Where To: Crowd-Aided Path Selection

Chen Jason Zhang Yongxin Tong Lei Chen
Hong Kong University of Science and Technology, Hong Kong, China
{czhangad,yxtong,leichen}@cse.ust.hk

ABSTRACT

With the widespread use of geo-positioning services (GPS), GPS-based navigation systems have become ever more of an integral part of our daily lives. GPS-based navigation systems usually suggest multiple paths for any given pair of source and target, leaving users perplexed when trying to select the best one among them, namely the problem of *best path selection*. Too many suggested paths may jeopardize the usability of the recommendation data, and decrease user satisfaction. Although existing studies have already partially relieved this problem through integrating historical traffic logs or updating traffic conditions periodically, their solutions neglect the potential contribution of human experience.

In this paper, we resort to crowdsourcing to ease the pain of the best path selection. The first step of appropriately using the crowd is to ask proper questions. For the best path selection problem, simple questions (e.g. binary voting) over compete paths cannot be directly applied to road networks due to their being too complex for crowd workers. Thus, this paper makes the first contribution by designing two types of questions, namely Routing Query (RQ) and Binary Routing Query (BRQ), to ask the crowd to decide which direction to take at each road intersection. Furthermore, we propose a series of efficient algorithms to dynamically manage the questions in order to reduce the selection hardness within a limited budget. Finally, we compare the proposed methods against two baselines, and the effectiveness and efficiency of our proposals are verified by the results from simulations and experiments on a real-world crowdsourcing platform.

1. INTRODUCTION

1.1 Alice's Dilemma

Imagine the following scenario. Alice is unfamiliar with Hong Kong, and needs to arrive at her new office before 9 am. Through a GPS-equipped system, three paths are suggested from her apartment to her office - taxi, bus, or subway. Taxi is convenient and comfortable, but quite expensive; while bus and subway are fairly affordable, but usually slow and crowded. Alice faces the same problem every weekday - 'which path should I pick today?'

Proceedings of the VLDB Endowment, Vol. 7, No. 14

Alice's dilemma represents a very practical problem - how to find the best path from a number of suggestions. Ideally, when the cost of each edge can be perfectly evaluated with a static value, then the 'best path' is simply the 'shortest path' over a deterministic graph, which has been adequately studied in many existing works. However, in many emerging applications, the cost of a path may be influenced by numerous complex factors that are very difficult to quantitatively model. For the example of Alice taking a taxi, the cost of a section of a road may be affected by the traffic conditions, speed limitations and toll charges, which are time-dependent factors and difficult to evaluate. Given this situation, many systems consider the paths preferred by humans (e.g. experienced drivers) [6, 18], and produce not just one best path, but rather a set of paths. Each suggested path has its own merits, and it would be unwise to simply ignore any of them.

However, such a setting raises another issue. Users encounter a problem termed 'Painful Options' [15] - too many alternatives may have an adverse effect on the satisfaction of users, since exercising options is usually associated with additional cost.

1.2 Candidate Paths - An Entropy-based Measurement

To study and ease the pain of selection, we first need to design a measurement to quantify the hardness of path selection. We consider the *best path* as a discrete random variable defined over the recommended set of paths, and use the *Shannon entropy* to measure the selection hardness. The characteristics of entropy are consistent with the rationale of selection hardness. First, for a fixed number of recommended paths, it is easy for users to choose when the distribution of the best path is skewed, which means the associated entropy is low; if it is close to a uniform distribution, its entropy is high. For example, let A and B be two recommended paths, with probabilities $Pr(A)$ and $Pr(B)$ of being the best choice, respectively. Then users would prefer '$Pr(A) = 0.9, Pr(B) = 0.1$' over '$Pr(A) = 0.5, Pr(B) = 0.5$'. Second, additional candidates tend to (not strictly) increase the hardness of selection, which is also a nature captured by the entropy. Following the above example, let C be another recommended path with the probability $Pr(C)$, then users are happier with '$Pr(A) = 0.5, Pr(B) = 0.5$' than '$Pr(A) = 1/3, Pr(B) = 1/3, Pr(C) = 1/3$'. Third, when there is only one option, the entropy is equal to zero.

One may wonder how to obtain the exact probability distribution for each path. In fact, a critical issue in any system that manages uncertainty is whether we have a reliable source of probabilities. Whereas obtaining reliable probabilities for our system is one of the most interesting areas for future research, there is quite a bit to build on. For a routing system integrated with different recommendation algorithms, it is possible to train and test the algorithms on a large number of queries such that each algorithm is given a probability

based on its performance statistics. In the case where the recommendation relies mainly on large amounts of historical trajectory data [6, 18], the distribution can easily be inferred by mining the frequent paths chosen by experienced drivers. For personalized applications, it is also reasonable to allow users to configure the distribution, which reflects his or her personal preferences [17]. Clearly, it would be a difficult task for an end-user to pick the best path from the suggested ones. This problem may be especially severe when the number of candidate paths is large. Inspired by the emerging concept of crowdsourcing for various intrinsic human tasks, we resort to the crowd to facilitate the selection.

1.3 Role of the Crowd

The first task to utilize the crowd is to design a proper worker interface - we need to determine which type of questions should be posted to the crowd. This is typically referred to as HIT (Human Intelligent Task) design. It is well-known that crowdsourcing works best when tasks can be broken down into very simple pieces [24]. A *complete path*, which usually contains hundreds of vertices and edges, may be too complex for a crowd. In fact, we conducted an experiment by asking the crowd to evaluate complete paths, but neither the latency nor the accuracy is satisfactory. On the other hand, asking open-ended questions is not recommended for a crowd, because it may be difficult to integrate multiple suggestions of heterogeneous semantics. As a result, we propose to ask the crowd questions regarding directions of an intersection, namely *Routing Query* (RQ). Each RQ consists of a given vertex v_{in}, a set of vertices $D = \{v_1, v_2, ..., v_{|D|}\}$, and a target t, such that $\forall v_i \in D$, v_i is any consecutive vertex of v_{in} in one of the suggested paths. Intuitively, D indicates the possible directions moving from v_{in} towards target t, and the crowd is to choose the best one among them.

As shown in the upper part of Table 1, there are four recommended paths from the source s to the target t - P_1, P_2, P_3 and P_4, each of which is associated with its probability of being the best path. Figure 1 demonstrates the graph containing them. We assume s is HKUST and t is HKU, v_1, v_2 and v_3 are Hang Hau, Choi Hung and Kowloon Bay respectively, then a routing query could be "Which direction should I go from HKUST(s) to HKU(t), Hang Hau (v_1), Choi Hung (v_2) or Kowloon Bay (v_3)?".

Suppose the answer from the crowd is v_3 since both v_1 and v_2 are usually congested. As indicated in Table 1, P_1 and P_2 go through v_1 and v_2 respectively, hereby can be ruled out. Accordingly, both P_3 and P_4 have a 50% probability of being the best path, so that selecting the path becomes less difficult for the user. In this paper, we propose algorithms that automatically select RQs and interact with the crowd. So the crowd would be performing tasks in the back-end, and users do not need to be involved with issuing tasks.

One concern with the crowd is the real-timeliness - are users willing to wait long enough for the crowd to respond? We believe the answer is quite positive. A car moving at 30 mph = 48 kmph takes 15 seconds to cover one $200m$ long city block. Even if a path decision had to be made at every intersection, we would have about 15 seconds to run the algorithm. In practice, one rarely needs to consider a turn at every intersection. Therefore the time interval spent waiting for the response from the crowd is much longer. In this paper, we address the problem of real-timeliness in Section 4.1, by exploring the parallel processing power of the crowd. In particular, we ask the most informative k questions, which different workers can pick up concurrently, cutting down the overall human processing time. As shown in our experimental results (Section 5.4), empowered by our technique ($k = 10$), the crowd can improve the precision of path selection from 25% to 70% within 10 seconds, which is less than half of the time cost without this technique (i.e. $k = 1$). Besides, there are also horizontal techniques of realtime crowdsourcing [3, 4], which have shown promise in returning results to users in 500 milliseconds.

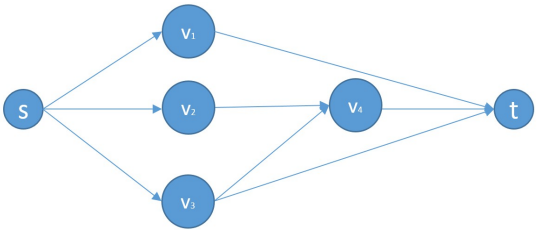


Figure 1: Candidate Paths

Candidate Paths	probability
P_1={$(s, v_1), (v_1, t)$}	.1
P_2={$(s, v_2), (v_2, v_4), (v_4, t)$}	.1
P_3={$(s, v_3), (v_3, v_4), (v_4, t)$}	.4
P_4={$(s, v_3), (v_3, t)$}	.4
Routing Query (RQ)	**pmf of A_{RQ} over D**
RQ_1: $s, D = (v_1, v_2, v_3), t$	$0.1, 0.1, 0.8$
RQ_2: $v_3, D = (v_4, t), t$	$0.5, 0.5$
Binary Routing Query (BRQ)	**pmf of A_{BRQ} (yes/no)**
BRQ_1: s, v_1, t	$0.1, 0.9$
BRQ_2: s, v_2, t	$0.1, 0.9$
BRQ_3: s, v_3, t	$0.8, 0.2$
BRQ_4: v_3, v_4, t	$0.5, 0.5$
BRQ_5: v_3, t, t	$0.5, 0.5$

Table 1: Path Distribution and Routing Queries

Another concern is that crowd workers may need to know the traffic conditions in all directions to answer an RQ. So when the degree of vertex is large, a worker may not be able to provide an accurate answer. For instance, for a given road intersection, a worker may be familiar with one or two of the directions, but not all of them. Therefore, we consider another type of questions, called *Binary Routing Query* (BRQ), as shown at the bottom of Table 1. Each BRQ contains only one direction, and asks the crowd to confirm or disconfirm, such as "From HKUST(v) to HKU(t), should I go via Hang Hau (v_1) or not?". This kind of much simpler and intuitive questions, in most circumstances, can be easily answered correctly.

Since each crowdsourcing question is usually associated with a cost, we need to design solutions to find the best path by crowdsourcing an optimal set of RQs. However, the selection is not trivial due to the following two obstacles: first, the crowd has the probability of returning incorrect answers; second, the RQs generated from a set of candidate paths are naturally correlated, so the utility of a set of RQs may be difficult to compute. To address this challenge, we design efficient and effective strategies to adaptively select and issue RQs by utilizing the submodular property of the selection hardness.

1.4 Contributions

We summarize our contributions as follows:

- First, we address the core optimization problem - how to select the most profitable questions with a given fixed budget. We derive a non-trivial property of the utility for each RQ, and propose an effective strategy to interactively select and publish RQs. Both the crowd's error and the correlation of RQs are gracefully handled in the proposed solution.

- Second, we indicate how to utilize noisy crowdsourced answers to adjust the probability distribution of the best path, and analyze the functional relations between the crowd's accuracy and a given RQ.
- Third, we consider two different extensions based on the proposed framework - 1. we study how to select multiple RQs and pose them concurrently to the crowd, in order to improve the time efficiency; 2. we investigate an easier type of questions, namely BRQ, as a replacement of RQ for high-degree vertices.
- Fourth, we conduct extensive experiments on both synthetic and real datasets. The experimental results show the superiority of our proposed methods in comparison with the baselines.

2. DEFINITIONS AND PROBLEM STATEMENT

In this section, we introduce the core definitions and related notations, then formally state the problem.

DEFINITION 2.1 (CANDIDATE PATH). *Given a source vertex s and a target vertex t over a **directed graph**, a candidate path is a sequence of edges $P = \{e_0, e_1, ..., e_{|P|}\}$, such that s is the head of e_0, t is the tail of $e_{|P|}$, and $\forall e_i, e_{i+1} \in P$ the tail of e_i is the head of e_{i+1}.*

Notation: For a given vertex v and a candidate path P, we use $\mathbf{P} \rightarrow \mathbf{v}$ and $\mathbf{P} \nrightarrow \mathbf{v}$ to denote 'P goes through v' and 'P does not go through v', respectively.

DEFINITION 2.2 (PATH SET (PS) & BEST PATH (BP)). *Given a source vertex s and a target vertex t, let PS be a set of suggested candidate paths from s to t. The best path, denoted by BP, is defined as a discrete random variable with PS as the sample space. Each candidate path $P \in PS$ has a probability $Pr(P)$ being the best path, and we have $\sum_{P \in PS} Pr(P) = 1$.*

Example: *P_1, P_2, P_3 and P_4 are candidate paths from s to t in Table 1. We have $P_1 \rightarrow v_1$ and $P_1 \nrightarrow v_2$, indicating that 'P_1 goes through v_1' and 'P_1 does not go through v_2', respectively. Moreover, $PS = \{P_1, P_2, P_3, P_4\}$ is the path set from s to t. We also have that the probability of P_1 being the best path is $Pr(P_1) = 0.1$.*

DEFINITION 2.3 (ROUTING QUERY (RQ)). *Given a Path Set PS with the source vertex s and the target vertex t, a Routing Query RQ is defined as a triple $< v_{in}, D, t >$, where*
*(1) v_{in} denotes the start vertex, indicating a particular **in**tersection;*
(2) t is the target vertex of the given PS;
(3) $D = \{v_1, ..., v_{|D|}\}$ denotes the set of all direct successors of v_{in} in PS, which indicates possible directions of moving from v_{in} to t, and $|D| \geq 2$.

From the perspective of a crowd worker, an RQ is a question that takes the form "From v_{in} to t, which direction should I go, v_0, v_1,..., or $v_{|D|}$".

Notation: Given a Path Set PS, we use U_{RQ} to denote the set of all the RQs.

Example: *In the middle part of Table 1, RQ_1 and RQ_2 indicate two RQs. In particular, RQ_2 is the question 'from source v_3 to target t, which direction should I go, v_4 or t'. Please note that the start vertex (i.e. v_{in}) of an RQ may not be the source vertex (i.e. s) of the path set, such as RQ_2. Besides, we have $U_{RQ} = \{RQ_1, RQ_2\}$.*

Notation	Meaning
P or P_i	a candidate path
$Pr(P)$ or $Pr(P_i)$	the probability of P (P_i) being the best path
PS	Path Set: the set of all candidate paths for a pair of source and target
$P \rightarrow v$	P goes through vertex v
$P \nrightarrow v$	P does not go through vertex v
$RQ :=< v_{in}, D, t >$	a Routing Query, with start vertex v_{in}, target t, and a set of directions D
A_{RQ}	the correct answer to RQ
U_{RQ}	the set of all RQs for a given PS
S_k	a subset of U_{RQ} containing k RQs
A_{S_k}	the set of k answers to S_k
BRQ	a Binary Routing Query
A_{BRQ}	the correct answer to BRQ
BP	the best path for a given PS
$H(BP)$	the selection hardness among candidate paths
ΔH_{RQ}	the expected reduction of selection hardness by asking the crowd RQ
ΔH_{S_k}	the expected reduction of selection hardness by asking the crowd all the RQs in S_k
ϵ	the error rate of a crowd worker
$X \perp Y \mid Z$	X is independent of Y given Z, where X, Y, Z are random variables

Table 2: Summary of Notations

DEFINITION 2.4 (SELECTION HARDNESS). *Given a path set $PS = \{P_1, P_2, ..., P_{|PS|}\}$, the hardness of selecting the best path BP, denoted by $H(BP)$, is defined as Shannon entropy of BP, that is,*

$$H(BP) = -\sum_{P \in PS} Pr(P) \log(Pr(P))$$

Remark: The essential purpose of collecting information from the crowd, is to make it easier to select the best path from the given candidates. Therefore, we need to quantitatively evaluate *how difficult it is to select the BP from a given PS* . Since BP can be seen as a discrete random variable with sample space PS, we use the *Shannon entropy* to measure the hardness of selecting BP from PS. As the reasons discussed in Section 1, *Shannon entropy* well captures the rationale of selection hardness. Besides, it is a non-parametric measurement that does not require any assumptions about external factors. This enables us to have a fairly simple but useful model.

DEFINITION 2.5 (PROBLEM DEFINITION). *Assume we are given a path set PS and a budget B of the number of RQs. Without exceeding the budget, we aim to design strategies to crowdsource RQs in order to maximally reduce the selection hardness $H(BP)$.*

3. RQ-BASED METHOD

In this section, we present a complete solution to select and crowdsource RQs in order to reduce the selection hardness. First, we use the expected reduction of selection hardness as the metric to evaluate RQs, and derive necessary formulas to enable the computation. Second, we study how to efficiently select the best RQ. Third, we present how to utilize conflicting crowdsourced answers. Lastly, we put these together to develop a framework of the RQ-based method, which reduces the selection hardness using a sequence of RQs.

3.1 RQ Selection Metric

In order to design an effective strategy for selecting RQs, it is essential to define a metric to estimate the utility of an RQ before it is answered. Since the final objective is to reduce the selection hardness, we use the *probabilistic expectation* of selection hardness conditioned on individual RQs as the metric.

For an arbitrary $RQ :=< v_{in}, D, t >$, let A_{RQ} be its correct answer. Probabilistically, A_{RQ} is a discrete random variable with sample space D. Therefore, the expectation of selection hardness after receiving A_{RQ}, denoted as $\mathbb{E}H(BP|A_{RQ})$, is

$$\begin{aligned}&\mathbb{E}H(BP|A_{RQ})\\ &= \sum_{v_i \in D} Pr(A_{RQ} = v_i) H(BP = P|A_{RQ} = v_i)\\ &= \sum_{v_i \in D} Pr(A_{RQ} = v_i) \sum_{P_j \in PS} [Pr(BP = P_j|A_{RQ} = v_i)\\ &\qquad \log(Pr(BP = P_j|A_{RQ} = v_i))]\end{aligned} \tag{1}$$

There are two parameters used in Equation 1 : $Pr(A_{RQ} = v_i)$ (i.e. the probability that v_i is the correct answer to RQ) and $Pr(BP = P_j|A_{RQ} = v_i)$ (i.e. the probability that P_j is the best path, given that v_i is the correct answer to RQ). Now we present the formulas to compute these two parameters with Lemma 3.1 and Lemma 3.2.

LEMMA 3.1 (COMPUTATION OF $\mathbf{Pr(A_{RQ} = v_i)}$). *For a given path set PS, let A_{RQ} be the correct answer to $RQ :=< v_{in}, D, t >$, $v_i \in D$ and $e := (v_{in}, v_i)$, we have*

$$Pr(A_{RQ} = v_i) = \frac{\sum_{P \in PS \wedge e \in P} Pr(P)}{\sum_{P' \in PS \wedge P' \to v_{in}} Pr(P')} \tag{2}$$

PROOF. *Please see appendix.* □

Equation 2 computes the probability of A_{RQ} taking each element of D, hereby we have the probability mass function (pmf) [9] of A_{RQ}.

Example: *In the example of Table 1, the probability mass functions of the answers for RQ_1 and RQ_2 are demonstrated. For RQ_2, we have $Pr(A_{RQ_2} = v_4) = Pr(P_4)/(Pr(P_3)+Pr(P_4)) = 0.4/(0.4 + 0.4) = 0.5$.*

LEMMA 3.2 (COMPUTATION OF $\mathbf{Pr(BP = P_j|A_{RQ} = v_i)}$). *For a given path set PS, let A_{RQ} be the answer to $RQ :=< v_{in}, D, t >$, $v_i \in D$, and $e := (v_{in}, v_i)$, we have*

$$Pr(BP = P_j|A_{RQ} = v_i) = \begin{cases} Pr(P_j) & P_j \nrightarrow v_{in} \\ \sum_{P \in PS \wedge P \to v_{in}} Pr(P) & e \in P_j \\ 0 & otherwise \end{cases} \tag{3}$$

PROOF. *Please see appendix.* □

The result of Lemma 3.2 is intuitive: when a candidate path P_j does not go through v_{in}, its probability remains unchanged; when P_j goes through $e := (v_{in}, v_i)$, P_j is consistent with the direction of A_{RQ}, then $Pr(P_i)$ is increased to the probability of $BP \to v_{in}$; otherwise, P_j contradicts the direction of A_{RQ}, and hence has a probability of zero.

Example: *In the example in Table 1, we have 1) $Pr(BP = P_1|A_{RQ_2} = v_4) = Pr(P_1) = 0.1$ since $P_1 \nrightarrow v_3$; 2) $Pr(BP = P_3|A_{RQ_2} = v_4) = Pr(BP \to v_3) = Pr(P_3) + Pr(P_4) = 0.8$ since $e := (v_3, v_4) \in P_3$; 3) $Pr(BP = P_4|A_{RQ_2} = v_4) = 0$ since $P_4 \to v_3$ and $e \notin P_4$, which indicates that the direction of P_4 contradicts $A_{RQ_2} = v_4$.*

Finally, by substituting Equations 2 and 3 into Equation 1, we can compute the expectation of selection hardness w.r.t each RQ.

3.2 Choosing the best RQ

A naive approach of selecting the best RQ is to traverse all the RQs. However, the computation regarding to each RQ requires accessing all the candidate paths in PS, so the computational cost will be high when the number of candidate paths is large. Fortunately, we find that the expected reduction of selection hardness for $RQ :=< v_{in}, D, t >$ is only related to the paths going through v_{in}. We conclude this discovery with the following theorem.

THEOREM 3.3. *For a given path set PS and a given $RQ :=< v_{in}, D, t >$, let ΔH_{RQ} be the expected reduction of selection hardness by asking RQ to the crowd. We have that ΔH_{RQ} is equivalent to 'the entropy of A_{RQ}' multiplying 'the probability of $BP \to v_{in}$', i.e.*

$$\begin{aligned}&\Delta H_{RQ} = H(BP) - \mathbb{E}H(BP|A_{RQ})\\ &= -(\sum_{P \to v_{in}} Pr(P)) \sum_{v_i \in D} Pr(A_{RQ} = v_i) \log Pr(A_{RQ} = v_i)\end{aligned} \tag{4}$$

PROOF. Please see appendix. □

Theorem 3.3 reflects two factors influencing the utility of an RQ - 'the entropy of the RQ' and 'the probability of the best path going through v_{in}'. Intuitively, the former indicates the amount of information gained by asking this question, so the higher the entropy is, the more important the question is; the latter indicates the structural position of the question, representing how useful the information gain is for determining the best path. It is worth noticing that, the common practice 'asking the most uncertain question' does NOT apply in our problem, as shown in the following example.

Example: *In the example of Table 1, we compute ΔH_{RQ_1} and ΔH_{RQ_2} as follows. For ΔH_{RQ_1}, since all the paths go through s, we have $\sum_{P \to s} Pr(P) = 1$, and $\Delta H_{RQ_1} = -0.1 * \log(0.1) - 0.1*\log(0.1) - 0.8*\log(0.8) = 0.28$; for ΔH_{RQ_2}, $\sum_{P \to s} Pr(P) = Pr(P_3) + Pr(P_4) = 0.8$ since P_3 and P_4 go through v_4, so $\Delta H_{RQ_2} = 0.8 * (-0.5 * \log(0.5) - 0.5 * \log(0.5)) = 0.24$. Therefore, we should choose RQ_2 over RQ_1, although the entropy of RQ_1 (0.3) is larger than RQ_2 (0.28).*

3.3 Utilization of Crowdsourced Answers

The essential objective of crowdsourcing is to use the answers to adjust the probability distribution of the best path. However, crowdsourced answers may have mistakes or be subjective. As a result, different workers may return conflicting answers for the same question. To handle this issue, we must allow each crowdsourced answer to be wrong with a probability. This probability can be estimated by the error rate of the worker. For an $RQ :=< v_{in}, D, t >$, let v_C be the result returned by a crowd worker with error rate ϵ.

Now we present how to use crowdsourced answers to adjust the probability of each candidate path P_i, that is to derive the formula to compute $Pr(BP = P_i|v_C \ returned \ by \ the \ crowd)$, as shown in Lemma 3.4.

LEMMA 3.4. *For a given path set PS and an $RQ :=< v_{in}, D, t >$, let v_C be the result returned by a crowd worker with error rate ϵ,*

then we have

$$Pr(BP = P_i|v_C \text{ returned by the crowd}) =$$

$$\begin{cases} Pr(P_i) & P_i \nrightarrow v_{in} \\ \dfrac{Pr(P_i)(1-\epsilon)}{Pr(A_{RQ}=v_C)(1-\epsilon)+(1-Pr(A_{RQ}=v_C))\epsilon} & (v_{in}, v_C) \in P_i \\ \dfrac{Pr(P_i)\epsilon}{Pr(A_{RQ}=v_C)(1-\epsilon)+(1-Pr(A_{RQ}=v_C))\epsilon} & otherwise \end{cases} \tag{5}$$

PROOF. *Please see appendix.* □

Actually, by considering P_i as a Bernoulli random variable, $Pr(P_i|v_C \text{ returned by the crowd})$ is the probability of $BP = P_i$ conditioning on the event "v_C returned by the crowd". Therefore, when more answers are received, the probability of $BP = P_i$ would be recursively adjusted by Equation 5, conditioning on each received answer and the error rate of the corresponding worker. Please note that different workers may have different error rates. Furthermore, after the probabilities of the candidate paths are adjusted by one answer, the probability distribution of each A_{RQ} is also updated by recomputing Equation 2. Thus, when the next answer is received, the adjustment is conducted with the updated probability of each P_i.

It is easy to perform the algebraic manipulations to show that, for any two answers v_C and v'_C, we have

$$\begin{aligned} &Pr(BP = P_i|v_C \text{ returned by the crowd}, \\ &\text{and then } v'_C \text{ returned by the crowd}) \\ =&Pr(BP = P_i|v'_C \text{ returned by the crowd}, \\ &\text{and then } v_C \text{ returned by the crowd}) \\ &= Pr(BP = P_i|v_C \text{ and } v'_C \text{ are returned by the crowd}) \end{aligned} \tag{6}$$

$$\sum_{P_i \in PS} Pr(BP = P_i|v_C) = 1 \tag{7}$$

These two equations resolve three issues of concern. The first is whether the sequence of answers received from the crowd affects the final result. Equation 6 indicates that, given two crowdsourced answers, the final result of PS is independent of the sequence of the answers being utilized. In other words, the final result of P_i is the probability of $BP = P_i$ conditioning on the event "both answers are received". The second issue is how to resolve the case when the same question is answered differently by multiple workers. Particularly, in Equation 6, v'_C and v_C may be conflicting answers for the same RQ from two workers. In this case, by recursively executing Equation 5 twice, v_C and v'_C are gracefully aggregated. Third, after the utilization of crowdsourced answers, whether the sum of probabilities of all candidate paths is always one. This is verified to be true as shown in Equation 7. We show how to use a crowdsourced answer with the following example.

Example: *In the example of Table 1, we have the current selection hardness* $H(BP) = -0.1 * \log(0.1) - 0.1 * \log(0.1) - 0.4 * \log(0.4) - 0.4 * \log(0.4) = 0.52$. *Now assume a crowd worker returns 'v_4' as the answer to RQ_2 with error rate* 0.2. *We apply Equation 5 on all the candidate paths as follows. Since P_1 and P_2 do not go through v_3, their probabilities remain unchanged. Besides,* $Pr(P_3|A_{RQ} = v_4) = 0.4 * (1 - 0.2)/(0.5 * (1 - 0.2) + (1 - 0.5) * 0.2) = 0.64$ *and similarly* $Pr(P_4|A_{RQ} = v_4) = 0.4 * 0.2/(0.5 * 0.2 + (1 - 0.5) * (1 - 0.2)) = 0.16$. *So the probabilities of P_1,P_2,P_3 and P_4 are* 0.1,0.1,0.64 *and* 0.16, *respectively. As a consequence, the selection hardness is reduced to* $H(BP|v_4 \text{ returned by the crowd}) = -0.1 * \log(0.1) - 0.1 * \log(0.1) - 0.64 * \log(0.64) - 0.16 * \log(0.16) = 0.45$.

Input: A path set PS, U_{RQ}, a total budget B

1 **while** $B \neq 0$ **do**
2 **for** *each* $RQ_i \in U_{RQ}$ **do**
3 calculate ΔH_{RQ_i} via Theorem 3.3;
  end
4 $RQ_{max} \leftarrow \text{argmax}_{RQ_i \in U_{RQ}} \Delta H_{RQ_i}$;
5 Ask RQ_{max} to crowd and receive the corresponding answer v_C;
6 **for** *each* $P_j \in PS$ **do**
7 $Pr(P_j) \leftarrow Pr(BP = P_j|v_C \text{ returned by the crowd})$ via Formula 5;
  end
8 $B \leftarrow B - 1$;
end

Algorithm 1: The Framework of RQ-based Method

3.4 The Framework of RQ-based Method

In this subsection, we provide the complete framework of the RQ-based method. Algorithm 1 illustrates this framework, which consists of two iterative phases:

- *Choosing the best RQ* - select the best RQ based on the current probabilities of candidate paths, and post it to the crowd (lines 1-5);
- *Utilization of Conflicting Crowdsourced Answers* - adjust the probabilities of all candidate paths according to the crowdsourced answers. (lines 6-8)

In Algorithm 1, these two phases are iteratively performed B times given the budget constraint. In each iteration, we first calculate the expected reduction of selection hardness, ΔH_{RQ}, for each RQ via Theorem 3.3. Then, the one with maximum ΔH_{RQ} is selected and published to the crowd. Second, we receive the answer v_C, and adjust the probabilities of all candidate paths through Formula 5, hereby reducing the selection hardness.

4. EXTENSIONS OF RQ-BASED METHOD

In this section, we propose two extensions of the RQ-Based method. First, to reduce the latency and improve the realtime performance, we extend the RQ-based method to the scenario of asking multiple questions concurrently. Second, we consider a different type of questions to ask the crowd, namely Binary Routing Query (BRQ), which is easier and more user-friendly.

4.1 Extension 1: Asking Multiple Questions

Up until now, we have been asking the crowd one question at a time. However, in order to reduce latency, we may also ask multiple questions at a time in a crowdsourcing environment. According to the strategy of choosing the best RQ in Section 3.2, a straightforward heuristic solution is to select the top-k questions which have the highest expected reduction of selection hardness. However, such a solution neglects the fact that there is a correlation existing between RQs. Thus, it is non-trivial to select the best combination of k questions. In this subsection, we first formulate the problem of selecting the best combination of k questions, and then propose an effective sampling-based solution.

4.1.1 Formulations and Notations

Before presenting our solution, we introduce several new notations and formulate the problem of selecting the best combination of k questions. Given the set U_{RQ} including all the RQs for a path set, let S_k be a size-k subset of U_{RQ}, that is $S_k = \{RQ_1, RQ_2, \cdots, RQ_k\} \subseteq U_{RQ}$. The set of answers of S_k is denoted as $A_{S_k} = \{A_{RQ_1}, A_{RQ_2}, \cdots, A_{RQ_k}\}$, which follows the

joint distribution of the random variables $A_{RQ_1}, A_{RQ_2}, \cdots, A_{RQ_k}$. Please note that $A_{RQ_1}, A_{RQ_2}, \cdots, A_{RQ_k}$ are *correlated* in general.

According to the above notations, the problem of selecting the best combination of k questions is to select k RQs from U_{RQ}, such that the expected selection hardness is maximally reduced. Let ΔH_{S_k} denote the expected reduction of the selection hardness, i.e.

$$\Delta H_{S_k} = H(BP) - \mathbb{E}H(BP|A_{S_k}) \tag{8}$$

Therefore, we have the optimization problem

$$\underset{S_k \subseteq U_{RQ}, |S_k| \leq k}{\operatorname{argmax}} \Delta H_{S_k}$$

where $H(\cdot)$ indicates the selection hardness in Definition 2.4, modeled in the form of the Shannon entropy. From the perspective of information theory [7], ΔH_{S_k} can be considered as the *mutual information* between BP and A_{S_k}.

4.1.2 Sampling-based Method

Based on the aforementioned formulation of selecting the best combination of k questions, there are $\binom{|U_{RQ}|}{k}$ possible size-k combinations. As a result, the size of the sample space of A_{S_k} is exponential of k. In other words, finding the optimal set S_k of RQs is very expensive when k is large. To address this challenge, we utilize the fact that ΔH_{S_k} can be considered as the mutual information, which is naturally a submodular function [2].

Submodularity is an intuitive property of diminishing returns, stating that adding an element to a smaller set helps more than adding it to a larger set. In our problem, the submodularity indicates that the utility of an RQ may be reduced when it is asked in conjunction with other RQs. Explicitly, if we consider ΔH_{S_k} as a set function with domain U_{RQ}, the selection problem is how to maximize ΔH_{S_k} with a budget constraint k. In general, maximizing the a submodular function is NP-hard, but fortunately approximable [2]. In particular, it is indicated that the problem of selecting a k-element subset maximizing a submodular function can be approximated with a performance guarantee of $(1 - 1/e) = 63\%$, by iteratively selecting the local optimal element given the ones selected so far.

As a result, the challenge of choosing the best combination of k questions is transformed to how to efficiently discover the local maximum in each iteration. According to the aforementioned formulation of ΔH_{S_k}, we need to compute $H(BP)$ and $H(BP|A_{S_k})$ efficiently. However, the mutual information is calculated over $\binom{|U_{RQ}|}{k}$ possible combinations, rendering the enumeration impractical. Therefore, we adopt a sampling-based method to estimate the mutual information. Considering ΔH_{S_k} as the mutual information of BP and A_{S_k}, we rewrite Equation 8 as

$$\Delta H_{S_k} = \sum_{BP, A_{S_k}} Pr(BP, A_{S_k}) \log \frac{Pr(BP, A_{S_k})}{Pr(BP) \cdot Pr(A_{S_k})} \tag{9}$$

Intuitively, we use the frequencies of samples to estimate the probabilities. Let function $freq(\cdot)$ denote the count (i.e. number of occurrences) of a specific value in the sampling process. We define the **estimator** of ΔH_{S_k}, denoted by $\Delta \hat{H}_{S_k}$, as follows:

$$\begin{aligned} \Delta H_{S_k} &\approx \Delta \hat{H}_{S_k} \\ &= \sum_{BP, A_{S_k}} freq(BP, A_{S_k}) \log \frac{freq(BP, A_{S_k})}{freq(BP) \cdot freq(A_{S_k})} \end{aligned} \tag{10}$$

where $freq(BP) = \sum_{A_{S_k}} freq(BP, A_{S_k})$ and $freq(BP) = \sum_{BP} freq(BP, A_{S_k})$.

Input: A path set PS, U_{RQ}, the number of RQs k for each round, a total budget B

1 **while** $B \neq 0$ **do**
2 $i = 0$; $paid = 0$; $S_k = \emptyset$;
3 **for** $i <= k - 1$ **do**
4 **for** *each* $RQ_i \in U_{RQ}$ **do**
5 calculate ΔH_{S_k} via Formula 10 ;
  end
6 $RQ_{max} \leftarrow \operatorname{argmax}_{RQ_i \in U_{RQ}} \Delta H_{S_k}$;
7 $S_k = S_k \cup RQ_{max}$;
  end
8 Ask RQs in S_k to crowd and receive the corresponding answers;
9 **for** *each crowdsourced answer* v_C **do**
10 **for** *each* $P_j \in PS$ **do**
11 $Pr(P_j) \leftarrow Pr(BP = P_j | v_C)$ via Formula 5;
  end
12 $paid \leftarrow paid + 1$;
  end
13 $B \leftarrow B - paid$;
14 **if** $B < k$ **then**
15 $k = B$;
  end
end

Algorithm 2: Asking Multiple Questions: k-selection

Based on the aforementioned estimator, we can obtain an unbiased estimation of mutual information through the sampling method.

To sum up, the complete solution for asking multiple questions is shown in Algorithm 2, namely *k-selection*. The basic idea is to select the best size-k subset S_k by means of the aforementioned sampling method, then publish k RQs at each round, and adjust the probabilities of the candidate paths according to the answers collected from the crowd.

In Algorithm 2, we first select the k-size subset S_k in a greedy fashion (lines 2-7). Then the questions in S_k are published to the crowd (line 8). Please note that the RQs are either answered or expired, and the expired ones are free of cost. For each crowdsourced answer, we adjust the probabilities of candidate paths by recursively using Formula 5 (lines 9-12).

4.2 Extension 2: Different Question Type

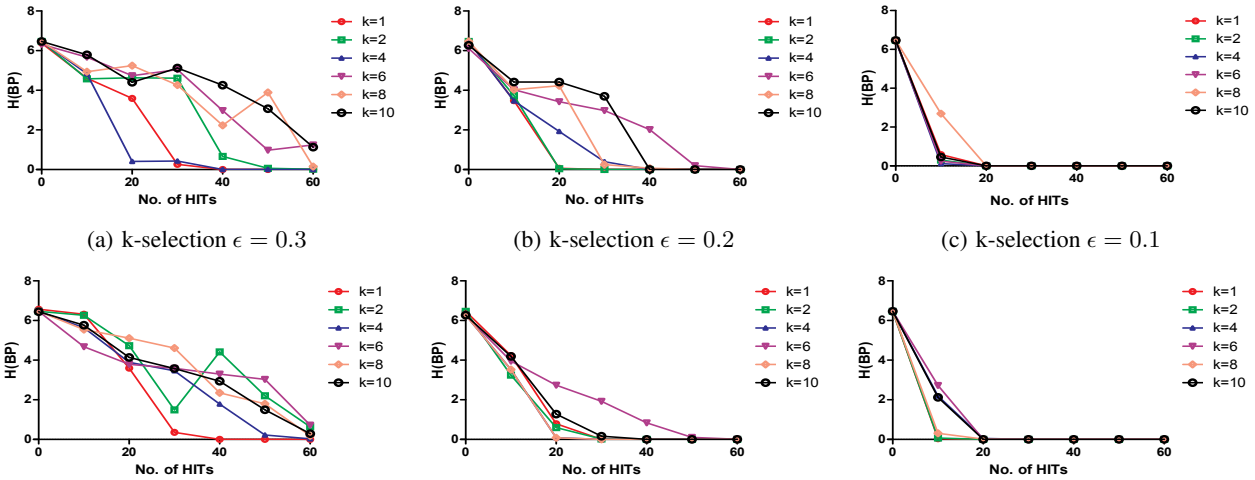
In the RQ-based method, each RQ is a multiple-choice question. For a high-degree vertex, the RQ would consist of too many options for a crowd worker. As suggested in [23, 26], crowds are good at tasks broken down into small pieces (often with a YES/NO answer). Motivated by this, we consider an extension that uses an easier type of questions, namely BRQ, as defined by the following Definition 4.1.

DEFINITION 4.1 (BINARY ROUTING QUERY (BRQ)). *For a given* $RQ :=< v_{in}, D = \{v_0, ..., v_{|D|}\}, t >$*, a Binary Routing Query* BRQ *is triple* $< v_{in}, v_d, t >$*, where* $v_d \in D$.

From the perspective of a crowd worker, a BRQ is a question that takes the form "From v_{in} to t, should I go in the direction of v_d?"

Let A_{BRQ} be the correct answer to BRQ, then A_{BRQ} can be probabilistically considered as a Bernoulli random variable since the answer of BRQ is either yes or no. The bottom part of Table 1 lists all the $BRQs$ for the PS. It is not hard to see that each RQ can be decomposed into $|D|$ distinct $BRQs$. As a new type of questions, BRQs can be easily embedded in Algorithms 1 and 2. Analogous to the RQ-based method, we also focus on studying how to select the best BRQ in this extension.

Finding the best BRQ: As shown in Theorem 3.3, the utility of an RQ is determined by its information gain and topological position. For $BRQs$, we reach a similar result, as shown in the following theorem.



(a) k-selection $\epsilon = 0.3$ (b) k-selection $\epsilon = 0.2$ (c) k-selection $\epsilon = 0.1$

(d) baseline $\epsilon = 0.3$ (e) baseline $\epsilon = 0.2$ (f) baseline $\epsilon = 0.1$

Figure 2: Performance VS k

THEOREM 4.1. *For a given path set PS and a given $BRQ :=< v_{in}, v_d, t >$, let ΔH_{BRQ} be the expected reduction of selection hardness by asking the BRQ to the crowd, we find that ΔH_{BRQ} is equivalent to 'the probability of $BP \rightarrow v_{in}$' multiplying 'the entropy of the A_{BRQ}', that is*

$$\Delta H_{BRQ} = H(BP) - \mathbb{E}H(BP|A_{BRQ})$$
$$= -(\sum_{P \rightarrow v_{in}} Pr(P))[Pr(A_{BRQ} = v_d) \log Pr(A_{BRQ} = v_d)$$
$$+ (1 - Pr(A_{BRQ} = v_d)) \log (1 - Pr(A_{BRQ} = v_d))]$$

PROOF. *Please see appendix.* □

5. EXPERIMENTAL EVALUATION

In this section, we report on the experimental study to validate the effectiveness and efficiency of our proposals. First, we use synthetic data and a simulated crowd to explore the effect of wide ranges of parameter values. Second, we conduct an experiment with real-world datasets on Amazon Mechanical Turk, which is a public crowdsourcing platform.

In the experiments, we compare the following three algorithms.

1. k-selection algorithm(**k-selection**), our proposed methods in Section 3 & 4. When $k = 1$, we select one best RQ following the algorithm introduced in Section 3; when $k > 1$, we applied the sampling-based algorithm described in Section 4.1.

2. baseline algorithm (**baseline**) - at each round, the RQs with top-k highest entropies are selected.

3. naive algorithm (**rand**) - each RQ is chosen randomly.

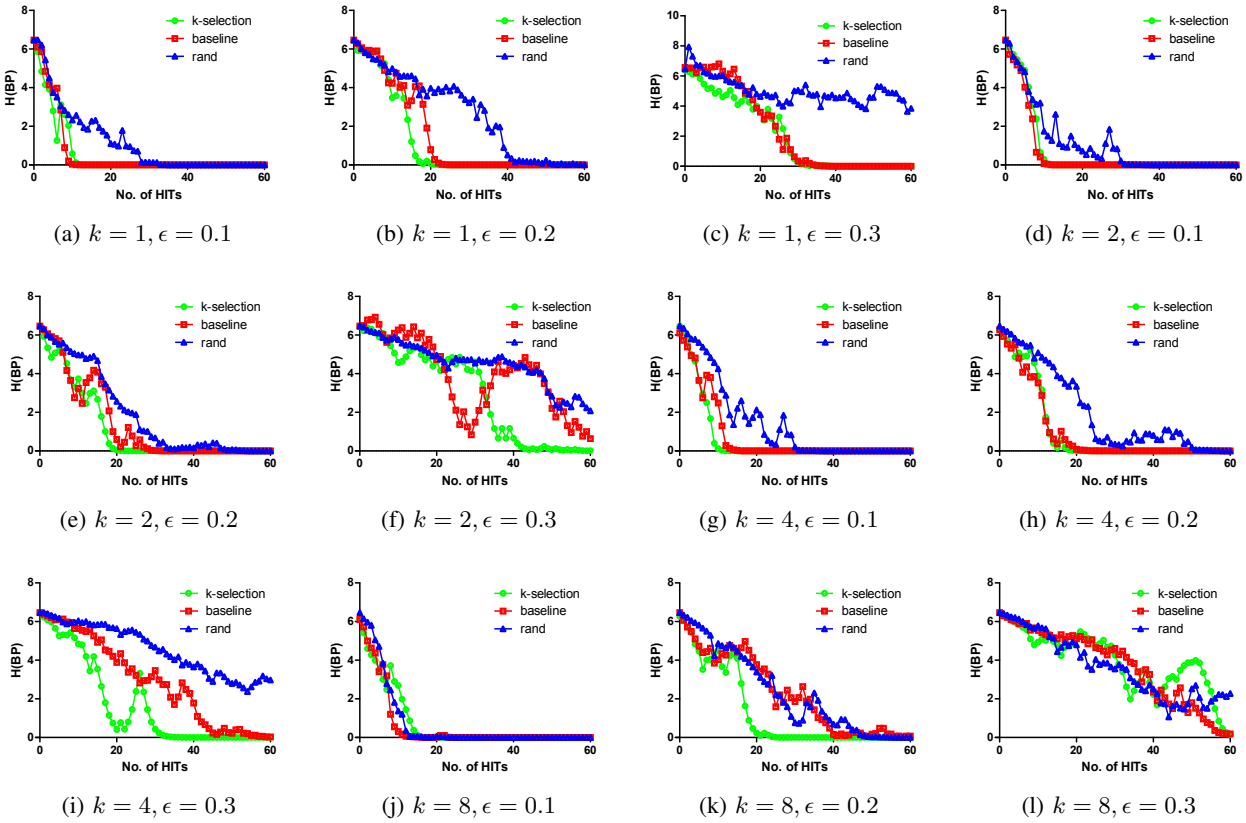
5.1 Simulation on Synthetic Data

In order to explore the effect of wide ranges of parameter values, i.e. ϵ and k, we conduct a series of experiments on a synthetic dataset with a simulated crowd. In particular, the crowd answers each RQ independently, and each RQ has a probability ϵ to be incorrectly answered. Each PS in the synthetic dataset contains 50 candidate paths, one of them is labeled as the best. We set the total budget of RQs $B = 60$ for a PS, and at each iteration we select and publish k RQs, so there are totally $\lceil 60/k \rceil$ iterations (there are maybe less than k RQs in the last iteration).

Effect of k: First, we test *k-selection* and *baseline* with varying k in Figure 2. Recall that k is the number of questions issued in each iteration. So, for a fixed budget, a larger k results in smaller a number of iterations. When $k = 1$, we essentially sort the list of all the RQs by their utilities (with Theorem 3.3), and choose the very best one at each iteration - all the RQs are at the top of the list when they are chosen. When $k > 1$, some RQs are not at the top of the list when they are chosen, since at each iteration, we choose k good RQs rather than the very best one. Therefore, we expect that the selection hardness is reduced more rapidly with a smaller k. This expectation is basically consistent with the experimental results, as shown in Figure 2. In particular, k is set to $1, 2, 4, 6, 8$ and 10 in the experiment, and we can see that a smaller k *tends* to be more effective on reducing the selection hardness. Although this tendency is very clear, but smaller k does not lead to absolutely more rapid reduction, due to the randomness of the sampling-based algorithm as well as the possibly wrong answers. The advantage of larger k is the time cost due to the less number of iterations, which is to be discussed in Section 5.4.

Effect of ϵ: Second, we present the results on varying the error rate ϵ. For all three competing algorithms, we find that the lower ϵ is, the faster the convergence is. This finding is supported by observing Figures 3(a)-3(c), Figures 3(d)-3(f), Figures 3(g)-3(i), and Figures 3(j)-3(l). In other words, we would need less RQs for a crowd with higher accuracy. Moreover, as shown in Figures 3(a)-3(l), regardless of the setting of k, ϵ and the option of the algorithm, the selection hardness $H(BP)$ is gradually decreased with the reception of crowdsourced answers. This suggests that the crowd is conducive for path selection in general.

Comparison of k-selection, baseline, and rand: Third, we compare the performances of *k-selection*, *baseline* and *rand* in terms of the reduction of selection hardness. As indicated in Figure 3, *baseline* and *k-selection* significantly outperform *rand* in all the cases. It is worth noting that *k-selection* and *baseline* sometimes have similar performance when $k = 1$ or when the error rate is small. This is because the initial probabilities of edges are usually the same, which causes that both algorithms always select the RQ with a probability closest to a uniform distribution, and achieve their respective local optima as a consequence. When $k > 1$ and the error rate is higher, *k-selection* performs better than *baseline* in most cases, which is supported by Figures 3(e)-3(g),3(i),3(k),3(l). This is because our *k-selection* considers the inherent correlation



(a) $k = 1, \epsilon = 0.1$ (b) $k = 1, \epsilon = 0.2$ (c) $k = 1, \epsilon = 0.3$ (d) $k = 2, \epsilon = 0.1$

(e) $k = 2, \epsilon = 0.2$ (f) $k = 2, \epsilon = 0.3$ (g) $k = 4, \epsilon = 0.1$ (h) $k = 4, \epsilon = 0.2$

(i) $k = 4, \epsilon = 0.3$ (j) $k = 8, \epsilon = 0.1$ (k) $k = 8, \epsilon = 0.2$ (l) $k = 8, \epsilon = 0.3$

Figure 3: $\epsilon = 0.1, 0.2, 0.3$ **and** $k = 1, 2, 4, 8$

among RQs, which is an intrinsic advantage comparing to *baseline*.

Another important finding is that, with the increase of the crowd error rate ϵ, the advantage of *k-selection* becomes much more notable compared to *baseline* and *rand*. This phenomenon suggests that the proposed algorithm has superior performance especially when the crowds are noisy. By adjusting k, users are able to tradeoff the time efficiency and utility of RQs - large k leads to high time-efficiency, but relatively low utility.

5.2 Testing on with Real Data

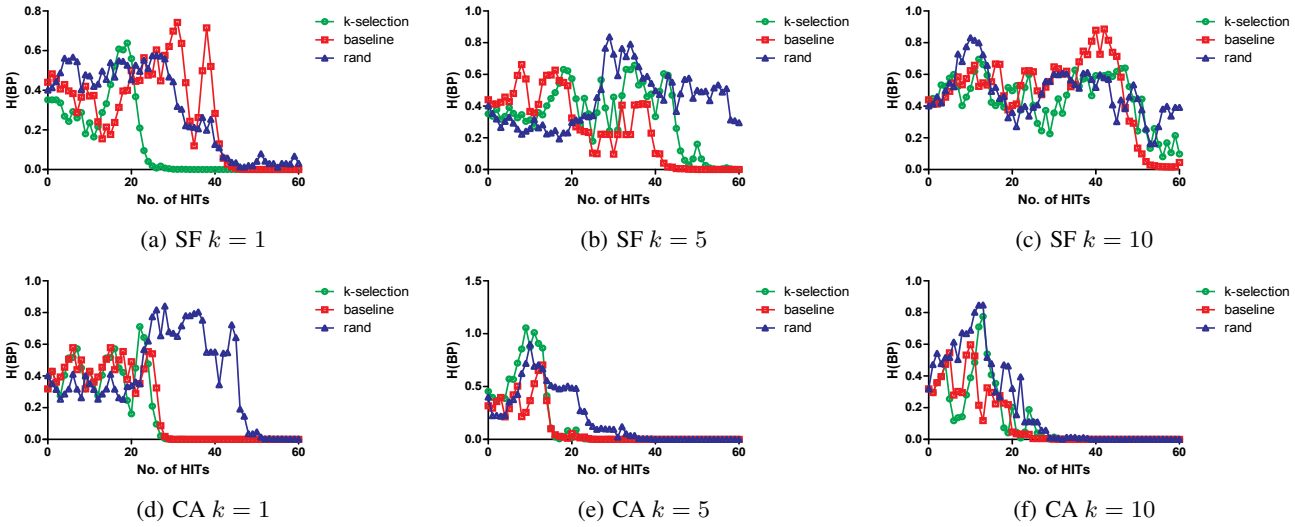
We use two real-world road-network datasets - *CA* and *SF* [29, 30]. In particular, *CA* consists of highways and main roads in California and *SF* contains detailed street networks in San Francisco. We construct 30 PSs from *SF*, and 40 PSs from *CA*. We add the random uncertainty to each edge, then, for each PS, the candidate paths and probabilities are aggregated from a number of suggested paths generated by two different recommendation algorithms [14, 32]. Each of them generates top-10 paths, and we have 3-15 distinct paths for each PS.

We tested these two real-world datasets on Amazon Mechanical Turk (AMT), which is a widely used crowdsourcing marketplace. We set the budget of RQs $B = 60$, and each RQ is awarded 0.03 USD for both datasets. We compare *k-selection*, *baseline* and *rand* with $k = 1, 5, 10$. The average performances are demonstrated in Figure 4. In terms of the reduction of selection hardness, one can see that the performance is basically consistent with the simulation - *k-selection* and *baseline* outperform *rand*. Please note a budget is set up for each PS, so users are allowed to set different budgets for different data instances. Moreover, we estimate the difficulty of each RQ with the method introduced in [12]. In particular, the difficulty of each RQ is evaluated by a Beta distribution, and too difficult RQs should be eliminated. As a result, only 3 RQs are removed through the entire experiment. This suggests that the overwhelming majority of RQs are not difficult for the crowd. Besides, we found that the workers usually have consistent answers. This suggests that our crowdsourcing model is valid.

In the experiment above, we assign random uncertainty to the road networks. Additionally, we conducted another set of experiments with different uncertainty distributions, which reflects the situation that downtown edges have lower uncertainty than rural areas. In particular, we partition the road network of CA into 20 small regions, and add uncertainty to edges from normal distributions with different variances. We simulate situations where the downtown edges have lower uncertainty (with high variance) than rural areas (with small variance). The experimental results are consistent with the ones included in this paper. Due to space limitations, we put detailed results in the technical report [33].

5.3 Effectiveness

We conducted experiments to exhibit the *goodness* of paths selected by the crowd. One thing worth noting is that the goodness of a path is not *absolute*. Nevertheless, in order to present a fairly objective evaluation, we carefully select 40 PSs (20 for CA and 20 for SF), such that each of them contains a candidate path recommended by different third-party systems, namely *Google Map* (https://maps.google.com), *Bing Map* (http://www.bing.com/maps), as well as *Yahoo Map* (https://maps.yahoo.com). The common



(a) SF $k = 1$ (b) SF $k = 5$ (c) SF $k = 10$

(d) CA $k = 1$ (e) CA $k = 5$ (f) CA $k = 10$

Figure 4: Testing on Real Data

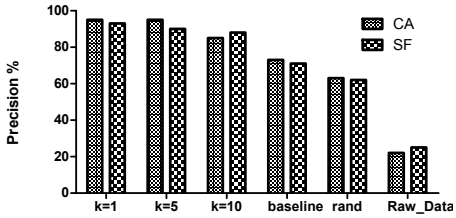
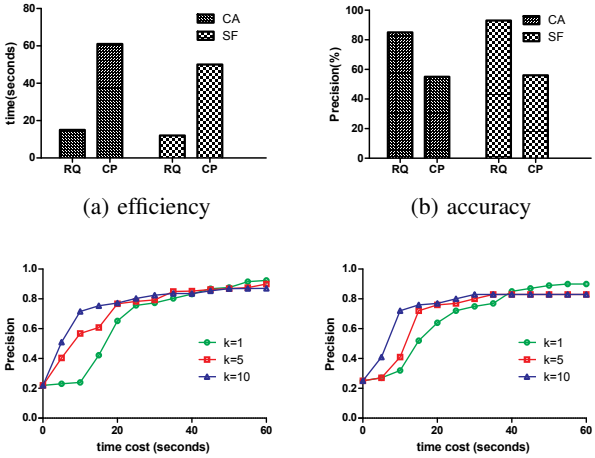


Figure 5: Precision



(a) efficiency (b) accuracy

(c) CA (d) SF

Figure 6: Efficiency and Accuracy

paths recommended by all the systems are assumed to be the ground truth [25].

For each path set PS, we run the algorithms and consider the candidate path with the *highest probability* as the output, which is compared with the ground truth. Moreover, for comparison, we use Raw_Data to indicate the candidate paths with the highest probabilities directly from the input PSs. As shown in Figure 5, the crowdsourced paths can achieve very high accuracy in both datasets. In addition, we find that *k-selection* outperforms *baseline* and *rand* in terms of precision. From Figure 5, we can also observe that Raw_Data has quite a low precision, which indicates the necessity of crowdsourcing.

5.4 Time Cost

In this subsection, we empirically examine the time-efficiency of the crowd, and the relation between it and the effectiveness. For comparison with RQ, we present an experiment asking the crowd to evaluate the ***complete paths*** (denoted by CP) - an entire candidate path is provided to the crowd, and the worker needs to confirm or disconfirm whether the given path is the best.

First, we compare the average time cost of RQ with that of CP, as shown in Figure 6(a). One can see that RQs can be finished within 10 seconds. But CP takes much longer as it is more complex for workers. In Figure 6(b), we also demonstrate the precision of CP and RQ. It is clear that RQ is more accurate than CP.

Second, we study the relationship between the time efficiency and the effectiveness of *k-selection*. Intuitively, the longer time it takes, the higher the precision is, since more workers could participate and return answers. We demonstrate the experimental results in Figures 6(c) and 6(d), in which we run *k-selection* for 60 seconds. As we can see, by setting $k = 10$, the precision achieves over 70% within 10 seconds for both datasets. Comparatively, when we set $k = 1$, it takes over 20 seconds to achieve the same precision. However, if the time constraint is 60 seconds, the precision would go up to 90%, by setting $k = 1$. In conclusion, a large k value is suggested if latency is the primary constraint, whereas a small value of k is recommended if the primary constraint is accuracy.

6. RELATED WORKS

6.1 Crowdsourcing

The recent development of crowdsourcing brings us a brand new opportunity to engage human intelligence in the process of answering queries (see [10] as a survey). Crowdsourcing provides a new

problem-solving paradigm [5,19], which has been blended into several research communities. In particular, crowdsourcing-based data management techniques have recently attracted much attention in the database and data mining communities. From a practical viewpoint, [11] proposed and developed a query processing system using microtask-based crowdsourcing to answer queries. Moreover, in [22], a declarative query model is proposed to cooperate with standard relational database operators. In addition, from the viewpoint of theoretical study, many fundamental queries have been extensively studied, including filtering [21], max [13], and so on. Besides, crowdsourcing-based solutions to many complex algorithms have also been developed, such as entity resolution [27, 28], trip planning [16] and so on.

6.2 Path Recommendation

Finding the most desirable path has been receiving tremendous research interest for decades. The most popular topic in this area is shortest path finding, which has been extensively studied for over fifty years [1]. If the weight on each edge represents travel time, shortest path finding becomes fastest path finding. Time-dependent shortest path problem [20] regards the travel time of an edge as a single-value function on the time of day. To improve routing services, new approaches [31] for fastest path finding are proposed aiming at using user-generated GPS trajectories to estimate the distribution of travel time on a given road network. In addition, [6, 18] propose efficient algorithms to discover the paths preferred by humans (e.g. experienced drivers) from GPS trajectories.

To the best of our knowledge, this is the first work studying the path selection problem with crowdsourcing. The essential objective is to make it easier for users to select the best path among a number of candidates. The recent work [25] proposes a system to leverage crowds' knowledge to improve the quality of recommended routes. Our paper distinguishes itself with [25] from the two aspects: first, we ask the crowd to identify the direction at each road intersection; second, we adjust the distribution of recommended paths, rather than identify the very best one.

7. CONCLUSION

A GPS-based navigation system usually suggests multiple paths for a pair of given source and target. Therefore, a problem struggling with users is how to select the best one among them, namely *the best path selection* problem. In this paper, we utilize crowdsourcing to ease the pain of best path selection. In particular, we design two types of questions, namely Routing Query (RQ) and Binary Routing Query (BRQ), to ask the crowd to decide the direction at each road intersection. Furthermore, we propose a series of efficient algorithms, which dynamically manage the questions in order to reduce the selection hardness with a limited budget of questions. Finally, we verified the effectiveness and efficiency of our proposed approaches through experiments with synthetic and real-world datasets.

8. ACKNOWLEDGMENT

This work is supported in part by the Hong Kong RGC Project N_HKUST637/13, National Grand Fundamental Research 973 Program of China under Grant 2012-CB316200, National Natural Science Foundation of China (NSFC) Grant No. 61328202, Microsoft Research Asia Gift Grant and Google Faculty Award 2013.

9. REFERENCES

[1] Ravindra K Ahuja, Kurt Mehlhorn, James Orlin, and Robert E Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM (JACM)*, 37(2):213–223, 1990.

[2] Carlos Guestrin Andreas Krause. A note on the budgeted maximization of submodular functions. Technical report, School of Computer Science, Carnegie Mellon University, March 2005.

[3] Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. Crowds in two seconds: enabling realtime crowd-powered interfaces. In *UIST*, pages 33–42, 2011.

[4] Michael S. Bernstein, David R. Karger, Robert C. Miller, and Joel Brandt. Analytic methods for optimizing realtime crowdsourcing. *CoRR*, abs/1204.2995, 2012.

[5] Daren C. Brabham. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence February 2008 vol. 14 no. 1 75-90*, 2008.

[6] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *ICDE*, pages 900–911, 2011.

[7] Thomas M. Cover and Joy Thomas. *Elements of Information Theory*. Wiley, 1991.

[8] A. Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(1):1–31, 1979.

[9] M.H. DeGroot and M.J. Schervish. *Probability and Statistics*. Addison-Wesley series in statistics. Addison-Wesley, 2002.

[10] AnHai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, 2011.

[11] Amber Feng, Michael J. Franklin, Donald Kossmann, Tim Kraska, Samuel Madden, Sukriti Ramesh, Andrew Wang, and Reynold Xin. Crowddb: Query processing with the vldb crowd. *PVLDB*, 4(12):1387–1390, 2011.

[12] Jinyang Gao, Xuan Liu, Beng Chin Ooi, Haixun Wang, and Gang Chen. An online cost sensitive decision-making method in crowdsourcing systems. In *SIGMOD Conference*, pages 217–228, 2013.

[13] Stephen Guo, Aditya G. Parameswaran, and Hector Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD Conference*, pages 385–396, 2012.

[14] Ming Hua and Jian Pei. Probabilistic path queries in road networks: traffic uncertainty aware path selection. In *EDBT*, pages 347–358, 2010.

[15] H. V. Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, and Cong Yu. Making database systems usable. In *SIGMOD Conference*, pages 13–24, 2007.

[16] Haim Kaplan, Ilia Lotosh, Tova Milo, and Slava Novgorodov. Answering planning queries with the crowd. *PVLDB*, 6(9):697–708, 2013.

[17] Long Liu, Jin Xu, Stephen Shaoyi Liao, and Huapin Chen. A real-time personalized route recommendation system for self-drive tourists based on vehicle to vehicle communication. *Expert Syst. Appl.*, 41(7):3409–3417, 2014.

[18] Wuman Luo, Haoyu Tan, Lei Chen, and Lionel M. Ni. Finding time period-based most frequent path in big trajectory data. In *SIGMOD Conference*, pages 713–724, 2013.

[19] T.W. Malone, R. Laubacher, and C. Dellarocas. Harnessing crowds: Mapping the genome of collective intelligence. Research Paper No. 4732-09, MIT, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, USA, February 2009. Sloan Research Paper No. 4732-09.

[20] Ariel Orda and Raphael Rom. Distributed shortest-path protocols for time-dependent networks. *Distributed Computing*, 10(1):49–62, 1996.

[21] Aditya G. Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD Conference*, pages 361–372, 2012.

[22] Aditya G. Parameswaran and Neoklis Polyzotis. Answering queries using humans, algorithms and databases. In *CIDR*, pages 160–166, 2011.

[23] Aditya G. Parameswaran, Anish Das Sarma, Hector Garcia-Molina, Neoklis Polyzotis, and Jennifer Widom. Human-assisted graph search: it's okay to ask questions. *PVLDB*, 4(5):267–278, 2011.

[24] Hyunjung Park and Jennifer Widom. Query optimization over crowdsourced data. *PVLDB*, 6(10):781–792, 2013.

[25] Han Su, Kai Zheng, Jiamin Huang, Hoyoung Jeung, Lei Chen, and Xiaofang Zhou. Crowdplanner: A crowd-based route recommendation system. In *ICDE*, 2014.

[26] Hao Su, Jia Deng, and Li Fei-Fei. Crowdsourcing annotations for visual object detection. In *AAAI Technical Report, 4th Human Computation Workshop*, 2012.

[27] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

[28] Steven Euijong Whang, Peter Lofgren, and Hector Garcia-Molina. Question selection for crowd entity resolution. *PVLDB*, 6(6):349–360, 2013.

[29] Xiaokui Xiao, Bin Yao, and Feifei Li. Optimal location queries in road network databases. In *ICDE*, pages 804–815, 2011.

[30] Man Lung Yiu, Dimitris Papadias, Nikos Mamoulis, and Yufei Tao. Reverse nearest neighbors in large graphs. *IEEE Trans. Knowl. Data Eng.*, 18(4):540–553, 2006.

[31] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *KDD*, pages 316–324, 2011.

[32] Ye Yuan, Lei Chen, and Guoren Wang. Efficiently answering probability threshold-based shortest path queries over uncertain graphs. In *DASFAA (1)*, pages 155–170, 2010.

[33] Chen Zhang, Yongxin Tong, and Lei Chen. Where to: Crowd-aided path selection. `http://ihome.ust.hk/~czhangad/whereto_tr.pdf`.

APPENDIX

Proof of Lemma 3.1

PROOF. Recall that RQ is a question asking how to move forward starting from v_{in}. Hence, $A_{RQ} = v_i$ indicates $e := (v_{in}, v_i) \in BP$, given BP goes through v_{in}. Then we have

$$Pr(A_{RQ} = v_i) = Pr(e \in BP | BP \to v_{in}) = \frac{Pr(BP \to v_{in} | e \in BP) Pr(e \in BP)}{Pr(BP \to v_{in})}$$

Please note that v_{in} is the head of edge e, so given the condition that e is on the best path (i.e. $e \in BP$), the best path must go through v_{in}, that is, $Pr(BP \to v_{in} | e \in BP) = 1$. Hence, we have

$$Pr(A_{RQ} = v_i) = \frac{Pr(e \in BP)}{Pr(BP \to v_{in})}$$

Following the Law of Total Probability [9], we have $Pr(e \in BP) = \sum_{P \in PS} Pr(e \in BP \cap BP = P) = \sum_{P \in PS \wedge e \in P} Pr(P)$ and $Pr(BP \to v_{in}) = \sum_{P' \in PS} Pr(P' \to v_{in} \cap BP = P') = \sum_{P' \in PS \wedge P' \to v_{in}} Pr(P')$. Finally, we have $Pr(A_{RQ} = v_i) = \frac{\sum_{P \in PS \wedge e \in P} Pr(P)}{\sum_{P' \in PS \wedge P' \to v_{in}} Pr(P')}$, which completes the proof. □

Proof of Lemma 3.2

PROOF. The main difficulty of deriving $Pr(BP = P_j | A_{RQ} = v_i)$ is to determine the correlation between '$BP = P_j$' and '$A_{RQ} = v_i$'. We observe that this correlation is closely related to '$BP \to v_{in}$', i.e. whether the best path goes through the start point of RQ. Therefore, we expand $Pr(BP = P_j | A_{RQ} = v_i)$ with the Law of Total Probability as follows:

$$\begin{aligned} &Pr(BP = P_j | A_{RQ} = v_i) = \\ &Pr(BP \to v_{in}) Pr(BP = P_j | A_{RQ} = v_i, BP \to v_{in}) \\ &+ (1 - Pr(BP \to v_{in})) Pr(BP = P_j | A_{RQ} = v_i, BP \nrightarrow v_{in}) \end{aligned} \tag{11}$$

where we have $Pr(BP \to v_{in}) = \sum_{P \in PS \wedge P \to v_{in}} Pr(P)$.

We derive $Pr(BP = P_j | A_{RQ} = v_i, BP \to v_{in})$ and $Pr(BP = P_j | A_{RQ} = v_i, BP \nrightarrow v_{in})$ by respectively analyzing two exclusive conditions - $BP \to v_{in}$ and $BP \nrightarrow v_{in}$.

Condition $BP \to v_{in}$: First, we analyze the situation that v_{in} is on the best path BP. For each $v_i \in D$, if edge $e := (v_{in}, v_i)$ is on the best path, then v_i must be the best direction going from v_{in} to t, i.e. the ground truth answer A_{RQ} should be v_i. Therefore, we have $e \in BP \Rightarrow A_{RQ} = v_i$.

Similarly, if $A_{RQ} = v_i$ and $BP \to v_{in}$, we can ensure that $e \in BP$. So $(A_{RQ} = v_i \wedge BP \to v_{in}) \Rightarrow e \in BP$. Overall, we conclude that $e \in BP$ if and only if $(A_{RQ} = v_i \wedge BP \to v_{in})$, i.e.

$$(A_{RQ} = v_i \wedge BP \to v_{in}) \Leftrightarrow e := (v_{in}, v_i) \in BP \tag{12}$$

Therefore, we have

$$\begin{aligned} &Pr(BP = P_j | A_{RQ} = v_i, BP \to v_{in}) \\ &= Pr(BP = P_j | e := (v_{in}, v_i) \in BP) \\ &= \frac{Pr(e \in BP | BP = P_j) Pr(P_j)}{Pr(e \in BP)} \\ &= \frac{Pr(P_j) Pr(e \in P_j)}{Pr(e \in BP)} = \begin{cases} 0 & e \notin P_j \\ \frac{Pr(P_j)}{\sum_{P \in PS \wedge e \in P} Pr(P)} & otherwise \end{cases} \end{aligned} \tag{13}$$

Condition $BP \nrightarrow v_{in}$: Second, we consider the condition when the best path does not go through v_{in}. Note each vertex in D indicates a path that is possibly the best direction going from v_{in} to t, and we are interested in the best path from the source vertex s to t. Therefore, the answer to RQ gives us useful information only if v_{in} is known to be on the best path. In other words, if v_{in} is not on BP, how to move from v_{in} towards the target does not affect the distribution of BP, since one will not even go to v_{in} in the first place. Probabilistically, BP and A_{RQ} are independent given that 'BP does not go through v_{in}'. Formally, we have

$$A_{RQ} \perp BP | BP \nrightarrow v_{in} \tag{14}$$

where we adopt $\perp$ to denote the operator indicating two random variables are conditionally independent [8].

From Formula 14, we have

$$\begin{aligned} &Pr(BP = P_j | A_{RQ} = v_i, BP \nrightarrow v_{in}) \\ &= Pr(BP = P_j | BP \nrightarrow v_{in}) \qquad Formula\ 14 \\ &= \frac{Pr(BP \nrightarrow v_{in} | BP = P_j) Pr(P_j)}{Pr(BP \nrightarrow v_{in})} \quad Bayes'\ theorem \\ &= \frac{Pr(P_j \nrightarrow v_{in}) Pr(P_j)}{\sum_{P \nrightarrow v_{in}} Pr(P)} \quad Law\ of\ Total\ Probablity \\ &= \begin{cases} 0 & P_j \to v_{in} \\ \frac{Pr(P_j)}{\sum_{P \in PS \wedge P \nrightarrow v_{in}} Pr(P)} & otherwise \end{cases} \end{aligned} \tag{15}$$

Then, by substituting Equations 15 and 13 into Equation 11, we complete the proof. □

Proof of Theorem 3.3

PROOF.

$$\begin{aligned} \Delta H_{RQ} &= H(BP) - \mathbb{E} H(BP | A_{RQ}) \\ &= H(BP) - \sum_{P \in PS} \sum_{v_k \in D} [Pr(A_{RQ} = v_k) \\ &\quad Pr(BP = P | A_{RQ} = v_k) \log Pr(BP = P | A_{RQ} = v_k)] \end{aligned} \tag{16}$$

Let $X_1 = -\sum_{P \nrightarrow v_{in}} \sum_{v_k \in D} [Pr(A_{RQ} = v_k) Pr(BP = P | A_{RQ} = v_k) \log Pr(BP = P | A_{RQ} = v_k)]$ and $X_2 = -\sum_{P \to v_{in}} \sum_{v_k \in D} [Pr(A_{RQ} = v_k) Pr(BP = P | A_{RQ} = v_k) \log Pr(BP = P | A_{RQ} = v_k)]$, then we have

$$\Delta H_{RQ} = H(BP) - X_1 - X_2 \tag{17}$$

Now we analyze X_1 and X_2 separately.

$$\begin{aligned} X_1 = -\sum_{P \nrightarrow v_{in}} \sum_{v_k \in D} &[Pr(A_{RQ} = v_k | BP = P) Pr(BP = P) \\ &\log \frac{Pr(A_{RQ} = v_k | BP = P) Pr(BP = P)}{Pr(A_{RQ} = v_k)}] \end{aligned} \tag{18}$$

Note $P \nrightarrow v_{in}$ and $BP = P$ indicate that $BP \nrightarrow v_{in}$, by Formula 14, we have

$$\begin{aligned} X_1 = - \sum_{P \nrightarrow v_{in}} \sum_{v_k \in D} [&Pr(A_{RQ} = v_k) Pr(BP = P) \\ &\log \frac{Pr(A_{RQ} = v_k) Pr(BP = P)}{Pr(A_{RQ} = v_k)}] \\ = - \sum_{P \nrightarrow v_{in}} &Pr(P) \log Pr(P) \end{aligned} \tag{19}$$

$$\begin{aligned} X_2 = - \sum_{P \rightarrow v_{in}} \sum_{v_k \in D} [&Pr(A_{RQ} = v_k | BP = P) Pr(BP = P) \\ &\log \frac{Pr(A_{RQ} = v_k | BP = P) Pr(BP = P)}{Pr(A_{RQ} = v_k)}] \end{aligned} \tag{20}$$

Note given $BP = P$ and $e := (v_{in}, v_i) \in P$, then $e := (v_{in}, v_i) \in BP$, therefore $A_{RQ} = v_i$ according to Equation 12. Let $e_i := (v_{in}, v_i) \in P_i$, i.e. v_i represents the direction of P_i, we have $Pr(A_{RQ} = v_k | BP = P) = 0$ if $v_k \neq v_i$; and $Pr(A_{RQ} = v_k | BP = P) = 1$ if $v_k = v_i$. Therefore, we write X_2

$$\begin{aligned} X_2 = &- \sum_{P_i \rightarrow v_{in}} \sum_{v_k \in D} [Pr(A_{RQ} = v_k | BP = P_i) Pr(BP = P_i) \\ &\log \frac{Pr(A_{RQ} = v_k | BP = P_i) Pr(BP = P_i)}{Pr(A_{RQ} = v_k)}] \\ = &- \sum_{P_i \rightarrow v_{in}} [Pr(A_{RQ} = v_i | BP = P_i) Pr(BP = P_i) \\ &\log \frac{Pr(A_{RQ} = v_i | BP = P_i) Pr(BP = P_i)}{Pr(A_{RQ} = v_i)}] \\ = &- \sum_{P_i \rightarrow v_{in}} [Pr(BP = P_i) \log \frac{Pr(BP = P_i)}{Pr(A_{RQ} = v_i)}] \end{aligned} \tag{21}$$

By substituting X_1 and X_2 back to Equation 17, we have

$$\Delta H_{RQ} = - \sum_{v_i \in D} \log Pr(A_{RQ} = v_i) \sum_{e:=(v_{in}, v_i) \in P} Pr(P)$$

Then by substituting Equation 2, we complete the proof. □

Proof of Lemma 3.4

PROOF. To prove this lemma, we need to consider three exclusive cases: 1) $P_i \nrightarrow v_{in}$, i.e. P_i does not go through v_{in}, so P_i is not affected by the answer to the RQ; 2) $(v_{in}, v_C) \in P_i$, i.e. P_i goes through v_{in} and v_C, which indicates that the crowdsourced answer is supportive for P_i; 3) $P_i \rightarrow v_{in} \wedge (v_{in}, v_C) \notin P_i$, i.e. P_i goes through v_{in} but not v_C, which indicates that the crowdsourced answer is against for P_i. We list the details for all three cases as follows.

Case 1) $\underline{BP \nrightarrow v_{in}}$: According to Equation 14, the answer to RQ is independent of $\underline{BP}$ given $BP \nrightarrow v_{in}$, so we have
$Pr(BP = P_i | v_C \text{ returned by the crowd}) = Pr(BP = P_i) = Pr(P_i)$;

Case 2) $\underline{(v_{in}, v_C) \in P_i}$: We conduct transformation with Bayes' theorem

$$\begin{aligned} &Pr(BP = P_i | v_C \text{ returned by the crowd}) \\ &= \frac{Pr(P_i) Pr(v_C \text{ returned by the crowd} | BP = P_i)}{Pr(v_C \text{ returned by the crowd})} \end{aligned} \tag{22}$$

On the one hand, we highlight the relation between two probabilistic events - 'v_C *returned by the crowd*' and '$A_{RQ} = v_C$'. '$Pr(A_{RQ} = v_C)$' indicates the probability that 'v_C' is the correct answer of the RQ. For v_C to be returned by the crowd, the event happened is either 'v_C is correct, and the crowd does not make a mistake', or 'v_C is wrong, and the crowd does make a mistake'. Formally, we have

$$\begin{aligned} &Pr(v_C \text{ returned by the crowd}) = \\ &Pr(A_{RQ} = v_C)(1 - \epsilon) + (1 - Pr(A_{RQ} = v_C))\epsilon \end{aligned}$$

On the other hand, because of $BP = P_i$ as well as $(v_{in}, v_C) \in P_i$, we have $(v_{in}, v_C) \in BP$, i.e. v_C is the correct answer to RQ. That means the crowd answers RQ correctly, i.e.

$$\begin{aligned} &Pr(v_C \text{ returned by the crowd} | BP = P_i) = \\ &Pr(\text{crowd answers the } RQ \text{ correctly}) = 1 - \epsilon \end{aligned}$$

So, in case of $(v_{in}, v_C) \in BP$, we have

$$\begin{aligned} &Pr(BP = P_i | v_C \text{ returned by the crowd}) = \\ &\frac{Pr(P_i)(1 - \epsilon)}{Pr(A_{RQ} = v_C)(1 - \epsilon) + (1 - Pr(A_{RQ} = v_C))\epsilon} \end{aligned} \tag{23}$$

where $Pr(A_{RQ} = v_C)$ is derived in Equation 2.

Case 3) $\underline{P_i \rightarrow v_{in} \wedge (v_{in}, v_C) \notin P_i}$: Analogous to case 2), since $(v_{in}, v_C) \notin P_i$ and $(v_{in}, v_C) \notin BP$, we know that v_C is an incorrect answer to RQ conditioning on $BP = P_i$, i.e. the crowd answers RQ correctly. So,

$$\begin{aligned} &Pr(v_C \text{ returned by the crowd} | BP = P_i) = \\ &Pr(\text{crowd answers the } RQ \text{ incorrectly}) = \epsilon \end{aligned}$$

Then we have

$$\begin{aligned} &Pr(BP = P_i | v_C \text{ returned by the crowd}) = \\ &\frac{Pr(P_i)\epsilon}{Pr(A_{RQ} = v_C)(1 - \epsilon) + (1 - Pr(A_{RQ} = v_C))\epsilon} \end{aligned} \tag{24}$$

By concluding the results of the above three cases, we have the proof completed. □

Proof of Theorem 4.1

PROOF. (Sketch) Since the selection hardness is essentially the Shannon entropy, we use $H(.)$ to denote both the selection hardness and the entropy of a random variable.

$$\begin{aligned} \Delta H_{BRQ} &= H(BP) - \mathbb{E}H(BP | A_{BRQ}) \\ &= -H(A_{BRQ} | BP) + H(A_{BRQ}) \\ &= - \sum_{BP = P_i} Pr(P_i) H(A_{BRQ} | BP = P_i) + H(A_{BRQ}) \\ &= - \sum_{P_i \rightarrow v_{in}} Pr(P_i) H(A_{BRQ} | BP = P_i) \\ &\quad - \sum_{P_j \nrightarrow v_{in}} Pr(P_j) H(A_{BRQ} | BP = P_i) + H(A_{BRQ}) \end{aligned} \tag{25}$$

On the one hand, when $P_j \nrightarrow v_{in}$, $BP = P_j$ and A_{BRQ} are independent, analogous to Formula 14. Therefore, we have

$$- \sum_{P_j \nrightarrow v_{in}} Pr(P_j) H(A_{BRQ} | BP = P_j) = - \sum_{P_j \nrightarrow v_{in}} Pr(P_j) H(A_{BRQ}) \tag{26}$$

On the other hand, when $P_i \rightarrow v_{in}$, $BP = P_i$ indicates that the correct of A_{BRQ} must be the direction of P_i. That is to say, we can ensure the correct answer if we know P_i is the best path. Therefore, we have

$$- \sum_{P_j \nrightarrow v_{in}} Pr(P_j) H(A_{BRQ} | BP = P_i) = 0 \tag{27}$$

By substituting Equations 26 and 27, we rewrite Equation 25

$$\begin{aligned} \Delta H_{BRQ} &= H(A_{BRQ}) - \sum_{P_j \nrightarrow v_{in}} Pr(P_j) H(A_{BRQ}) \\ &= -(1 - \sum_{P \nrightarrow v_{in}} Pr(P)) H(A_{BRQ}) \\ &= -(\sum_{P \rightarrow v_{in}} Pr(P)) [Pr(A_{BRQ} = v_d) \log Pr(A_{BRQ} = v_d) \\ &\quad + (1 - Pr(A_{BRQ} = v_d)) \log (1 - Pr(A_{BRQ} = v_d))] \end{aligned} \tag{28}$$

So we have the proof completed. □