirbas, Crowdsourcing for multiple-choice question answering, in: AAAI Conference on Artificial Intelligence (AAAI), 2014, pp. 2946–2953.
[20] Y. Zheng, G. Li, R. Cheng, DOCS: domain-aware crowdsourcing system, Proc. VLDB Endow. (PVLDB) 10 (2016) 361–372.
[21] P. Mavridis, D. Gross-Amblard, Z. Miklós, Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing, in: International Conference on World Wide Web (WWW), 2016, pp. 843–853.
[22] M. Saberi, O.K. Hussain, E. Chang, Quality management of workers in an in-house crowdsourcing-based framework for deduplication of organizations' databases, IEEE Access 7 (2019) 90715–90730.
[23] J. Zhang, V. Sheng, J. Wu, Crowdsourced label aggregation using bilayer collaborative clustering, IEEE Trans. Neural Netw. Learn. Syst. 30 (2019) 3172–3185.
[24] V.K. Singh, S. Mukhopadhyay, F. Xhafa, A budget feasible peer graded mechanism for iot-based crowdsourcing, CoRR, arXiv:1809.09315 [abs], 2018.
[25] A.D. Sarma, A.G. Parameswaran, J. Widom, Towards globally optimal crowdsourcing quality management: the uniform worker setting, in: International Conference on Management of Data (SIGMOD), 2016, pp. 47–62.
[26] C. Kavitha, R.S. Lakshmi, J.A. Devi, U. Pradheeba, Evaluation of worker quality in crowdsourcing system on hadoop platform, Int. J. Reason.-Based Intell. Syst. (IJRIS) 11 (2019).
[27] S. Sarker, M.A. Razzaque, M.M. Hassan, A. Almogren, G. Fortino, M. Zhou, Optimal selection of crowdsourcing workers balancing their utilities and platform profit, IEEE Int. Things J. (IoT-J) 6 (2019) 8602–8614.
[28] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, J. Han, Faitcrowd: fine grained truth discovery for crowdsourced data aggregation, in: ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2015, pp. 745–754.
[29] Z. Yang, Z. Zhang, Y. Bao, X. Gan, X. Tian, X. Wang, Ontac: online task assignment for crowdsourcing, in: IEEE International Conference on Communications (ICC), 2016, pp. 1–6.
[30] J. Li, Y. Zhu, Y. Hua, J. Yu, Crowdsourcing sensing to smartphones: a randomized auction approach, IEEE Trans. Mob. Comput. (TMC) 16 (2017) 2764–2777.
[31] D. Yang, G. Xue, X. Fang, J. Tang, Incentive mechanisms for crowdsensing: crowdsourcing with smartphones, IEEE/ACM Trans. Netw. (TON) 24 (2016) 1732–1744.
[32] H. Zhang, M. Sugiyama, Task selection for bandit-based task assignment in heterogeneous crowdsourcing, in: Conference on Technologies and Applications of Artificial Intelligence (TAAI), 2015, pp. 164–171.
[33] J. Pilourdault, S. Amer-Yahia, D. Lee, S.B. Roy, Motivation-aware task assignment in crowdsourcing, in: International Conference on Extending Database Technology (EDBT), 2017, pp. 246–257.
[34] C.C. Aggarwal, Recommender Systems - the Textbook, Springer, 2016.
[35] R.T. Ng, J. Han, CLARANS: a method for clustering objects for spatial data mining, IEEE Trans. Knowl. Data Min. (TKDE) 14 (2002) 1003–1016.