Solving Team Making Problem for Crowdsourcing with Hybrid Metaheuristic Algorithm

Han Wang
School of software
Dalian University of Technology
Dalian, China
Eara@mail.dlut.edu.cn

Zhilei Ren
School of software
Dalian University of Technology
Dalian, China
zren@dlut.edu.cn

Xiaochen Li
School of software
Dalian University of Technology
Dalian, China
li1989@mail.dlut.edu.cn

Xin Chen
School of software
Dalian University of Technology
Dalian, China
chenxin4391@mail.dlut.edu.cn

He Jiang
School of software
Dalian University of Technology
Dalian, China
jianghe@dlut.edu.cn

ABSTRACT

For a typical crowdsourcing process, a task publisher first publishes a task with an acceptable budget. Then hundreds of crowdsourced workers apply for the task with their desired bids. To recruit an adequate Crowdsourced Virtual Team (CVT) while balancing the profits of the task publisher and crowdsourced workers, previous studies proposed various algorithms, including Genetic Algorithm (GA), Alternating Variable Method (AVM), etc. However, the performance is still limited. In this study, we propose a novel hybrid meta-heuristic algorithm CVTMaker to help publishers identify ideal CVTs. CVTMaker is effective which combines (1+1) Evolutionary Strategy ((1+1)-ES) and AVM to search solutions. Experimental results show that CVTMaker significantly outperforms GA and AVM over 3,117 and 5,642 of the 6,000 instances respectively.

CCS CONCEPTS

• **Software and its engineering** → **Search-based software engineering**;

KEYWORDS

Virtual Team Making, Crowdsourcing, Evolution Strategy, Local Search, hybrid meta-heuristic Algorithm

1 INTRODUCTION

Recent years have witnessed great advantages of crowdsourcing to facilitate software engineering activities, e.g., crowdsourced software development, crowdsourced testing. Making a Crowdsourced Virtual Team (CVT) is one of the most important problem in crowdsourcing software engineering, which aims to select a set of workers from the crowd to build a team for software tasks. In a high-quality CVT, team members gain payments that are close to their requested payments and match their capabilities. Meanwhile, the total payments and the actual teamsize are under the constraint of the predefined budget bound and team size bound respectively. One of the most popular way to solve software engineering task is to use search based algorithm [2]. Tao et al. [3] attempt various simple search based algorithms to construct high-quality CVTs, but the performances are still limited. In this paper, we proposed a hybrid meta-heuristic algorithm CVTMaker for the CVT problem, which combines (1+1) Evolution Strategy ((1+1)-ES) with 1/5 success rules and Alternating Variable Method.

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07...$15.00
https://doi.org/10.1145/3205651.3205796

2 PROBLEM DEFINITION

On a crowdsourcing platform C, there are m workers in C , i.e., $C = \{r_1, r_2, \cdots, r_m\}$, Each worker i has an attribute $Value_i$ to qualify his capability, which can be calculated by the average of four factors of this worker, including, *Experience*, *SuccessfulRating*, *PaymentHistory*, and *CustomerRating*. The *Experience* is the number of historical task competed by a worker. The *SuccessfulRating* is the rate of historical tasks successfully participated by a worker. The *PaymentHistory* is the average payment of a worker in history. The *CustomerRating* is the ranking of a worker on C. These four factors measure both the experience and skill of a worker, which are normalized between 0 and 1 by the largest values.

A task publisher submits a task by defining a budget *Budget* and a team size $TeamSize$. Each worker i who is willing to complete the task, requests a payment $RBid_i$. The solution contains a group of workers who bid for the task $V = \{r_1, r_2, \cdots, r_n\}(n < m)$ and a set of payment $\{Bid_1$, Bid_2, $\cdots$, $Bid_n\}$ for each of them, where n is the size of the crowdsourced virtual team. The constraints and fitness function of the CVT problems are shown as follows:

Constraint 1.The actual team size n and payments should be controlled within the reasonable ranges $[TeamSize - 1, TeamSize + 1]$ and $[0.9 * Budget, 1.1 * Budget]$ respectively. Thus, we define the distance from actual payment to predefined budget as follow:

$$f_{distBudget}(n) = \begin{cases} nor(Budget_{min} - \sum_{i=1}^{n} Bid_i) & \sum_{i=1}^{n} Bid_i < Budget_{min} \\ 0, & Budget_{min} \le \sum_{i=1}^{n} Bid_i \le Budget_{max} \\ nor(\sum_{i=1}^{n} Bid_i - Budget_{max}), & Budget_{max} < \sum_{i=1}^{n} Bid_i \end{cases} \quad (1)$$

Constraint 2. The payment for each worker Bid_i should be close to the required payment by the worker $RBid_i$. We use $f_{bidGap}(n)$ to indicate the average bias between Bid_i and $RBid_i$ of all workers in a CVT. $f_{bidGap}(n)$ is calculated as:

$$f_{bidGap}(n) = \frac{\sum_i^n nor(|RBid_i - Bid_i|)}{n} \quad (2)$$

where $nor(\cdot)$ is a normalization function calculated as $nor(x) = x/((x+1))$. According to [1], $nor(\cdot)$ is robust than other normalization functions in search-based software engineering.

Constraint 3. The actual payments offered by the publisher should be fair to the values of the workers. We use the following formula to calculate the matching degree between the actual payments and workers' values $f_{similarity}(n)$:

$$f_{similarity}(n) = \frac{\sum_i^n nor(|Value_i * Budget_{max} - Bid_i|)}{n} \quad (3)$$

Fitness function. Based on these three constraints, the fitness function of this problem is:

$$f_{Fitness}(n) = \frac{f_{distBudget(n)} + f_{bidGap}(n) + f_{similarity}(n)}{3} \quad (4)$$

$f_{Fitness}(n)$ ranges from 0 and 1. The objective of this study is to selecting a set of workers that minimizes $f_{Fitness}(n)$.

3 METHOD

Given that hybrid algorithms are a promising way of obtaining robust and better solutions, we attempt hybrid metaheuristic to resolve the CVT problem. We adopt (1+1) Evolutionary Strategy with 1/5 success rule as the global search algorithm and Alternating Variable Method as the local search algorithm. Based on the 1/5 success rule, CVTMaker can quickly find the direction of the global optimal solution. One of the main drawbacks of evolutionary strategy is that it oscillates near the optimal solution later. Then, the local search algorithm AVM is applied to search the local space of the current best individual, which optimize each dimension in turn. The pseudo code of CVTMaker is presented in Algorithm 1.

Algorithm 1: CVTMaker

Input: instance CVT (budget,teamsize,volunteers), max number of iteration w, max number of iteration for AVM;
Output: solution V*;

i ← 0, σ ← (maxIndex−minIndex)/2 , $\alpha \leftarrow 2^{\frac{1}{teamsize_{max}}}$;
randomly generate an initial individual, i ← i + 1;
while $i < w$ **do**
  update the parent with ES as offspring $V^{(i+1)}_{offspr} \leftarrow V^{(i)}_{parent} + N(0, {\sigma^{(i)}}^2 I)$;
  if $f(V_{offspring}) > f(V_{update})$ **then**
    $V_{offspring} \leftarrow V_{update}, \sigma^{(i+1)} \leftarrow \sigma^i \cdot \alpha$;
  else
    $\sigma^{(i+1)} \leftarrow \sigma^i \cdot \alpha^{-1/4}$;
  if $f(V_{offspring}) = 0)$ **then**
    return $V_{offspring}$;
  j=0;
  while $j < 100$ **do**
    for *variable* r_i *in individual* $V_{offspring}$ **do**
      directions ← {-1, +1}, delta ← 1
      for *d in directions* **do**
        $V_{temp} \leftarrow V_{offspring}$;
        $r_i \leftarrow r_i + d * delta$;
        if $f(V_{offspring}) < f(V_{temp})$ **then**
          $delta \leftarrow delta * 2$;
        else
          $V_{offspring} \leftarrow V_{temp}$, $r_i \leftarrow r_i - d * delta$;
        j ← j+1
  i←i+1;

Table 1: Results for Wilcoxon signed-rank test and Vargha and Delaney $\hat{A}_{12}$ test on 6000 instances

CVTMaker vs.	$\hat{A}_{12}$			wilcoxon		
	A > B	A < B	A = B	A > B	A < B	A = B
GA	5,097	861	42	3,117	303	2580
AVM	5,753	231	16	5,642	124	234
(1+1)-ES	5,584	377	39	3,601	51	2,348

4 EXPERIMENTS AND RESULTS

We employee the benchmark with 6,000 instances from [3] to evaluate the performance of CVTmaker and the baselines including GA, AVM and (1+1)-ES. We set the number of iterations for all algorithms as 2,000 and run each algorithm 100 times. We adapt the Vargha and Delaney statistics ($\hat{A}_{12}$) to calculate the effect size of these algorithms [3]. When comparing arrays of A and B, the value of $\hat{A}_{12}$ indicates the probability that array A gets larger values than array B. As mentioned in Section 2, the objective of the CVT problem is to minimize the fitness function. Thus, when $\hat{A}_{12}$ is lower than 0.5, it means that algorithm A is more likely to achieve better solutions than algorithm B and vice versa. Furthermore, we conduct the Wilcoxon signed-rank test on the results to check how many instances are significantly better than the baselines. We choose the confidence level as 95%. As shown in Table 1, out of the 6,000 instances, CVTMaker significantly outperforms GA, AVM and (1+1)-ES over 3,117, 5,642 and 3,601 instances respectively.

REFERENCES

[1] Andrea Arcuri. 2013. It really does matter how you normalize the branch distance in search-based software testing. *Software Testing, Verification and Reliability* 23, 2 (2013), 119–147.

[2] He Jiang, Zhilei Ren, Xiaochen Li, and Xiaochen Lai. 2015. Transformed Search Based Software Engineering: A New Paradigm of SBSE. In *International Symposium on Search Based Software Engineering*. Springer, 203–218.

[3] Tao Yue, Shaukat Ali, and Shuai Wang. 2015. An evolutionary and automated virtual team making approach for crowdsourcing platforms. In *Crowdsourcing*. Springer, 113–130.