Sequential Multi-Class Labeling In Crowdsourcing: A Ulam-Renyi Game Approach

Qiyu Kang
Nanyang Technological University
School of Electrical and Electronic Engineering
KANG0080@e.ntu.edu.sg

Wee Peng Tay
Nanyang Technological University
School of Electrical and Electronic Engineering
wptay@ntu.edu.sg

ABSTRACT
We consider a crowdsourcing platform where workers are posed questions by a crowdsourcer, who then uses their responses to determine the hidden state of a multi-class labeling problem. Workers may be unreliable, therefore by designing the questions using error correction coding approaches, the crowdsourcer can achieve a more reliable overall result. We propose to perform sequential questioning in which workers are asked q-ary questions sequentially, and questions are determined based on the workers' previous responses. We propose an optimization framework to determine the best q and questioning strategy to use, subject to a crowdsourcer budget constraint. For a fixed q, this problem is equivalent to finding an optimal questioning strategy to a q-ary Ulam-Rényi game, which is in general intractable. We propose a heuristic to find a suboptimal strategy, and demonstrate through simulations that our solution outperforms another error correction coding strategy that does not utilize previous workers' responses. Simulations also suggest that q can in general be chosen to be much smaller than the number of classes in the multi-class labeling problem.

CCS CONCEPTS
• Human-centered computing → Collaborative and social computing;

KEYWORDS
Crowdsourcing, multi-class labeling, Ulam-Rényi game, cooperative work, question design

ACM Reference format:
Qiyu Kang and Wee Peng Tay. 2017. Sequential Multi-Class Labeling In Crowdsourcing: A Ulam-Renyi Game Approach. In Proceedings of WI ’17, Leipzig, Germany, August 23-26, 2017, 7 pages.
https://doi.org/10.1145/3106426.3106446

1 INTRODUCTION
In a crowdsourcing platform, workers are given a task, like classifying or labeling an object in a picture, to perform. The crowdsourcer then makes a final decision based on the collective answers from all participating workers. For example, in [1], crowdsourcing was used to produce taxonomies whose quality approaches that of human experts. Crowdsourcing platforms like Amazon Mechanical Turk [15] typically has many participating workers. The goal is to make use of the abundance of workers to perform simple but tedious microtasks that do not require much domain expertise. However, workers may be highly unreliable [14, 16]. Therefore, the microtasks are usually designed to be simple binary questions [17]. In this paper, we use the terms microtask and question interchangeably by assuming that workers are always required to answer a question posed by the crowdsourcer. To improve the reliability of the final decision, various inference algorithms and question allocation methods have been proposed. For example, in [7, 9], the authors proposed a task assignment scheme using a bipartite graph to model the affinity of workers for different binary tasks and an iterative algorithm based on belief propagation to infer the final decision from the workers' responses. In [10], the authors proposed an iterative inference algorithm based on a spectral method. They assumed that tasks have different difficulties and workers have different reliabilities using a generalized Dawid-Skene model [20]. How to design the binary tasks or questions for the workers was however not addressed.

A multi-class labeling problem was considered in [8], in which an allocation algorithm is developed to assign tasks to different workers, and an inference method achieving an order-optimal redundancy-accuracy trade-off was developed. Redundancy here refers to assigning the same task to multiple workers and using a majority voting rule to infer the final answer. From coding theory [18, 19], we know that it is possible to further reduce the inference error through the use of error correction codes. Therefore, [17] developed an algorithm called Distributed Classification Fusion using Error-Correcting Codes (DCFEC) using error-correction coding to divide a single multi-class labeling task into binary questions, which are then assigned to the workers. The classification is then done by performing a Hamming distance decoding of all the workers' binary responses, which can be treated as the noisy versions of a coded message. We note that DCFEC is based on a non-feedback coding strategy. One can use a feedback coding strategy to further reduce the misclassification probability. In this paper, we consider the same multi-class labeling problem as [17] but now allow questions posed to the workers to depend on the workers' previous responses. Our goal is to develop a sequential questioning strategy (SQS) to design questions for the crowdsourcing platform.

Our strategy for designing sequential questions for the workers is based on a q-ary Ulam-Rényi game [5, 11, 13]. This is an iterative game in which one player chooses a state out of M possible states, and the other player asks the first player q-ary questions in order...
to determine the state chosen. The questions are asked sequentially, and a question at one iteration can be based on the responses to the questions from previous iterations. The first player may answer some of the questions posed wrongly. The first player’s choice of the state corresponds to the true state of our multi-class labeling problem, while giving a wrong answer to a question corresponds to the unreliability of a worker in our crowdsourcing platform. Our main contributions are the following: (i) We show how to develop a SQS for a single worker based on a binary Ulam-Rényi game, and verify through simulations that our SQS outperforms the DCFECC method in terms of misclassification error. (ii) We develop a $q$-ary SQS for multiple players using a $(M, q, e)$ Ulam-Rényi game and ideas from the DCFECC approach. We show how to find the best parameter $q$ and $e$ in order to achieve an optimal trade-off between the misclassification error and the number of iterations of questions required. As finding the optimal strategy for a Ulam-Rényi game is in general difficult, heuristics for solving a binary Ulam-Rényi game have been proposed in [5, 11]. However, to the best of our knowledge, strategies for a general $q$-ary Ulam-Rényi game have been proposed only when the number of questions allowed is sufficiently large. In this paper, we propose an efficient heuristic, which is however suboptimal in general.

The rest of this paper is organized as follows. Section 2 develops the mathematical model of the crowdsourcing multi-class labeling problem. Section 3 shows the Sequential Question Strategy. In addition, our heuristic for $(M, q, e)$ Ulam-Rényi game is described. Systems with independent crowding workers and systems with peer-dependent reward schemes are analyzed in Section 4. Simulation results show our SQS approach outperforms DCFECC approach. How to find the best parameter to conduct SQS under the constraints of limited budget is token into consideration in Section 5. Section 6 is conclusion and future work. In this paper, we use $[a, b]$ to denote the set of integers $\{a, a+1, \ldots, b\}$. The notation $p(x \mid y)$ represents the conditional probability of the random variable $x$ given $y$.

2 SYSTEM MODEL

Consider a crowdsourcing platform that allows a crowdsourcer to pose questions or tasks to workers on the platform. In this paper, a crowdsourcer wishes to solve a $M$-ary labeling problem: determine a hidden state $H \in S = \{1, M\} \{2, M\}$ with the help of groups consisting of $N$ workers each. For example, the crowdsourcer wishes to classify an image of a dog into one of $M = 4$ breeds: Dachshund, Golden Retriever, Pembroke Welsh Corgi and Alaskan Malamute. We have $S = \{1, 2, 3, 4\}$, where each state corresponds to a particular dog breed in the order listed. However, since workers may not be experts in such image classification tasks, and can only answer simple multiple choice questions, the crowdsourcer asks each group a sequence of $q$-ary questions, where $q \in [2, M]$. A $q$-ary question can be represented as a $q$-tuple $T = (T_0, T_1, \ldots, T_{q-1})$, where the sets $T_j, j \in [0, q - 1]$, are pairwise disjoint subsets of $S$. The $q$-tuple can be interpreted as a question of the form “Which one of the sets $T_0, \ldots, T_{q-1}$ does $H$ belong to?”. At each iteration, the crowdsourcer poses a $q$-ary question to a group, and this question can be designed based on the responses from previous groups. Each $q$-ary question is assigned to the whole group of $N$ workers, and may be further broken down into simpler microtasks that are assigned to each individual worker within the group. In this paper, we investigate the case where a $q$-ary question is transformed into binary questions, which are then assigned to individual workers within the group. This is because it has been observed that workers on crowdsourcing platforms typically do not have enough expertise to answer non-binary questions reliably [2].

For a concrete illustration, consider again the example of classifying an image of a dog into one of four breeds, where $H = 1, 2, 3, 4$ indicates that the dog in the image is a Dachshund, Golden Retriever, Pembroke Welsh Corgi or Alaskan Malamute, respectively. Suppose that the crowdsourcer uses binary $(q = 2)$ questions. Then the question, “Does the dog in the image have short legs?”, can be represented as the binary tuple $T = (1, 3), (2, 4)$ since Dachshund and Pembroke Welsh Corgi are breeds with short legs, while the other two are not. The crowdsourcer then assigns this question to the first group of workers. Suppose that he receives the answer “Yes”, then his next question could be “Does the dog have erect ears?”, which can be represented as $T = (\{1\}, \{3\})$. By iteratively asking different questions to groups of workers, the crowdsourcer is able to narrow down the exact dog breed in the image.

The response of a worker $k \in [1, N]$ in group $l$ is denoted as $y_{l,k} \in [0, q - 1]$, which indicates to the crowdsourcer which subset in $T$ the worker $k$ thinks $H$ belongs to. We assume that the worker $k$ has an associated reliability $p_{l,k}$, drawn from a common distribution, with mean $\mu_q$. Let $p_l = \{p_{l,k}\}_{k=1}^N$. Conditioned on $H$ and $p_l$, we assume that workers’ responses are independent from each other. The response of worker $k$ in group $l$ has the following probability mass function:

$$p(y_{l,k} \mid H, p_l) = \begin{cases} p_{l,k} & \text{if } H \in T_{y_{l,k}}, \\ \frac{1 - p_{l,k}}{q-1} & \text{otherwise.} \end{cases}$$

The crowdsourcer’s goal is to design a SQS in which $q$-ary questions are posed to the worker groups iteratively. Let $B(q, e)$ be the minimum number of questions required to determine $H$ if at most $e$ of the questions have wrong responses. Since most crowdsourcing platforms like Amazon Mechanical Turk requires the crowdsourcer to financially compensate a worker for each task he completes, the crowdsourcer’s expenditure increases with increasing $B(q, e)$. Therefore, $B(q, e)$ can be used as a proxy for the minimum budget the crowdsourcer needs to spend on the labeling problem if at most $e$ errors occur during the questioning process. Let $R(q, e)$ be the probability that the crowdsourcer finds the hidden state $H$ correctly at the end of $B(q, e)$ questions. We call $R(q, e)$ the reliability of the strategy. The crowdsourcer wishes to maximize $R(q, e)$ while keeping $B(q, e)$ below a given budget. In this paper, our aim is to find a SQS and the corresponding $q$ and $e$ value to

$$\max_{q, e} R(q, e)$$

subject to $B(q, e) \leq b, \quad q \in [2, M], \quad e \in [0, \infty),$

for a given budget $b$. In general, this is an intractable problem due to the integer constraints. Therefore, in the sequel, our goal is to design heuristics that will approximately solve (2), and use simulations to verify the performance of our heuristics.
3 ULAM-RÉNYI GAME AND SQS

Our SQS is based on a \((M, q, e)\) Ulam-Rényi game [5, 11, 13]. In a \((M, q, e)\) Ulam-Rényi game, a responder first chooses a number \(H \in S\), and a questioner asks the responder a sequence of \(q\)-ary questions to determine \(H\). The responder is allowed to lie at most \(e\) times. The questioner chooses his question at each step based on the previous responses, and his goal is to determine \(H\) with the minimum number of questions under the constraint that the responder can lie at most \(e\) times. The Ulam-Rényi game is similar to our crowdsourcing problem: the questioner is our crowdsourcer, while the responder is a group of workers. The difference is that in our crowdsourcing formulation, the workers do not know \(H\) a priori, therefore a wrong answer is modeled to be stochastic and there is no upper limit \(e\) to the number of wrong answers the workers can give. In the following, we first give a brief overview of the Ulam-Rényi game and present a heuristic strategy to solve it. We then apply this strategy to our SQS.

In a Ulam-Rényi game, a game status \(\sigma = (A_0, A_1, \ldots, A_e)\), where \(A_i, i \in [0, e]\), are pairwise disjoint subsets of \(S\). The set \(A_0\) contains all states \(m \in S\) that are potentially \(H\) if the responder has lied exactly \(i\) times. The initial game status \(\sigma\) is given by \((S, \emptyset, \ldots, \emptyset)\). At each question, if the current status is \(\sigma = (A_0, A_1, \ldots, A_e)\) and the answer to the question \(T = (T_0, T_1, \ldots, T_{q-1})\) is \(j\), then \(T_0, T_1, \ldots, T_{q-1}\) are pairwise disjoint sets and \(\bigcup_{j=0}^{q-1} T_j = \bigcup_{i=0}^e A_i\), then the game status is updated as:

\[
\sigma' = (A_0 \cap T_j, (A_0 \setminus T_j) \cup (A_1 \setminus T_j), \ldots, (A_e \setminus T_j) \cup (A_e \cap T_j)).
\]

(3)

For example, in the \((4, 2, 1)\) Ulam-Rényi game, the initial status is \(\sigma = ((1, 2, 3, 4), 0)\). Suppose the responder chooses \(H = 1\) and the questioner’s first question is \(T = (T_1, T_0)\) where \(T_0 = \{1, 2\}\) and \(T_1 = \{3, 4\}\). The responder can lie to this question and answer “\(H\) is in \(T_0\)”, and the questioner updates the game status \(\sigma\) to \((3, 4, 1, 2)\). This game status tells the questioner that if \(H \in \{1, 2\}\), then the responder has lied once, whereas if \(H \in \{3, 4\}\), then the responder has not lied. Suppose the second question posed to the responder is \(T = ((1, 3), (2, 4))\). Since the responder has already lied once, he can only give the answer “\(H\) is in \(T_0\)”. The game status then becomes \(\sigma = ((3, 4), 1, 2)\). Finally, if the questioner poses the question the game status \(\sigma = ((1), (3, 4))\) followed by another question \(T = ((1), (3))\), the game status \(\sigma = (\emptyset, 1)\) is reached, at which point the questioner determines that \(H = 1\).

In general, it is intractable to find an optimal strategy for a \((M, q, e)\) Ulam-Rényi game [5, 12]. The reference [4] proposed a search strategy that is near optimal when the number of states is sufficiently large. In the following, we propose a heuristic adaptive strategy. For each \(i \in [0, e]\), let \(|A_i|\) be the number of states in \(A_i\). We use \(|\sigma|\) to denote \(|(|A_0|, |A_1|, \ldots, |A_e|)\) the type of \(\sigma\) [4]. Same as [12], we define the weight of an element \(x \in A_i\) in status \(\sigma\) with \(w\) iterations left as \(W_\infty(x) = \sum_{j=0}^{q-1} (\frac{1}{q})^|(q-1)^j|\) and the weight of the status \(\sigma\) as the cumulative weight of all the elements in it, \(W_\infty(\sigma) = \sum_{x \in A_i} |A_i| W_\infty(x)\).

From Proposition 3.1 of [3], we have \(W_\infty(|\sigma|) = \sum_{j=0}^{q-1} W_{\infty-1}(|\sigma^j|)\).

Furthermore, suppose that the responder lies at most \(e\) times. Proposition 3.1 of [3] and Theorem 3.1 of [12] show that the questioner is guaranteed to find \(H\) in \(w\) questions only if \(W_\infty(|\sigma|) \leq q^w\). Consider the case where \(W_\infty(|\sigma|) = q^w\) for some \(w\). If the next question yields \(W_{\infty-1}(|\sigma^j|) > q^{w-1}\) for some \(j \in [0, q-1]\), the responder can answer \(j\) to prevent the questioner from finding \(H\). Therefore, the crowdsourcer should use a question \(S\) such that \(W_{\infty-1}(|\sigma^j|)\) is approximately the same for all \(j \in [0, q-1]\). We first write for each \(i \in [0, q-1]\),

\[
T_j = (T_{j,0}, T_{j,1}, \ldots, T_{j,e}),
\]

where \(T_{j,i} = T_j \cap A_i\).\(\cup_{j=0}^e T_{j,i} = A_i\) and \(T_{j,i}, i \in [0, q-1]\), are pairwise disjoint sets. We use \(|T_{j,i}|\) to denote \(|(T_{j,0}, T_{j,1}, \ldots, T_{j,e})|\). We observe from our experiments that \(W_\infty(i)\) is much larger for smaller values of \(i\). For example, when \(q = 4\) and \(e = 7\), we have \(W_\infty(0) = 497, 452\) while \(W_\infty(7) = 1\). This suggests the following heuristic: We first determine the value of \(|T_{j,i}|\) for small \(i\) and then use subsequent \(|T_{j,i}|\) of larger \(i\) to even the weight \(V_{\infty-1}(|\sigma^j|)\). Suppose \(|T_{j,0}|, \ldots, |T_{j,i}|\), for all \(j \in [0, q-1]\) have been determined. Subsequently for \(i \in [0, e]\) and \(j \in [0, q-1]\), let \(|\sigma^j| = (|A_0^j|, |A_1^j|, \ldots, |A_e^j|, 0, \ldots, 0)\), where \(|A_i^j| = |A_{i-1}^j| - |T_{j,i-1}| + |T_{j,i}|\) and \(|A_{i-1}^j| = |T_{j,i-1}| - |T_{j,i-1}| + 1 = 0\). We find \(|T_{j,i}|\) to minimize \(\sum_{i=0}^q (V_{\infty-1}(|\sigma^j|) - V_{\infty-1}(|\sigma^{'j}|))^2\), which can be written as:

\[
\max_{(T_{j,i})_{i=0}^q} \sum_{j=0}^q (V_j - V_j) + (W_{\infty-1}(i) - W_{\infty-1}(i + 1)) (|T_{j,i}| - |T_{j,i+1}|) \tag{4}
\]

subject to \(|T_{j,i}| \in [0, |A_i|], \sum_{i=0}^q |T_{j,i}| = |A_i|,\)

where \(V_j = \sum_{k=0}^q W_{\infty-1}(k) |A_k^j| - W_{\infty-1}(i) |T_{j,i-1}|\) and \(W_{\infty-1}(e + 1) = 0\). After obtaining the optimal \((|T_{j,i}|)_{i=0}^q\) from (4), \(A_i\) is arbitrarily partitioned by randomly choosing \(|T_{j,i}|\) elements from it to form \(T_{j,i}\). The objective function (4) can be rewritten in a Mixed-Integer Quadratic Programming form as:

\[
\min_{x} -\frac{1}{2} x^T Q x + c^T x
\]

subject to \(|T_{j,i}| \in [0, |A_i|], \sum_{i=0}^q |T_{j,i}| = |A_i|,\)

where \(Q \in R^{q \times q}\) has diagonal entries \((q-1) \times (W_{\infty-1}(i) - W_{\infty-1}(i + 1))\), and for \(j \neq k\), the \((j, k)\)-th entry is \(W_{\infty-1}(i + 1) - W_{\infty-1}(i)\). The vector \(c \in R^q\) has \(j\)-th entry \(c_j = \sum_{i=0}^q V_j - V_{j-1}\), and \(x_j\) denotes the \(j\)-th entry of vector \(x\). The above heuristics are presented in detail in Algorithm 1.

To approximately solve (2), we propose a heuristic to compute \(B(q, e)\) in Algorithm 2. The Ulam-Rényi game ends when the game status contains only a single state. Given the initial game status \(\sigma = (S, \emptyset, \ldots, \emptyset)\) and the number of questions \(w\), we use Algorithm 1 to generate a \(q\)-ary tree whose depth is \(w\) and root corresponds to the initial game status. Each node in the tree corresponds to a game status. The \(j\)-th child of a node \(\sigma\) is the status \(\sigma^j\) updated from answer \(j\) as defined in (3). To find \(B(q, e)\) then corresponds to finding the minimum \(w\) that leads to a \(q\)-ary tree in which every leaf node has a game status that contains only one state. We initialize
Table 1: Estimated $B(2,e)$ of the $(M,2,e)$ Ulam-Rényi game where $M = 2^m$.

| $m$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|-----|---|---|---|---|---|---|---|---|
| 1   | 3 | 5 | 7 | 9 | 11| 13| 15| 17|
| 2   | 5 | 8 | 11| 14| 17| 20| 23| 26|
| 3   | 6 | 9 | 12| 15| 18| 21| 24| 27|
| 4   | 7 | 10| 13| 16| 19| 22| 25| 28|
| 5   | 9 | 12| 15| 18| 21| 24| 27| 30|
| 6   | 10| 13| 16| 19| 22| 25| 28| 31|
| 7   | 11| 14| 17| 20| 23| 26| 29| 32|
| 8   | 12| 15| 18| 21| 24| 27| 30| 33|
| 9   | 13| 17| 20| 23| 26| 29| 32| 35|
| 10  | 14| 18| 21| 24| 27| 30| 33| 36|
| 11  | 15| 19| 22| 25| 28| 31| 34| 37|
| 12  | 17| 20| 23| 27| 30| 33| 36| 39|
| 13  | 18| 21| 25| 28| 31| 34| 37| 40|
| 14  | 19| 22| 26| 29| 32| 35| 38| 41|
| 15  | 20| 24| 27| 30| 34| 37| 40| 43|
| 16  | 21| 25| 28| 32| 35| 38| 41| 44|

w as $N_{\min}(e) = \min\{n | M \sum_{j=0}^{e} (\binom{n}{j}(q-1)^j \leq q^n\}$ since $B(q,e) \geq N_{\min}(e)[4, 12]$. We then increment w by one each time to generate a $q$-ary tree. The above procedure is summarized in Algorithm 2.

To verify the performance of our heuristics, we compare our estimated $B(2,e)$ with that of the optimal $(M,2,e)$ Ulam-Rényi game strategy in [5], where $M = 2^m$ is a power of 2. (Note that finding optimal strategies for $q > 2$ is an open problem.) The results are shown in Table 1, in which the numbers in brackets indicate the values found by [5] if our method differs from the optimal values. We see that our algorithm computes $B(2,e)$ correctly in most cases.

Algorithm 1 and 2 can be applied to our SQS for crowdsourcing. In the $M$-ary labeling problem, for a given $(q,e)$, the crowdsourcer is the questioner in a $(M, q,e)$ Ulam-Rényi game, and each group of workers corresponds to the responder. We can think of the group of workers knowing $H$ a priori, and making a mistake in its response to a question as a lie. However, in our crowdsourcing problem, the worker groups are not constrained to making at most $e$ errors. Therefore, after $B(q,e)$ questions, the crowdsourcer finds the correct $H$ only with probability $R(q,e) < 1$. Our goal is to find $(q,e)$ that maximizes $R(q,e)$ with keeping $B(q,e)$ bounded by $b$. In our SQS, the crowdsourcer stops when there is only one element in the status, which may not be $H$ since more than $e$ mistakes could have occurred. We call this the stopping time $\tau \leq B(q,e)$ of our SQS.

In the next two sections, we derive error bounds for the case $q = 2$, and demonstrate that using a SQS outperforms the DCFECC approach. When $q > 2$, since we do not allow sequential questions within each group of workers, we propose a hybrid strategy using the DCFECC approach to decompose each $q$-ary questions into binary questions for each worker group.

### Algorithm 1

**Heuristic strategy for a $(M,q,e)$ Ulam-Rényi game.**

**Input:** Current status $\sigma = (A_0, A_1, A_2, \ldots, A_k)$, with $w$ questions left.

**Output:** $T$

Let $m$ denote the smallest index s.t $|A_m| \neq 0$

$\delta = \text{mod}(|A_m|, q)$; $\omega = \lfloor \frac{|A_m|}{q} \rfloor$;

for $j = 0$ to $\delta - 1$ do

$|T_{j,m}| = \omega + 1$;

end for

for $j = \delta$ to $q - 1$ do

$|T_{j,m}| = \omega$;

end for

$|T_{j,i}| = 0$ for all $j \in [0, q - 1]$ and $i \in [0, m - 1]$.

for $i = m + 1$ to $e$ do

if $|A_i| \neq 0$ then

Compute $V_i$ for all $j \in [0, q - 1]$, solve:

$$\min \frac{1}{2} x^T Q x + e^T x$$

subject to $1^T x = |A_i|$

$x_j \in [0, |A_i|], j \in [0, q - 1]$,

where $Q \in R^{q \times q}$ has diagonal entries $(q - 1) \times (W_{i-1}(i) - W_{i-1}(i+1))$, and for $j \neq k$, the $(j,k)$-th entry is $W_{i-1}(j+1) - W_{i-1}(i)$. The vector $e \in R^q$ has $j$-th entry $e_j = \sum_{k=0}^{q-1} (V_j - V_k)$, and $x_j$ denotes the $j$-th entry of vector $x$.

for $j = 0$ to $q - 1$ do

$|T_{j,i}| = x_i$

end for

else

$|T_{j,i}| = 0$, for all $j \in [0, q - 1]$

end if

$A_j$ is arbitrarily partitioned by randomly choosing $|T_{j,i}|$ elements from it to form $T_{j,i}, i \in [0, q - 1]$ for all $i \in [0, e]$.

### Algorithm 2

**Heuristic to compute $B(q,e)$.**

**Input:** $M, q, e$

**Output:** $B(q,e)$

Set $w = N_{\min}(e) = \min\{n | M \sum_{j=0}^{e} (\binom{n}{j}(q-1)^j \leq q^n\}$.

while $i$ do

Use Algorithm 1 to generate a $q$-ary tree whose depth is $w$ and root is the initial status, as described after (5).

if every leaf node’s game status contains only one state then

break;

else

$w = w + 1$;

end if

end while

return $B(q,e) = w$.
4 Binary Questions and Performance Comparison

In this section, we consider the case where \( q = 2 \), and each worker group has \( N = 1 \) worker. The crowdsourcer designs a sequence of at most \( b \) questions and assign each question to one worker (group) at each time in order to determine \( H \) from all the workers’ responses. In the following, we provide lower bounds for \( R(2, e) \) when workers are independent and when pairs of workers are correlated. These bounds allow us to determine the number of workers required to ensure that \( R(2, e) \) satisfies a given threshold.

4.1 Independent Crowd Workers

When the workers’ responses are conditionally independent given \( H \), we have for all \( i \geq 1 \),

\[
p(y_{l,1} \mid H) = \begin{cases} 
\mu_2 & \text{if } H \in T_{y_{l,1}}, \\
1 - \mu_2 & \text{otherwise}, 
\end{cases}
\]

where \( \mu_2 \) is the mean of the distribution from which \( p_{l,1} \) is drawn from. Let \( S_b \) be the total number of erroneous responses made by the workers over \( b \) questions. Then, for any \( e \in \{ e : B(2, e) \leq b \} \), we have

\[
R(2, e) = \mathbb{P}(S_b \leq e) \\
= \mathbb{E} \left\{ \sum_{i=0}^{e} \binom{e}{i} (1 - \mu_2)^i \mu_2^{e-i} \right\} \\
\geq \sum_{i=0}^{e} \binom{B(2, e)}{i} (1 - \mu_2)^i \mu_2^{B(2, e)-i},
\]

where the last inequality follows because the stopping time of our proposed SQS \( \tau \leq B(2, e) \) and \( f(x) = \sum_{i=0}^{e} \binom{e}{i} (1 - \mu_2)^i \mu_2^{e-i} \) is a decreasing function which can be proved as follows. Rewrite \( f(x) \) as \( f(x, e) = \sum_{i=0}^{e} \binom{e}{i} (1 - \mu_2)^i \mu_2^{e-i} \). We want to prove \( f(x, e) > f(x + 1, e) = \mu_2 f(x, e) + (1 - \mu_2) f(x, e - 1) \), hence we want to prove \( (1 - \mu_2) f(x, e) > (1 - \mu_2) f(x, e - 1) \), which is correct obviously.

4.2 Pair-dependent Crowd Workers

We now consider the case where workers are paired and their rewards are based on the comparative performance among the paired workers [6, 17]. We assume that the responses of the two workers in each pair are correlated with correlation coefficient \( \rho \) and workers in different pairs are independent of each other. A worker \( l \) has a corresponding partner \( l' \). The correlation between \( p_{l,1} \) and \( p_{l',1} \) is given by

\[
\text{corr}(p_{l,1}, p_{l',1}) = \begin{cases} 
0 & \text{if } l \neq l', \\
\rho & \text{if } l = l'.
\end{cases}
\]

It can be shown that for any \( e \in \{ e : B(2, e) \leq b \} \), we have

\[
R(2, e) = \mathbb{P}(S_b \leq e) \\
\geq \sum_{i=0}^{e} \sum_{l=1}^{b+1} \binom{B(2, e)/2}{i} \\
\cdot (1 + \rho \nu^2 + \mu_2^2 - 2 \mu_2) (B(2, e)/2)^j
\cdot (2 \mu_2 - 2 \rho \nu^2 - 2 \mu_2^2) (\mu_2^2 + \rho \nu^2)^{B(2, e)/2-j},
\]

where \( \nu^2 \) is the variance of the distribution of \( p_{l,1} \), and we have assumed \( B(2, e) \) is even for simplicity so that there are in total \( B(2, e)/2 \) pairs.

4.3 Simulation Comparison With DCFECC

In the DCFECC approach proposed by [17], a code matrix is designed to determine what questions are assigned to each worker to solve a \( M \)-ary labeling problem. Suppose that the total number of workers is \( N_c \). A code matrix is a \( M \times N_c \) binary matrix, where each column corresponds to a question \( T = \{ T_0, T_1 \} \) for each worker, and each row corresponds to a code for each state of the \( M \)-ary labeling problem. The responses from all the \( N_c \) workers can then be viewed as a noisy version of one of the rows in the code matrix, and a Hamming distance decoder is used to decode for the hidden state \( H \). A code matrix can be found in [19] by minimizing the following upper bound to the average error probability:

\[
\frac{1}{M} \sum_{i=1}^{M} \sum_{k=1}^{M} \left( \frac{1}{N_c} \left( \sum_{j=1}^{N_c} (a_{k,j} \oplus a_{i,j}) (1 - 2p_{j,1}) \right) \right)^2 \frac{d(a_k, a_i)}{d(a_k, a_i)},
\]

where \( a_i \) is the \( i \)-th row of the code matrix \( A = (a_{i,j}) \), \( d(\cdot , \cdot) \) is the Hamming distance between two vectors, and \( \oplus \) denotes the XOR operation.

We perform simulations using our SQS, the DCFECC and the majority voting (MV) [17] methods for the labeling problem. We use the maximal \( e \) subject to \( B(2, e) \leq b \), where \( B(2, e) \) is approximated by Algorithm 2, to perform SQS simulations. For each set of parameters, we perform 50,000 simulation trials and compute the fraction of times \( H \) is found correctly. In this section, we assume that each group has \( N = 1 \) worker, therefore for fair comparison, we let \( N_c = \lfloor b \rfloor \), where \( \lfloor \cdot \rfloor \) is the average number of workers used in SQS. When workers are independent, we vary the parameters \( b, M \) and \( \mu_2 \). The results are shown in Fig. 1(a)(b). We see that SQS outperforms both DCFECC and MV in all cases. This is because SQS uses sequential questions that depend on the previous responses, whereas in DCFECC and MV, all the questions are designed in advance.

In the pair-dependent workers scenario, we use the Beta(1, 0.5) distribution to generate the worker reliabilities \( p_{l,1} \), \( l \geq 1 \). We vary the correlation coefficient \( \rho \) on the number of workers to perform simulations and show the results in Fig. 1(c) and (d) respectively. We observe that again SQS outperforms both DCFECC and MV in all cases, with the reliability increasing when \( \rho \) increases.

5 Sequential Questioning With Error Correcting Codes

In this section, we consider the case where \( q > 2 \). The SQS developed in Section 3 allows us to pose a \( q \)-ary question to each group of workers at each time. However, since it is preferable to pose binary questions to crowd workers [8], we propose the use of a DCFECC code matrix to treat the \( q \)-ary question as a \( 2^q \)-ary multi-class labeling problem, and design binary questions for each worker in the responder group. We call this the sequential questioning strategy.
Fig. 1: Performance comparison between several methods. (a) Reliability when using independent crowd workers with varying number of workers, different $\mu$ and fixed $M = 32$. (b) Reliability when using independent crowd workers with varying $M$, different $\mu_j$ and the same average number of workers. (c) Reliability when using pair-dependent crowd workers with varying number of workers and fixed $M = 32$. (d) Reliability when using pair-dependent crowd workers with varying $\rho$, fixed $M = 32$ and the same number of workers.

with error correcting codes (SQSECC). The binary responses of the workers in a group are then decoded using a Hamming distance decoder to determine an overall answer $j \in [0, q - 1]$, which is then used in our SQS to determine the next $q$-ary question.

For a $q$-ary question $T = (T_0, \ldots, T_{q-1})$, let $A$ be the corresponding code matrix generated by DCFECC. Let $u \in \{0, 1\}^N$ be the vector of responses from the $N$ workers, which is decoded to $T_j$ if the $l$-th row $a_l$ in $A$ has the smallest Hamming distance to $u$, with ties broken randomly. The set of responses whose Hamming distance to $a_l$ is minimal is called the decision region of $T_j$. It is shown in [17] that for any $i = [i_1, i_2, \ldots, i_N] \in \{0, 1\}^N$, we have

$$P \left( u = i \mid H \in T_j \right) = \prod_{j=1}^{N} \left( 1 - i_j + 2(i_j - 1) \frac{\mu_q a_{lj} + 1 - \mu_q}{q - 1} \sum_{k \neq j} a_{kj} \right).$$

Assuming that all workers are independent given $H$, and $P \left( H \in T_j \right) = 1/q$ for all $j \in [0, q - 1]$, then the probability of getting an erroneous answer to the $q$-ary question is given by [17]

$$P_e(q) = \frac{1}{q} \sum_{j=0}^{q-1} \prod_{i=1}^{N} P \left( u = i \mid H \in T_j \right) C_j^i,$$

with $C_j^i$ being the number of decision regions $j$ belongs to. By asking $B(q, e)$ questions, the final reliability $R(q, e)$ can be approximately lower bounded by

$$\sum_{k=0}^{e} \binom{B(q, e)}{k} P_e(q)^k (1 - P_e(q))^{B(q, e) - k},$$

because $B(q, e)$ is the minimum number of questions required if at most $e$ questions are answered wrongly. To find the best $(q, e)$ in (2) for SQSECC, we optimize the lower bound (12) instead. We find for each $q \in [2, M]$, an $e$ that maximizes (12) subject to $B(q, e) \leq b$, where $B(q, e)$ is approximated by Algorithm 2.

5.1 Performance Evaluation

We perform simulations using $M = 2^7$, $N = 18$, $b = 5$ and

$$\mu_q = \begin{cases} 0.64, & \text{if } q = 2, \\ 0.69 - 0.03q, & \text{if } 3 \leq q \leq 16, \\ 0.2, & \text{if } q > 16. \end{cases}$$

We perform exhaustive search to find the best $(q, e)$ in order to maximize $R(q, e)$ subject to $B(q, e) \leq b$. This is then compared to the lower bound (12) in Fig. 2(a), where the optimal $(q, e) = (4, 1)$ for the lower bound (12) is also optimal for $R(q, e)$, which achieves $R(4, 1) = 0.87$. The average number of workers used in this case is 78.9. For comparison, we use $N_0 = 79$ workers in the DCFECC method, i.e., we construct a $M \times N_0$ code matrix by minimizing (9). The reliability for DCFECC is found to be 0.34, which is worse than the reliability of 0.87 achieved by SQSECC. The MV strategy with 80 workers has reliability 0.25, which is worse than both DCFECC and
We have proposed a heuristic to find a strategy for the game in which a responder answers at most $e$ questions wrongly. We then apply this strategy to our crowdsourcing problem, where we utilize the DCFECC approach [17] to convert the $q$-ary question into individual binary questions for each worker in a group. We obtained a lower bound for the reliability, and used it to find the best $(q,e)$ to use in our strategy, subject to a budget constraint. Simulations suggest that our strategy performs better in terms of reliability than a pure DCFECC approach using the same average number of workers. Future work includes adaptively changing the value of $q$ in the sequential questioning process.

REFERENCES

[1] Jonathan Bragg, Mausam, and Daniel S Weld. 2013. Crowdsourcing multi-label classification for taxonomy creation. In AAAI Conf. Hum. Comput. and Crowdsourcing. AAAI, California, USA, 25–33.
[2] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. 2010. Visual recognition with humans in the loop. In Proc. European Conf. Comput. Vision. Springer, Crete, Greece, 438–451.
[3] Ferdinando Cicalese. 2013. Fault-Tolerant Search Algorithms: Reliable Computation with Unreliable Information. Springer, New York, NY, 9–63 pages.
[4] Ferdinando Cicalese and Christian Deppe. 2007. Perfect minimally adaptive $q$-ary search with unreliable tests. J. Stat. Plan. Inference 137, 1 (2007), 162–175.
[5] Ferdinando Cicalese and Ugo Vaccaro. 2000. An improved heuristic for ‘Ulam–Rényi game’. Inform. Process. Lett. 73, 3 (2000), 119–124.
[6] Shih-Wen Huang and Wai-Tat Fu. 2013. Don’t hide in the crowd! Increasing social transparency between peer workers improves crowdsourcing outcomes. In Proc. SIGCHI Conf. Hum. Factors Comput. Syst. ACM, Paris, France, 621–630.
[7] David R Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. In Advances in Neural Inform. Process. Syst. The MIT Press, Granada, Spain, 1953–1961.
[8] David R Karger, Sewoong Oh, and Devavrat Shah. 2013. Efficient crowdsourcing for multi-class labeling. ACM SIGMETRICS Performance Evaluation Review 41, 1 (2013), 81–92.
[9] David R Karger, Sewoong Oh, and Devavrat Shah. 2014. Budget-optimal task allocation for reliable crowdsourcing systems. Oper. Res. 62, 1 (2014), 1–24.
[10] A. Khetan and S. Oh. 2016. Reliable Crowdsourcing under the Generalized Dawid-Skene Model. arXiv preprint arXiv:1602.03481 (2016).
[11] Eugene L Lawler and Sergei Sarkissian. 1995. An algorithm for ‘Ulam’s Game’ and its application to error correcting codes. Inform. Process. Lett. 56, 2 (1995), 89–93.
[12] S Muthukrishnan. 1994. On optimal strategies for searching in presence of errors. In Proc. ACM-SIAM Symp. Discrete Algorithms Society for Industrial and Applied Mathematics, Virginia, USA, 680–689.
[13] Andrzej Pecu. 2002. Searching games with errors—fifty years of coping with liars. Theoretical Comput. Sci. 71, 302 (2002), 71–109.
[14] Joel Ross, Lilly Irani, M Silberman, Andrew Zaldivar, and Bill Tomlinson. 2010. Who are the crowdsworkers? Shifting demographics in Mechanical Turk. In Proc. SIGCHI Conf. Hum. Factors Comput. Syst. ACM, Georgia, USA, 2863–2872.
[15] Rion Snow; Brendan O’Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In Proc. Conf. Empir. Meth. Natural Lang. Process. Association for Computational Linguistics, Hawaii, USA, 254–263.
[16] Lav R Varshney. 2012. Privacy and reliability in crowdsourcing service delivery. In Annu. SRI Global Conf. IEEE, California, USA, 55–60.
[17] Aditya Vempaty, Lav R Varshney, and Pramod K Varshney. 2014. Reliable crowdsourcing for multi-class labeling using coding theory. IEEE J. Sel. Topics Signal Process. 8, 4 (Aug. 2014), 667–679.
[18] Tsz-Yi Wang, Yunghsiang S Han, Pramod K Varshney, and Po-Ning Chen. 2005. Distributed fault-tolerant classification in wireless sensor networks. IEEE J. Sel. Areas Commun. 23, 4 (2005), 724–734.
[19] Churen Yao, Po-Ning Chen, Tsz-Yi Wang, Yunghsiang S Han, and Pramod K Varshney. 2007. Performance analysis and code design for minimum Hamming distance fusion in wireless sensor networks. IEEE Trans. Inf. Theory 53, 5 (2007), 1716–1734.
[20] Dengyong Zhou, Qiang Liu, John C Platt, Christopher Meek, and Nihar B Shah. 2015. Regularized minimax conditional entropy for crowdsourcing. arXiv preprint arXiv:1503.07240 (2015).