Distributed ACO Based Reputation Management in Crowdsourcing

Safina Showkat Ara(✉), Subhasis Thakur, and John G. Breslin

Insight Centre for Data Analytics, NUI Galway, Galway, Ireland
{safina.ara,subhasis.thakur,john.breslin}@insight-centre.org

Abstract. Crowdsourcing is an economical and efficient tool that hires human labour to execute tasks which are difficult to solve otherwise. Verification of the quality of the workers is a major problem in Crowd sourcing. We need to judge the performance of the workers based on their history of service and it is difficult to do so without hiring other workers. In this paper, we propose an Ant Colony Optimization (ACO) based reputation management system that can differentiate between good and bad workers. Using experimental evaluation, we show that, the algorithm works fine on the real scenario and efficiently differentiate workers with higher reputations.

Keywords: Crowdsourcing · Reputation · Ant Colony Optimization · Decentralised social network

1 Introduction

Crowdsourcing is an economical and scalable tool that allows one to hire workers to perform certain tasks which are difficult to perform with a computer program. Workers of a crowdsourcing platform have different levels of expertise. Due to the unknown level of expertise of the workers, it is difficult to ensure quality control on crowdsourced tasks. The difficulty of verification of the quality of workers of crowdsourcing originates as it is an example of the principle agent problem [10]. The only way to verify the performance of the workers is to employ other workers to perform the same task.

Selecting a resourceful an efficient worker is a very crucial challenge in crowdsourcing. For any types of crowdsourcing, the reputation of a worker is very helpful to analyse the performance of the worker and for selecting appropriate worker [5]. In this paper, borrowing the idea of Ant Colony Optimizations, we propose a reputation management algorithm for generating the reputation of the workers in the crowd. Ant-Colony Optimization (ACO) is a heuristic design proposed by Dorigo [7]. In ant colony optimization Ants deploy pheromone trail as they walk; this trail guides other ants to choose the path that has the higher pheromone. We create a reputation management algorithm based on the fundamental idea of ACO to find some resources in crowdsourcing. Alike the ACO's random walk, to initiate the search process each worker will have a random walk

R. Montella et al. (Eds.): IDCS 2019, LNCS 11874, pp. 379–390, 2019.
https://doi.org/10.1007/978-3-030-34914-1_36

in their neighbourhood and explore the first resourceful neighbour and afterwards, other workers will follow the trail of the search. The contribution of our work lies in two points: first, design and implement the algorithm. The second is to evaluate the performance of the algorithm. Here we have used [1] dataset and arranged it as a decentralised social network, and on this social network, we executed the search for the workers with higher reputation.

The rest of the paper has been organized as below. Section 2 states the problem definition, including the search approach. In Sect. 3, we Develop our algorithm for the reputation management and we are showing an experimental evolution of the algorithm in Sect. 4. In Sect. 5, we have discussed the related work that has done on reputation management for crowdsourcing and in Sect. 6 we conclude our work.

2 Problem Statement

Each crowd-sourcing job is executed in the following sequence. Let there be k tasks $(t_1, t_2, t_3, \ldots t_n)$. Tasks are executed in the sequence of their subscripts. Let there be n workers $w = (w_1, \ldots, w_n)$. $w^g \subset w$ and $w^b \subset w$ will indicate the sets of good and the bad workers such that $w^g \cup w^b = w$ and $w^g \cap w^b = \emptyset$. We assume that, a good worker performs each task correctly and a bad worker randomly chooses an answer. For each task $\alpha \geq 1$ workers are chosen from w uniformly at random. We denote the set of chosen workers for task i as $w(i) \subset W$. In each task, every worker gets $\beta \geq 1$ options to choose from as the solution. From the above mentioned execution of k crowd-sourcing tasks we form a network among the workers as follows:

1. We create a graph G with n vertices $v = (v_1, \ldots, v_n)$. Next, we add the edges as follows.
2. For each task say t_i, for each worker $w_x \in w(i)$ we do the following:
   (a) We add an edge (w_x, w_y) for each $w_y \in w(i)$ (if the edge is not added previously).
   (b) If the answer of w_x matches the answer of w_y then we increase the weight of the edge by 1 (initially weights of all edges is 0). Otherwise we decrease the edge weight by 1.
3. After the construction of the graph G, we create G^P (the positive graph) and G^N (the negative graph) as follows:
   (a) G^P and G^N has the same vertex set as G.
   (b) If the weight of an edge is positive the it is added to G^P and if it is negative then it is added to G^N.

We will illustrate the above mentioned graph construction process with a small simulation. Let there are 100 workers and among them only 10 are good workers. We construct the positive and the negative graph after 50 tasks using the above mentioned procedure. The graphs are shown in Fig. 1. Note that, we have used two parameters α and β. They control the graph formation process as follows:

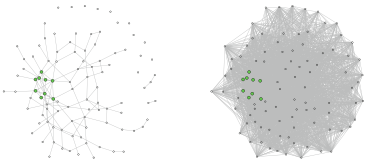

Fig. 1. The positive graph is shown in the left-hand side and the negative graph is shown in the right-hand side. The set of good workers are shown as the green colored vertices. (Color figure online)

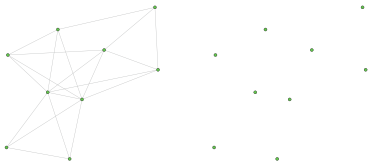

Fig. 2. The induced sub graph (with the vertices representing the good workers) on the positive graph is shown on the left-hand side and the induced sub graph (with the vertices representing the good workers) on the negative graph is shown on the right-hand side.

1. By increasing α (number of workers performing the same task) we can construct the graph quickly. But α depends on the budget for executing the tasks. In a low budget task, we may not be able to employ many workers.
2. β is the number of candidate solutions for each task. If the value of β is increased then, it is more likely that the answers of the bad workers with any other worker will not match. Hence the negative graph will get dense (also the total weight of its edges will increase). The growth of the positive graph will become slow with higher values of β.

Our objective is to identify the vertices belonging to the good workers from the above constructed graphs. We can do so by identifying certain characteristics of the vertices owned by good workers. Let $v^g \subset v$ be the vertices belonging to the good workers. The induced sub graph by v^g on G^P is a connected graph (after certain number of tasks). This is because, the good workers always provide correct answer. Thus their answer matches and the weight of the edges among them increases. The induced sub graph by v^g on G^N is always an empty graph. This is because the good workers do not contradict each other. We consider these as the basic characteristics of these vertices.

The above mentioned characteristics of the vertices belonging to the good workers is shown in Fig. 2. The graphs G^P and G^N changes as more and more tasks are executed. The characteristics of the vertices owned by the good workers is as follows:

1. We consider two parameters, (a) the weight of the edges in the sub graph induced by v^g on the positive (negative) graph and (b) the total weight of edges from vertices $v - v^g$ to v^g. The first parameter indicates the cohesiveness of the answers provided by the good workers. The second parameter indicates the 'difficulty' to separate a good worker from a bad worker. A Higher value for the second parameter means that answers of the good and the bad workers have matched and hence it is difficult to distinguish among them.

2. We evaluate the simulation results of 100 workers (with 10 good workers) with the interval of 30 tasks. The observations are as follows:
   - The weight of the sub graph induced by v^g on the positive graph increases as more and more tasks are executed. This is because good workers agree with each other's answer.
   - Weight of the edges from $v - v^g$ to v^g on the positive graph initially increase and then after the execution of certain number of tasks they decrease. This is because, as the number of times a bad worker participates in the same task with a good worker increases it gets less likely that their answers will match at every instance.
   - The weight of the edges from $v - v^g$ to v^g on the negative graph decreases.

Based on the above mentioned characteristics of the graphs we define the problem of identification of the good workers as follows:

- Partition the vertices v into groups as $\pi = (\pi_1, \ldots, \pi_z)$, such that $\pi_1 \cup \pi_2 \cup \cdots \cup \pi_z = v$, $\pi_i \cap \pi_j = \emptyset$ for all $\pi_i, \pi_j \in \pi$ and there is no $\pi_i \in \pi$ such that $\pi_i = \emptyset$. Each π_i induces a connected graph in G^P and induces an empty graph in G^N. The total weight of induced sub graphs on G^P is the maximum.

3 Algorithm

Algorithm 1 Ant colony algorithm for partitioning

$G^P = (V, E^P)$ and $G^N = (V, E^N)$ be the positive and the negative graph respectively.
A partition over V's of G^P $Ant_i \leftarrow$ be the ant at v_i
Each round Each ant Ant_i $Walk_i \subset V$ be the set of vertices generated by a walk.
$P \leftarrow v_i$, $Q \leftarrow \emptyset$
Each $v_x \in Walk_i$ $G_1 \leftarrow Induced - Subgraph(G^N, \{v_x, P\})$
$G_2 \leftarrow Induced - Subgraph(G^N, \{v_x, Q\})$
$|E(G_1)| > 0$ Add v_x to Q
$|E(G_2)| > 0$ Add v_x to P

% Increase pheromones on edges among P %
Each edge e_i in $Induced - Subgraph(G^P, P)$ $e_i\$Ph \leftarrow (1-\alpha)e_i\$Ph + \alpha\beta(1 - exp^{-e_i\$weight})$ % Decrease pheromones on edges between P and Q %
Each edge e_i in edges between P and Q in G^P $e_i\$Ph \leftarrow (1-\alpha)e_i\$Ph$

% After every R rounds%
Delete the edges with an amount of pheromone less than the average pheromone on all edges.

We use an ant colony algorithm to find the partition. The algorithm is as follows:

1. We create an ant for each vertex. The objective of each ant is to construct a neighborhood surrounding its vertex so that it contains vertices that induce a connected maximum weighted sub graph in G^P and that induce an empty graph in G^N.
2. The number of vertices of G^P can be very high and hence we restrict the exploration of each ant within a limited surroundings around its vertex. δ (positive integer) will be used for such restriction.
3. At each round, each ant makes an random walk of length δ starting from its vertex. The random walk is as follows: Each vertex is added to the random walk as follows:
   (a) Generate a random number between $(0, 1)$ uniformly at random and if it is less than .5 then choose a neighbor (in G^P) of the last vertex in the walk uniformly at random.
   (b) Else, order the neighbors of the last vertex(v_x) in the random walk with decreasing weight, calculated as follows:

$$\mathbf{Pheromone - level} * (\mathbf{edge - weight})^{\mathbf{x}}$$

   where x is ≥ 1.
4. After constructing a random walk for each ant, we partition the vertices of the walk into two sets P and Q. P contains the vertices which can induce a connected graph in G^P and empty graph in G^N. Other vertices in the random walk are added to Q.
5. We increase the pheromone level among the edges in P with the following formula:

$$\mathbf{e_i(Ph)} \leftarrow (\mathbf{1} - \alpha)\mathbf{e_i(Ph)} + \alpha\beta(\mathbf{1} - \mathbf{exp}^{-\mathbf{e_i\$weight}})$$

6. We decrease the pheromone level on the edges among P and Q with the following formula:

$$\mathbf{e_i(Ph)} \leftarrow (\mathbf{1} - \alpha)\mathbf{e_i(Ph)}$$

7. After every R rounds, we delete edges whose pheromone level is less than the average pheromone level in G^P.

4 Experimental Evaluation

In this section, we present the experimental evaluation of our proposed crowdsourcing algorithm on both synthetic and real datasets. We implemented algorithms where in each iteration we reduced the social graph with number of bad worker with the lowest score and recomputed the score for the remaining workers. We conducted this evaluation with three significant datasets. Firstly, We prepared a synthetic training dataset for checking the accuracy confirmation of the algorithms. Secondly, we used a synthetic test dataset for the experiment with 100 and 500 workers, where good and bad workers are unknown. Finally, we prepared the dataset with the text of 10% of questions and answers from the Stack Overflow programming Q&A website and Stack Overflow website datasets, to evaluate the performance of our reputation management algorithms on real data.

Algorithm 2 Random walk algorithm

$G^P = (V, E^P)$ and $G^N = (V, E^N)$ be the positive and the negative graph respectively,v_i be the start location and δ be the length of the walk. A random walk of length δ from v_i. $Walk_i \leftarrow \emptyset$
$Next \leftarrow v_i$
Each $iin[1:\delta]$ $Random(1) > .5$ $N(Next)$ be the neighbors of $Next$ in G^P which are not in $Walk_i$
Wt be a matrix with 1 row and $|N(Next)|$ columns
$x \in [1 : |N(Next)|]$ e_x be the edge between x and $Next$ in V^P
$wt[x] < -e_x\$Ph * (e_x\$Weight)^\gamma$
$v_y \in N(Next)$ be the neighbor with maximum wt
Add v_y to $Walk_i$ and $Next \leftarrow v_y$
v_x be a neighbor of $Next$ and $\notin Walk_i$
Add v_x to $Walk_i$ and $Next \leftarrow v_x$

Return $Walk_i$

4.1 Modeling and Simulation

We generate a graph for the simulation, considering the users/workers as node and their weight as the score for the calculation of reputation. Users/workers are connected to other users/workers following the relation that exists in the stack overflow i.e. users create an edge with other user who answered a question or commented on the question. We model the dynamic crowd-sourcing platform as follows:

Growth rate: At every step ρ (positive integer) new workers are registered and available for work. Let $\alpha\rho$ and $(1-\alpha)\rho$ be the sets of good and bad new workers.

Decay rate: ω is the decay rate. At every round ω workers becomes inert, i.e., do not participate in crowd sourcing anymore. We assume that the probability that a worker becomes inert depends on the number of jobs it has completed. For example at step t, if x is the total number of jobs executed by all active workers and y is the number of jobs executed by the worker w_i then the probability that w_i becomes inert at step $t+1$ is y/x.

Crossing time/steps: Using following equations we are calculating the crossing time t, along with the expectation of edges with a good worker E and expected increase of weight W. n_1 be the number of good workers. The probability that a good worker is chosen in a job is $\frac{\alpha}{n}$. In the same job the probability that $k \leq \alpha - 1$ good workers are also chosen is: $(\frac{n_1-1}{n-1})^k$.

Expectation of edges with good workers is shown in the Eq. 1:

$$E = \frac{\alpha}{n}(1\frac{n_1-1}{n-1} + 2(\frac{n_1-1}{n-1})^2 + \cdots + k(\frac{n_1-1}{n-1})^k) \quad = \frac{\alpha}{n}(\frac{a}{1-r} + \frac{rd}{(1-r)^2})$$
$$= \frac{\alpha}{n}(\frac{1}{1-(\frac{n_1-1}{n-1})} + \frac{1(\frac{n_1-1}{n-1})}{(1-(\frac{n_1-1}{n-1}))^2})$$
$$= \frac{\alpha}{n}(\frac{n-1}{n-n_1} + \frac{(n_1-1)(n-1)}{(n-n_1)^2})$$
$$= \frac{\alpha}{n}(\frac{(n-1)(n-n_1+n_1-1)}{(n-n_1)^2})$$
$$= \frac{\alpha}{n}(\frac{(n-1)(n-1)}{(n-n_1)^2}) \qquad = \frac{\alpha}{n}(\frac{(n-1)^2}{(n-n_1)^2}) \tag{1}$$

Expected weight of the graph induced by good workers after t steps is: $tn_1\frac{\alpha}{n}(\frac{(n-1)^2}{(n-n_1)^2}) = tZ$, Where $Z = n_1\frac{\alpha}{n}(\frac{(n-1)^2}{(n-n_1)^2})$. The probability that a good worker will meet k bad workers in a job is: $(\frac{\alpha}{n}(\frac{n-n_1}{n})^k)$. The expected increase of weight is:

$$W1 = \frac{\alpha}{n}(1(\frac{n-n_1}{\beta n})^1 + 2(\frac{n-n_1}{\beta n})^2 + \cdots + k(\frac{n-n_1}{\beta n})^k) \tag{2}$$
$$= \frac{\alpha}{n}(\frac{a}{1-r} + \frac{rd}{(1-r)^2}) \tag{3}$$
$$= \frac{\alpha}{n}(\frac{1}{1-\frac{n-n_1}{\beta n}} + \frac{\frac{n-n_1}{\beta n}}{(1-\frac{n-n_1}{\beta n})^2}) \tag{4}$$
$$= \frac{\alpha}{n}(\frac{\beta n}{\beta n - n + n_1} + \frac{\beta n(n-n_1)}{(\beta n - n + n_1)^2}) \tag{5}$$
$$= (\frac{\alpha\beta^2 n}{(\beta n - n + n_1)^2}) \tag{6}$$

At step t the total expected increase in weight is: $((t)(n_1)(\frac{\alpha\beta^2 n}{(\beta n-n+n_1)^2}))$. $w_x \in W^G$ and $w_y \in W^B$: w_x has met w_y before and their answer does not match. Step is t. Probability of choosing both w_x and w_y in one step is: $(\frac{\alpha}{n})^2$.
Hence the probability that they met before at least once in the last $t-1$ steps is: $((t-1)(\frac{\alpha}{n})^2)$. The probability that their answers do not match is $(\frac{\beta-1}{\beta})$. Thus the probability that they have met before and their answers do not match is $((t-1)(\frac{\alpha}{n})^2\frac{\beta-1}{\beta})$. Now the expected loss of weight due to one good worker is:$((n-n_1)(t-1)(\frac{\alpha}{n})^2\frac{\beta-1}{\beta})$, and the expected loss from all good workers is : $((n_1)(n-n_1)(t-1)(\frac{\alpha}{n})^2\frac{\beta-1}{\beta})$. Thus the weight W_1 after t steps is:

$$= tX - (t-1)Y \tag{7}$$

where $(X = n_1(\frac{\alpha\beta^2 n}{(\beta n - n + n_1)^2}))$ and $(Y = (n_1)(n - n_1)(\frac{\alpha}{n})^2 \frac{\beta-1}{\beta})$. At a certain time t, the following holds:

$$t > Y/(Z - X + Y)$$
$$t > 1/(Z/Y - X/Y + 1) \tag{8}$$

$$Hence, t > 1/(\frac{\beta}{\alpha(\beta-1)} - \frac{\beta^3}{\alpha(\beta-1)^3} + 1) \tag{9}$$
$$> (\alpha(\beta-1)^3)/(\beta(\beta-1)^2 - \beta^3 + \alpha(\beta-1)^3) \tag{10}$$

4.2 Experiments

Synthetic Datasets: We simulate a social network with 500 workers to generate the negative and positive sub graph. We assume that 10% of workers have a with high reputation score. We analysed the performance of our reputation management algorithm on random partitioned graph, After collecting a list of each worker's resourceful neighbors from random search, the algorithm partitions the graph into a negative and positive sub graphs. In this scenario, the worker-reputation graph forms inevitably after the update of worker performance scores.

Real Datasets: We evaluate our algorithm on some standard datasets. We combine [1] and [2] datasets. It is observed that the best accuracy achieved when up to 500 workers are filtered using our reputation management algorithms. The dataset contain questions and answers of the user with their score. In our setup we simulate the scenario in a distributed manner such that when a question triggers in the system the reputation management algorithm floods the question to users with higher score as good workers. In this way after each question and answer the system will update the score of the users. Figure 3 describe the comparison of negative and positive graph and show they way we are accumulating good workers. Figures 4 and 5 shows the outcome of the reputation management algorithm executed with synthetic data. Moreover, We execute the simulation with the identified good and bad users in Stack-Overflow dataset. Figures 7 and 9 shows the outcome for the execution of 100 task with 100150 Users of Stack-Overflow, and the algorithm allocate reputation to 90% and 80% of the good worker respectively. Figures 6, 8 and 9 shows the reputation of 500 workers after the execution of 100 tasks with both dataset. The algorithm allocate 50% and 80% of the good workers respectively with 1% and 2% of the bad workers reputation which are greater than the average reputation of all workers.

It clearly shows that the difference between the reputation of good and bad workers is much higher than the difference between the score of the good and the bad worker. Hence, using our algorithm of reputation management it is easy to distinguish between a good and a bad workers.

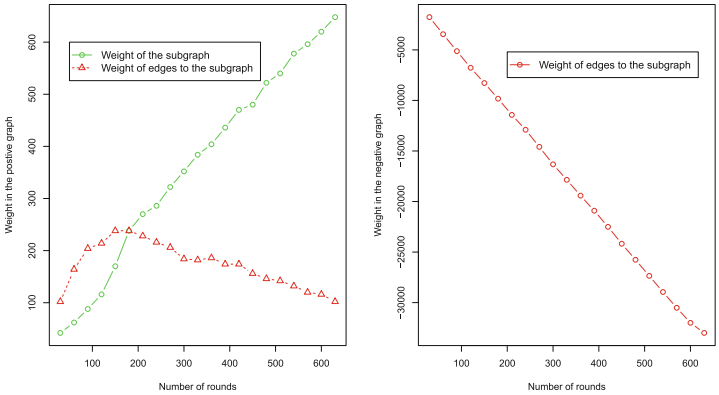

Fig. 3. Graph on the left-hand side: The green points indicates the weight of the induced sub graph on the positive graph by the vertices representing the good workers. The red points indicates the weight of the edges from any vertex (not good workers) to the good workers. Graph on the right-hand side: The red points indicates the weight of the edges from any vertex (not good workers) to the good workers. (Color figure online)

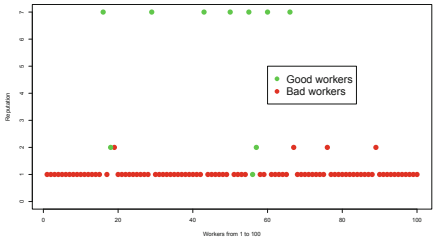

Fig. 4. Shows the reputation of the workers after the execution of 120 tasks. The algorithm allocate high reputation to 70% of the good workers.

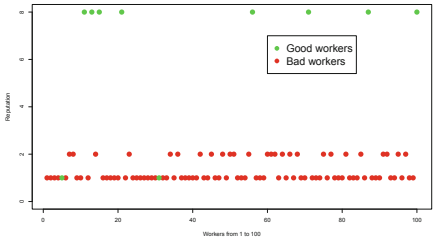

Fig. 5. Shows the reputation of the workers after the execution of 150 tasks. The algorithm allocate high reputation to 80% of the good workers.

5 Related Work

In search for a good optimization algorithm in crowdsourcing we explore the concept and method used for optimisation. [21] solves an optimization problem that includes various factors as interest of different stakeholders in the CS and trade-off between the quality and the quantity of the CS. [13,19] proposed a trust evaluation model to differentiate honest workers and dishonest workers. Classification of trust on the workers based on various contexts, i.e., type of task and task reward amount. [8] designs matrices to calculate trust which includes various parameters regarding software development process. [11,20] present a model for deriving the correct answers along with difficulty levels of questions and ability levels of participants in multiple problem domains. They are also claiming that the joint inference of correct answers is the key attribute a high

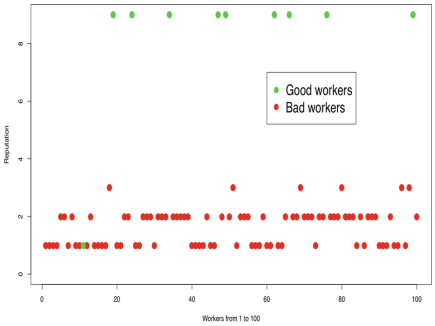

Fig. 6. Shows the algorithm allocate high reputation to 90% of the good workers in synthetic data.

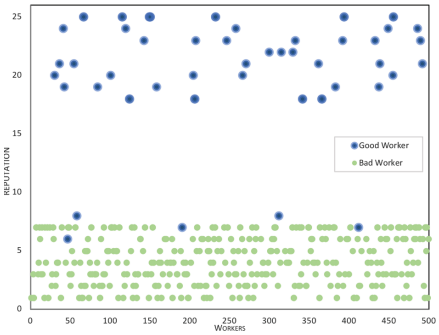

Fig. 7. Shows the algorithm allocate high reputation to 90% of the good workers in real data.

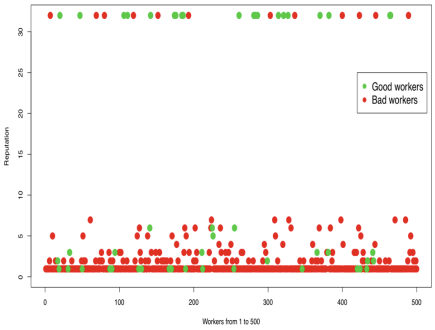

Fig. 8. Shows the algorithm allocate 50% of the good workers and 1% of the bad workers more reputation than the average reputation of all workers.

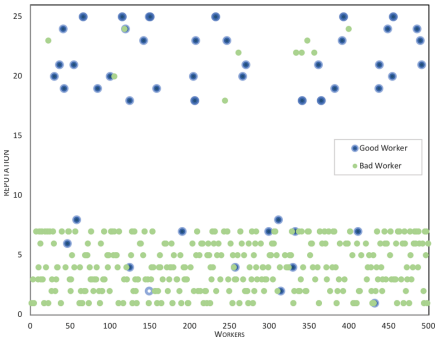

Fig. 9. Shows the algorithm allocate 80% of the good workers and 2% of the bad workers more reputation than the average reputation of all workers in real data.

level of accuracy. [17] designed a method for real-time, automatic prediction of the quality of submissions to a knowledge base, their model immediately verifies the accuracy of submission. [3,12] develop a fair incentive mechanism for crowdsourcing that enhances the operation of crowd sourcing for both task authors and contributors. [4] develop an algorithm to search expertise in decentralised social network and provide incentives for them. It is a general problem in crowd sourcing to map the set of users and a set of binary choice questions with the truthful answer. [6] generalized setting of the problem where the user–question graph can be arbitrary. [14] presented a mechanism to induce truthful report for crowd sourced consensus tasks. They devised a scoring system that infer honest reporting of customer reviews and the evaluated honest review found to be a Nash equilibrium. [15] Provided a solution for multi objective Optimization

Problem using a distributed ACO (Ant Colony Optimization) algorithm based on a crowd sourcing model. [7] solves Ant Colonies and the Mesh-Partitioning Problem. Ant-Colony Optimization (ACO) is a heuristic design to optimize the search highly motivated by the nature of real ant's walking and food collection. [16] conducted an in-depth analysis of the reputation system, studying the historical data they presented a method to predict the influential long-term contributors. [18] detected local communities around a trusted node for the defence of Sybil attack. [9] presented a model for worker's decision that whether the worker will compute the task.

6 Conclusions

In this paper, we have developed reputation management algorithm using the ant colony optimization concept. In our experimental evaluation we have shown how the algorithm is giving us separated sets of good workers and bad workers. This is very helpful to identify the reputation of workers in crowd sourcing. Moreover, our customised real dataset for this experimental evaluation is a unique one, using this we can extend our work for the comparison of centralised and decentralised Stack Overflow's reputation management. For future work, we are implementing other optimization algorithms to analyse competitive characteristics of our algorithms.

Acknowledgement. This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/ 2289-P2(Insight) and by a research grant from SFI and the Department of Agriculture, Food and the Marine on behalf of the Government of Ireland under Grant Number SFI/12/RC/3835(VistaMilk), co-funded by the European Regional Development Fund.

References

1. https://stackoverflow.com/
2. https://www.kaggle.com/stackoverflow/stacksample
3. von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2004)
4. Ara, S.S., Thakur, S., Breslin, J.G.: Expertise discovery in decentralised online social networks. In: ASONAM 2017 (2017)
5. Awwad, T.: Context-aware worker selection for efficient quality control in crowdsourcing (2018)
6. Dalvi, N., Dasgupta, A., Kumar, R., Rastogi, V.: Aggregating crowdsourced binary ratings. In: WWW 2013 (2013)
7. Dorigo, S.: Ant Colonies and the Mesh-Partitioning Problem, pp. 203–208. MIT Press, Cambridge (2004)
8. Dwarakanath, A., Shrikanth, N.C., Abhinav, K., Kass, A.: Trustworthiness in enterprise crowdsourcing: a taxonomy and evidence from data. In: ICSE 2016 (2016)

9. Anta, A.F., Georgiou, C., Mosteiro, M.A., Pareja, D.: Algorithmic mechanisms for reliable crowdsourcing computation under collusion. PLoS One **10**, e0116520 (2015)
10. Ho, C.J., Slivkins, A., Vaughan, J.W.: Adaptive Contract Design for Crowdsourcing Markets: Bandit Algorithms for Repeated Principal-Agent Problems (2014)
11. Jagabathula, S., Subramanian, L., Venkataraman, A.: Reputation-based worker filtering in crowdsourcing. In: Advances in Neural Information Processing Systems 27 (2014)
12. Kamar, E., Horvitz, E.: Incentives for truthful reporting in crowdsourcing. In: AAMAS 2012 (2012)
13. Liu, N., Yang, H., Hu, X.: Adversarial detection with model interpretation. In: KDD 2018 (2018)
14. Liu, S., Miao, C., Liu, Y., Yu, H., Zhang, J., Leung, C.: An incentive mechanism to elicit truthful opinions for crowdsourced multiple choice consensus tasks. In: WI-IAT (2015)
15. Lu, J., Pan, L., Liu, S., Liu, X.: Distributed ACO based on a crowdsourcing model for multiobjective problem. In: CSCWD (2017)
16. Movshovitz-Attias, D., Movshovitz-Attias, Y., Steenkiste, P., Faloutsos, C.: Analysis of the reputation system and user contributions on a question answering website: stackoverflow. In: ASONAM 2013 (2013)
17. Tan, C.H., Agichtein, E., Ipeirotis, P., Gabrilovich, E.: Trust, but verify: predicting contribution quality for knowledge base construction and curation. In: WSDM 2014 (2014)
18. Viswanath, B., Post, A., Gummadi, K.P., Mislove, A.: An analysis of social network-based sybil defenses. SIGCOMM Comput. Commun. Rev. **41**, 363–374 (2011)
19. Ye, B., Wang, Y., Liu, L.: Crowd trust: a context-aware trust model for worker selection in crowdsourcing environments. In: ICWS (2015)
20. Bachrach, Y., Graepel, T., Minka, T., Guiver, J.: How to grade a test without knowing the answers - a Bayesian graphical model for adaptive crowdsourcing and aptitude testing. In: ICML 2012 (2012)
21. Yu, H., Shen, Z., Leung, C.: Bringing reputation-awareness into crowdsourcing. In: ICICS 2013 (2013)