J Inf Process Syst, Vol.14, No.1, pp.129~139, February 2018
https://doi.org/10.3745/JIPS.04.0055

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Crowdsourcing Software Development: Task Assignment Using PDDL Artificial Intelligence Planning

Muhammad Zahid Tunio*, Haiyong Luo**, Cong Wang*, Fang Zhao*,
Wenhua Shao*, and Zulfiqar Hussain Pathan***

Abstract

The crowdsourcing software development (CSD) is growing rapidly in the open call format in a competitive environment. In CSD, tasks are posted on a web-based CSD platform for CSD workers to compete for the task and win rewards. Task searching and assigning are very important aspects of the CSD environment because tasks posted on different platforms are in hundreds. To search and evaluate a thousand submissions on the platform are very difficult and time-consuming process for both the developer and platform. However, there are many other problems that are affecting CSD quality and reliability of CSD workers to assign the task which include the required knowledge, large participation, time complexity and incentive motivations. In order to attract the right person for the right task, the execution of action plans will help the CSD platform as well the CSD worker for the best matching with their tasks. This study formalized the task assignment method by utilizing different situations in a CSD competition-based environment in artificial intelligence (AI) planning. The results from this study suggested that assigning the task has many challenges whenever there are undefined conditions, especially in a competitive environment. Our main focus is to evaluate the AI automated planning to provide the best possible solution to matching the CSD worker with their personality type.

Keywords

AI, Crowdsourcing, Personality, Planning Language, Software Development, Task

1. Introduction

Crowdsourcing software development (CSD) uses an open call format online to catch a large number of workers to participate. The open call format of CSD involves three types of roles: the requester (i.e., for the one the project is undertaken), the platform (i.e., the service provider), and the CS developer (i.e., the person for coding and testing). This kind of call format always collects large numbers of self-selected tasks. In this process, several numbers of developers can register and select the task at the platform. The platform is also responsible to evaluate the submitted tasks to decide for the best solution, from the developers, to pay the rewards. The authors [1,2] stated that the selection of an effective and appropriate task from the extensive large set is a hectic work for CS developers. It is also a tiring and

Manuscript received April 7, 2017; first revision August 29, 2017; accepted September 1, 2017.
Corresponding Author: Muhammad Zahid Tunio (Zahid.tunio@bupt.edu.cn)
* School of Software Engineering, Beijing University of Posts and Telecommunication, Beijing, China ({wangc, zfsse}@bupt.edu.cn, shaowenhua@ict.ac.cn)
** Institute of Computer Technology, Chinese Academy of Science, Beijing; and Beijing Key Laboratory of Mobile Computing and Pervasive Devices, Beijing, China (yhluo@ict.ac.cn)
*** School of Economics and Management, Beijing University of Posts and Telecommunications, Beijing, China (zulfi2k3bcs@yahoo.com)

time-consuming job for platform workers to evaluate thousands of submitted tasks from developers. For example, [3] also maintained that finding an effective task from the submitted tasks is a hard and time-consuming job for the platform workers. Based on the studies by [4,5], receiving or assigning an improper task to improper CS developer may not only decrease the quality of the software deliverables but also causes overburden on both platform and developers. Mostly, workers view a fewer numbers of recent tasks are posted on the CS platform because hundereds of tasks are posted every day [6]. By considering the skills and expertise level of the CS developers, unrealistic matching of CS developers and tasks would affect the software quality. Synchronization between expertise or knowledge of developers and tasks is a serious problem which can benefit both (platform and CS developers) [7]. Based on the findings [8], CSD does not only face technical issues but also deals with human-related issues at the same time. It is an admitted statement that software is made for the people by the people [9]. The personality of CS developers is an important element of people factor which directly impacts the results or outcomes of task development in CSD [10]. Currently, the artificial intelligence (AI) methods being applied in a conventional system are growing for the solution of problems. For examples, these days, planning methods including robotics and space machine are widely used in some particular areas. Nowadays, researchers have also started to use these techniques in other areas [11,12]. This study is going to use the classical planning of AI techniques to assist the CSD platform and CS developers to assign and choose the task logically with their personality type match. By using this technique, this study not only provided the best solution to reduce the overburden of searching and matching the task but also increased the efficiency which helped in providing quality deliverables.

This paper comprises five sections. Section 1 describes the introduction of the study. In Section 2 related work and background information are discussed. In Section 3 the methodology and study setting are presented. Section 4 presents the analysis of the results and their discussion. The conclusion is given in Section 5.

2. Background and Related Work

Crowdsourcing is a distributed outsourcing to an indeterminate, usually outsized crowd of people in an open call format. It has attracted a great attention from the industry and academia. When employing crowdsourcing to accomplish software development tasks, CSD faces challenges of assigning, sorting, and searching the suitable developers for the specific tasks. Until today, most developed tasks are assigned in the form of bidding or competition. As a result, a large amount of human effort and time are wasted in the searching for the suitable task with personal preference. However, many CSD developers cannot compete for the tasks [5]. To recommend a CSD task to a developer [2] developed a content-based technique, which takes a record from the history of the developer i.e. winning and registration to match automatically the task and the developer. The authors [13] described and proposed bias correction in crowd data in the form of modeling. They used a gold standard data set to estimate the CSD workers model accuracy. However, this method was used in micro-tasking. [14] used an implicit modeling based on skills and interest of CSD worker to recommend the classification-based task. The authors [15] mentioned and proposed an approach based on task matching which encourages and motivates CSD workers to do a task for continuous and long run. This approach focused on the recommendation features of the tasks to be best matched with the workers.

The authors of [12] presented an architecture for general crowdsourced systems with an automated planner role. In this approach, two key differences were found. Firstly, random versus guided behavior was designed for assigning the CS workers. Secondly, the performance results of open CS were incomplete and poor. Instead, coordination algorithms which are currently used in guided crowdsourcing are not suitable for CSD for more useful solutions. The work in [11], suggested a guided crowdsourcing which tested the incorporate crowdsourcing. The algorithm was used to schedule the resources and dynamically assign micro-task to crowdsource workers. In that way, AI methods can evaluate to coordination with users for achieving the specific collective goal performance in a CSD environment. [8] presented a Planning Domain Definition Language (PDDL) approach to generalize CS environment. However, in this paper, we introduced AI planning by using PDDL in CSD especially in Competitive Scenario.

2.1 Use of Personality in Crowdsourcing Software Developing

In order to avoid the risks of giving the task to improper personality types of CSD, the authors [16] proposed a suitable model. According to that model, it was suggested that, a task assigned to a developer must be based on suitable personality types. For instance, the personality of a programmer should be Introvert (I), the personality of a system analyst should be Extrovert (E), and the tester has Sensing (S) and Thinking (T) personalities. The software designer should be with Intuitive (N) and Think (T) personalities. This is because the personality types of the developer are the important human aspects to ensure the quality of software tasks. It is also confirmed that a technically sound individual cannot perform satisfactorily unless he/she is assigned with development tasks based on the person's personality types. [17] mentioned that assigning a specific personality to the task in software development is well suited for their traits to increase the successful outcome of the tasks. In order to help participants with the detection of appropriate tasks according to their individual preferences, this study suggested new techniques to facilitate the required self-identification process. The classification of individual personality types was classified on the Myer Briggs Type Indicator (MBTI) test, which follows the combination of four-dimensional pairs. From the four dimensional pairs, there were 16 personality combinations as shown in Table 1. To evaluate the personality of a CSD worker, this study will use the MBTI personality type as the instrument. This instrument is widely used in software engineering the research [18-21].

Table 1. MBTI personality numbers and types

Personality #	Personality type	Personality #	Personality type
1	ISTJ	9	ESTP
2	ISFJ	10	ESFP
3	INFJ	11	ENFP
4	INTJ	12	ENTP
5	ISTP	13	ESTJ
6	ISFP	14	ESFJ
7	INFP	15	ENFJ
8	INTP	16	ENTJ

3. Research Methodology

This study focused on searching for the possible results of the following key research questions by simulating the CSD developers and task:

(Research Question 1) Is it possible to generate plans to attract and assign the task to CSD developer who matches with the task and personality relationship criteria for the CSD tasks?

(Research Question 2) Is it possible to generate plans to lessen the overburden of platform and CSD developer for searching the open tasks and optimize the assigning and participation of CSD developer?

3.1 Implementation of the Planning

In order to get the answers of the above Research Questions, this study implemented the AI algorithm known as the automated planning to describe the basic mechanism and evaluation of CSD task assignments problem. This study used the PDDL, a planning language for AI planning systems. In PDDL first model, a domain explanation was defined, and then the problem which signified the situations will be defined. In CSD task assignment or sorting problem case, the domain description should have the defined objects, predicates, and actions which must relate with the tasks and CSD developer, as described in Tables 2–4. It is also very important that all criteria constraints and features are applicable to the execution of a well-defined plan.

Table 2. Task predicates

Task_open call	Open Call for tasks on CSD platform for competition
Task has_registration	CSD developer has to register for competition.
Task has_submission	Task submitted by CSD developer for evaluation to CSD platform
Task has_duration	Task has the time duration to submit within the time
Pays_rewards	If there is winner of the competition, CSD developer has to be paid a reward.

Table 3. CSD developer and personality predicates

has_registration	CSD developer has to be registered for competition.
has_personality type	CSD developer has to submit a personality type (by MBTI test).
has_submission	CSD developer must submit his submission for competition.
Winning reward	A winner of the competition gets a reward.

Table 4. Plan predicates

Task matching	Possible to be matched with suitable personality.
Personality	Should make task possible to be matched according to the personality types.
Task_assingned	Making possible that a right personality to have the right suitable task.
Winning_reward	A winner of the competition gets a reward.

3.2 Explanation of Predicates and Object Domain

When requester wants to resolve a task posted on the platform by a CSD developer through an open call format, the developer must possess the requirements to resolve the task and in the same way, the task poster needs to provide the basic for the solution. The platform is an agent, the initial state is the task on the platform, and the goal is to assign a task to requester that match the specific requirements. For improving the tasks assigning plan, a number of the premeditated elements are separated to the problem of crowdsourcing software development. The basic elements in competitive environment are uttered in Table 5.

Table 5. Basic elements in CSD competitive environment

Basic element	Description
Task (T)	Task posted on CSD platform
CSD developer (D)	The CSD developer who has to register for competition
Personality (P)	Personality type of CSD developer (through the MBTI test)
Submission of task (S)	Submission of task on platform
Evaluation (E)	Evaluation of task by the platform personals
Reward (R)	If there is a winner of the competition, the reward will be announced.

3.3 Domain Actions Definition

To generate an intelligent plan for task matching, some actions needed to be defined in the algorithm.

- Match: Try to match a CSD developer to whom the task will be given by considering personality match.
- Assign: To assign a task to a matched personality if the task is posted according to his/her personality type.
- Submission: Mark the task as submitted when submitted from the CSD developer for competition
- Review: After submitting the task reviewed and evaluated for reward.
- Reward: If the submitted task wins the competition, then pays the reward.

3.4 Definition of the Problem

The context established for the task requirement in the matching situation presented by this study for a personality and a CSD task in terms of time and monetary benefits, are depicted in Fig. 1. As discussed in the problem section, it is possible to analyze and match a task for a CSD platform based on its firmed description. The CSD platform has the ability to match the CSD developers that are the best matched for the developing and solving of the task.

It is very essential to the relationship between the objects' of personality and task with the conditioned personality, time, and monetary reward. This relationship sets a correlation between, incentive which a CSD developer attracts to participate (Get_reward ?p ?r), the personality type of the CSD developer that a CSD has (has_personlity ?P ?T), if required, time constraints to compete the task (Task_ time ?t ?T)and submission of task (submission_for ?S ?T). To implement the plan and discover the arrangements and different situations between personalities and task (objects) problem

files were formed. In order to incorporate all necessary set of objects for suggested combination, the initial state is where a set (T) is started. Tasks to be assigned, matched to a developer who required the set of the Personality (P). It is also mentioned that if the CSD task has an additional requirement such as special skills, time constraints, and incentive attraction, they have to motivate. To allocate a huge number of CSD developers to assign tasks on the CSD platform from the registered personalities who match the criteria is the goal and final state of the planner.

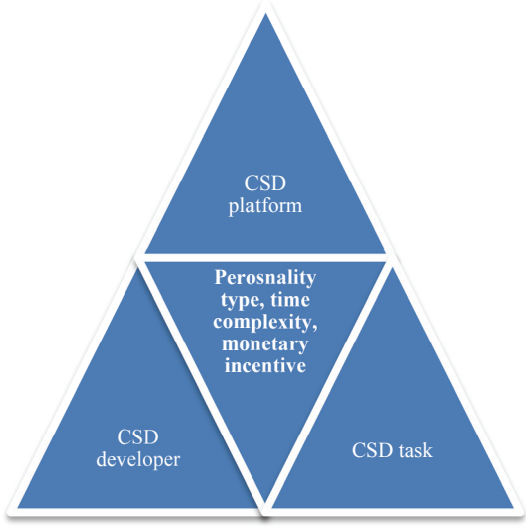

Fig. 1. Mapping of the CSD elements.

3.5 Generation of AI Plan

This section describes three situations where task and personality type are connected with predicates in a different context. The changing of two variables, i.e., reward (R) and time complexity (T) is the main goal. To represent some firm and normal situations in the domain of task matching, different results based on the merits and demerits of the classical planning approach are generated by each configuration. Table 6 represents the symbols used in predicates.

Table 6. Symbols representing the predicates

Symbol	Representing the predicates
P	Personality type
T	Task
t	Time
r	Reward
S	Submission
R	Registration
s	Skills (if required)

Situation 1: This situation was planned to illustrate an uncomplicated successful way, where each task was posted on the CSD platform with time constraints, the starting and ending date, monetary incentive, and registration with MBTI Personality test. At least one matching personality with the

criteria was available. In the CSD context, this situation was simply matched by a relationship where enough CSD developers were available in the competition to match all tasks conditions; with no adjustment, was required for the variables into the system.

P1 has R and has t to S1, S2 the T1, T3 for competition of r

P2 has R and has t to S3, S1 the T2, T1 for competition of r

P3 has R and has t to S4, S5 the T4, T5 for competition r

PN has R and t to Sn the Tn for competition of r

A CSD developer with matching variables for each task was preferred for the generated plan.

Situation 2: In this situation, no matching personality type was available, for a single task within the registered crowd. It was also mentioned that special skills required to match the task and reward were also put high, but there is no CSD developer willing to register itself for the reason of time complexity.

P1 has R and is available for t, take T1, and S for r

P5 has R, and available for t, take T1 and S for r

Pn not R, and not available for t, do not take T1 and S for r

For this situation, there was no plan generated as a result. As the goal was to assign the tasks, it was impossible find the suitable matching for the task due to time complexity factor. When analyzing the requester for the set of tasks where there was no matching in the CSD crowd the algorithm did not generate any plan. In this situation, in order to achieve the goal successfully, this study suggested that the variables should be adjusted according to the situation to better match the task and variables. Moreover, the complexity was relaxed for (Tn).

Situation 3: The aim of this situation was, to adjust the time complexity when the task did not match in the previous situation. The time of submission of task reset in order to attract more CSD developers for competition. In this situation, the plan was generated as a result. The features of this algorithm allowed changing variables according to the different situations, for instance, to increase the monetary reward and reduce time complexity. Someone can also increase the time complexity factor and reduce the rewards incentive. The plans will be generated if there is flexibility in variables and at least one match is available, which matches the task and personality relationship.

4. Results and Discussion

(Research Question 1) Is it possible to generate plans to attract and assign the task to CSD developer who matches with the task and personality relationship criteria for the CSD tasks?

When the requirement for each task is matched with at least one CSD developer the algorithm generates plans. It was also mentioned that whenever there is no requirement matched with the task, the algorithm did not generate the plan in that situation. It is quite obvious that AI planning technique is always searching for a CSD worker who meets all the requirement or at least one required conditions for matching with the task and personality relationship to ensure the deliverable quality for submission.

(Research Question 2) Is it possible to generate the plans to lessen the burden of platform and CSD developer for searching the open tasks and optimizing the assigning, and participation of CSD developer?

Within the competitive environment, considering only the classical planning approach, the results for

this algorithm showed that to generate plans, can be impossible when you optimize any variable. This occurs regardless of whether their feasibility is suitable or not. This approach indicated that the algorithm is flexible enough to test many configurations, in different contexts in order to best match with task and personality types.

In this context, a new research question has been determined throughout the research. Can the resultant plans help to optimize the progress in the quality of the solution required by the requester? The answer to this research question is that by the setting of situations and different variables this algorithm is generating better plans, as discussed in Situation 3 above. The requirements of variables should be flexible in order to achieve the optimized results. For instance, to increase time complexity, short duration, reward motivation, and relaxing, special skills are required. Assigning the right task to the right CSD developer according to his/her personality types and getting benefits and exploring more of AI Planning techniques in crowdsourcing software development is at the initial state.

5. Conclusion

As discussed above, the main objective of this research is to evaluate a PDDL planning for providing a relationship to match a CSD developer with personality and task. Moreover, it also provides a quality submission and reduces the burden from the platform as well as the CSD developer to match the task and CSD developer. From the results, it was very obvious that the automated planning has a significant impact, as the planner has to pre-categorize the tasks and personality types. It was also noted that planning algorithm has the capacity of allowing changes in the variables and it provided more and proper match.

The limitation of this planning algorithm is that in inflexible conditions for the matching of CSD task and developer, the plans will not be generated. Therefore, the variables must be adjusted according to the situations to achieve the best results.

It is suggested that CSD platforms may adopt the automated planning by using PDDL AI techniques, in order to reduce a large amount of human effort wasted in searching and assigning the tasks. In future, we will implement this algorithm in different situations and on real crowdsourcing platforms to get more validity.

Acknowledgement

This work is supported in part by the National Key Research and Development Program (No. 2016-YFB-0502004), the National Natural Science Foundation of China (No. 61374214), the and the Open Project of Beijing Key Laboratory of Mobile Computing and Pervasive Devices.

References

[1] K. Mao, Y. Yang, Q. Wang, Y. Jia, and M. Harman, "Developer recommendation for crowdsourced software development tasks," in *Proceeedings of the 9th IEEE Symposium on Service-Oridented System Engineering*, San Francisco Bay, CA, 2015, pp. 347-356.

[2] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *Journal of Systems and Software*, vol. 126, pp. 57-84, 2017.

[3] Y. Fu, H. Chen, and F. Song, "STWM: a solution to self-adaptive task-worker matching in software crowdsourcing," in *Algorithms and Architectures for Parallel Processing*. Cham: Springer International Publishing, 2015, pp. 383-398.

[4] L. B. Chilton, J. J. Horton, R. C. Miller, and S. Azenkot, "Task search in a human computation market," in *Proceedings of the ACM SIGKDD Workshop on Human Computation*, Washington, DC, 2010, pp. 1-9.

[5] E. Aldhahri, V. Shandilya, and S. Shiva, "Towards an effective crowdsourcing recommendation system: a survey of the state-of-the-art," in *Proceedings of IEEE Symposium on Service-Oriented System Engineering*, San Francisco Bay, CA, 2015, pp. 372-377.

[6] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay, "Cascade: crowdsourcing taxonomy creation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Paris, France, 2013, pp. 1999-2008.

[7] T. D. LaToza and A. Van Der Hoek, "A vision of crowd development," in *Proceedings of the 37th IEEE International Conference on Software Engineering*, Florence, Italy, 2015, pp. 563-566.

[8] L. Machado, R. Prikladnicki, F. Meneguzzi, C. R. de Souza, and E. Carmel, "Task allocation for crowdsourcing using AI planning," in *Proceedings of the 3rd International Workshop on Crowdsourcing in Software Engineering*, Austin, TX, 2016, pp. 36-40.

[9] A. R. Gilal, J. Jaafar, M. Omar, S. Basri, and A. Waqas, "A rule-based model for software development team composition: team leader role with personality types and gender classification," *Information and Software Technology*, vol. 74, pp. 105-113, 2016.

[10] P. K. Murukannaiah, N. Ajmeri, and M. P. Singh, "Acquiring creative requirements from the crowd: understanding the influences of personality and creative potential in crowd RE," in *Proceedings of IEEE 24th International Requirements Engineering Conference (RE)*, Beijing, China, 2016, pp. 176-185.

[11] I. Lykourentzou, D. J. Vergados, K. Papadaki, and Y. Naudet, "Guided crowdsourcing for collective work coordination in corporate environments," in *Computational Collective Intelligence: Technologies and Applications*. Heidelberg: Springer, 2013, pp. 90-99.

[12] K. Talamadupula, S. Kambhampati, Y. Hu, T. A. Nguyen, and H. H. Zhuo, "Herding the crowd: automated planning for crowdsourced planning," in *Proceedings of the 1st AAAI Conference on Human Computation and Crowdsourcing*, Palm Springs, CA, 2013.

[13] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Honolulu, HI, 2008, pp. 254-263.

[14] V. Ambati, S. Vogel, and J. G. Carbonell, "Towards task recommendation in micro-task markets," in *Human Computation: Papers from the 2011 AAAI Workshop*. Menlo Park, CA: AAAI Press, 2011, pp. 80-83.

[15] M. C. Yuen, I. King, and K. S. Leung, "Task matching in crowdsourcing," in *Proceedings of 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing (iThings/CPSCom)*, Dalian, China, 2011, pp. 409-412.

[16] L. F. Capretz and F. Ahmed, "Making sense of software development and personality types," *IT Professional*, vol. 12, no. 1, pp. 6-13, 2010.

[17] L. F. Capretz, D. Varona, and A. Raza, "Influence of personality types in software tasks choices," *Computers in Human Behavior*, vol. 52, pp. 373-378, 2015.

[18] D. Geiger and M. Schader, "Personalized task recommendation in crowdsourcing information systems: current state of the art," *Decision Support Systems*, vol. 65, pp. 3-16, 2014.

[19] S. Cruz, S., F. Q. da Silva, and L. F. Capretz, "Forty years of research on personality in software engineering: a mapping study," *Computers in Human Behavior*, vol. 46, pp. 94-113, 2015.

[20] R. Valencia-Garcia, F. Garcıia-Sanchez, D. Castellanos-Nieves, J. T. Fernandez-Breis, and A. Toval, "Exploitation of social semantic technology for software development team configuration," *IET Software*, vol. 4, no. 6, pp. 373-385, 2010.

[21] N. R. Mead, "Software engineering education: How far we've come and how far we have to go," *Journal of Systems and Software*, vol. 82, no. 4, pp. 571-575, 2009.

Muhammad Zahid Tunio https://orcid.org/0000-0002-9406-0586

He received his B.E. in Computer Systems Engineering and Masters of Engineering in Information Technology from Mehran University of Engineering and Technology, Jamshoro, Pakistan, in 2000 and 2010, respectively. He has 15 years experience in teaching at graduate and Post graduate level. currently, he is a Ph.D. scholar at School of Software Engineering, Beijing University of Post and Telecommunication, China. He is also Assistant Profesor in Department of Computer Systems Engineering at Dawood University of Engineering and Technology, Karachi, Pakistan. His research interests are crowdsource software development, pervasive devices, and mobile computing.

Haiyong Luo https://orcid.org/0000-0001-6827-4225

He received the B.S. degree in the Department of Electronics and Information Engineering from Huazhong University of Science and Technology, Wuwhan, China in 1989, M.S. degree in School of Information and Communication Engineering from the Beijing University of Posts and Telecommunication, China in 2002, and Ph.D. degree in Computer Science from the University of Chinese Academy of Sciences, Beijing, China in 2008. Currently, he is Associate Professor at the Institute of Computer Technology, Chinese Academy of Science (ICT-CAS) China. His main research interests are location-based services, pervasive computing, mobile computing, and the Internet of Things.

Cong Wang https://orcid.org/0000-0002-2504-4056

She received the Ph.D. degree in Control Theory and Engineering from the University of Science Technology, Beijing, China in 2002. She was the Vice Director in charge of pattern recognition and intelligent systems between 2002 and 2011 with the Intelligent Science and Technology Research Center, Beijing University of Posts and Telecommunication. Currently, she is a Professor with the School of Software Engineering, Beijing University of Post and Telecommunications. She is also the vice director of the Key Laboratory of Trustworthy Distributed Computing and Services, Ministry of Education. Her research interests include control theory, Internet of Things, and industrial systems engineering.

Fang Zhao https://orcid.org/0000-0002-4784-5778

She received the B.S. degree in the School of Computer Science and Technology from Huazhong University of Science and Technology, Wuhan, China in 1990, M.S. and Ph.D. degrees in Computer Science and Technology from Beijing University of Posts and Telecommunication, Beijing, China in 2004 and 2009, respectively. She is currently Professor in School of Software Engineering, Beijing University of Posts and Telecommunication. Her research interests include mobile computing, location-based services, and computer networks.

Wenhua Shao https://orcid.org/0000-0002-2440-9981

He is currently pursuing the Ph.D. degree with the School of Software Engineering, Beijing University of Posts and Telecommunications, China. His, current main interests include location-based services, pervasive computing, and convolution neural networks and machine learning.

Zulfiqar Hussain Pathan https://orcid.org/0000-0002-3687-6456

He received B.S. in Computer Science and M.S. degrees in Technology Innovation Management from Mehran University of Engineering and Technology, Jamshoro, Pakistan. Since September 2014, he is a PhD scholar at Beijing University of Posts and Telecommunications.