Optimal Budget Allocation for Crowdsourcing Labels for Graphs

Adithya Kulkarni[1] **Mohna Chakraborty**[2] **Sihong Xie**[3] **Qi Li**[4]

[1,2,4]Computer Science Dept., Iowa State University, Ames, Iowa, USA
[3]Computer Science & Engineering Dept., Lehigh University, Bethlehem, Pennsylvania, USA

Abstract

Crowdsourcing is an effective and efficient paradigm for obtaining labels for unlabeled corpus employing crowd workers. This work considers the budget allocation problem for a generalized setting on a graph of instances to be labeled where edges encode instance dependencies. Specifically, given a graph and a labeling budget, we propose an optimal policy to allocate the budget among the instances to maximize the overall labeling accuracy. We formulate the problem as a Bayesian Markov Decision Process (MDP), where we define our task as an optimization problem that maximizes the overall label accuracy under budget constraints. Then, we propose a novel stage-wise reward function that considers the effect of worker labels on the whole graph at each timestamp. This reward function is utilized to find an optimal policy for the optimization problem. Theoretically, we show that our proposed policies are consistent when the budget is infinite. We conduct extensive experiments on five real-world graph datasets and demonstrate the effectiveness of the proposed policies to achieve a higher label accuracy under budget constraints.

1 INTRODUCTION

Recently, crowdsourcing platforms like Amazon Mechanical Turk (mTurk) [1] have provided convenient and affordable ways to obtain labels for instances by employing a less expensive crowd of non-expert workers. For labeling data instances, each worker is incentivized with monetary rewards. Each instance can have a different labeling difficulty. Therefore, to properly learn the underlying true label of a hard instance, more workers may be needed compared to an easy instance. Given a pre-fixed budget, it is challenging to optimally allocate the labeling budget to a set of instances, as the allocation decisions have to be made in an online manner to gauge the labeling difficulty of the instances while spending the budget[2].

Previous methods tackle the challenge from different directions. Some of the directions include how to assign instances to proper workers [Zheng et al., 2015, Zhang and Sugiyama, 2015], how to set price for each worker label [Miao et al., 2022, Dizaji et al., 2020], and how to select an instance to query worker label [Sheng et al., 2008, Frazier et al., 2008, Chen et al., 2013, Li et al., 2016]. Specifically, Frazier et al. [2008], Chen et al. [2013], Li et al. [2016] tackle the budget allocation challenge by proposing policies to choose the instances to label. The budget allocation problems are formulated as optimization problems where the objective is to either maximize the overall label accuracy [Frazier et al., 2008, Chen et al., 2013] or maximize the labeling quality [Li et al., 2016]. These studies consider each instance as i.i.d, assuming no dependencies among the instances, and solve the optimization problem using Bayesian Markov Decision Process (MDP) [Bellman, 1957].

However, instances may be related, and the dependencies can be utilized for budget allocation optimization. Specifically, if two instances are dependent, knowing the label of one instance should help infer the label of the other instance. For example, considering citation networks [Bojchevski and Günnemann, 2017] where vertices are publications, and edges indicate citation relationship between publications. The connected publications are thus dependent and likely to belong to the same research area since publications generally cite publications from peers in the same field. Similar observations can be made for social networks [Leskovec and Mcauley, 2012], trust networks [Kumar et al., 2016, 2018], etc.

In this work, we tackle the unique challenge of budget allocation on graphs where each edge connects two dependent

[1]https://www.mturk.com/mturk/

[2]The code can be found at https://github.com/kulkarniadithya/GraphOBA

Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023), PMLR 216:1154–1163.

instances. The instances (i.e., vertices in the graph) cannot be considered as i.i.d since the vertices are dependent. The vertices connected by edges can have a positive or negative pairwise vertex dependency. Due to this dependency and the graph structure, allocating a unit of labeling budget to a vertex need to be considered carefully, as obtaining a noisy label can influence the estimation of the labels of the connected vertices in the graph, especially for high-degree vertices that have more influence than low-degree vertices.

To find an optimal budget allocation policy for graph datasets, we adopt the Bayesian setting and formulate an optimization problem for online budget allocation for labeling instances connected in a graph. The objective is to maximize the overall label accuracy under budget constraints. The final expected accuracy is decomposed as a sum of *stage-wise rewards* following the technique proposed in Xie and Frazier [2012]. The problem discussed by Xie and Frazier [2012] is an *infinite-horizon* one which optimizes the stopping time, but Chen et al. [2013] shows that the technique can also be applied for *finite-horizon* MDP problem. Our proposed stage-wise expected reward computation considers the aggregated change in the distributions of labels of all vertices in the graph. To estimate the stage-wise expected reward of annotating a specific vertex, we infer the label distributions of all vertices given the current noisy vertex labels at stage t using belief propagation (BP) [Pearl, 2022]. Specifically, we treat the noisy label(s) for each labeled vertex as the parameter of the prior Beta distribution. Modulated by the dependencies among the vertices, these noisy priors are then propagated to all the vertices in the graph to infer the posterior vertex distributions. We propose two approximate policies using the proposed stage-wise expected reward and prove that the policies are consistent; that is, when the budget T goes to infinity, the accuracy converges to 100% almost surely.

In summary, we made the following main contributions.

1. We are the first to address the budget allocation problem in the crowdsourcing tasks on a graph of instances where each edge connects dependent instances.

2. We propose a novel stage-wise reward function that considers the effect of worker labels on the whole graph at each stage. Using the novel stage-wise reward function, we propose two optimal approximate policies and theoretically prove that the policies are consistent. To propagate labeling information to other vertices, we model the dependency between vertices as a factor graph and adopt belief propagation.

3. We conduct extensive experiments and ablation studies on the benchmark datasets and empirically validate the effectiveness of the proposed method.

2 RELATED WORKS

The convenient accessibility and affordability of crowdsourcing platforms have motivated many research studies to develop new algorithms and designs for crowdsourcing tasks.

Motivated by the labeling cost concern, some studies [Karger et al., 2014, Zheng et al., 2015, Zhang and Sugiyama, 2015, Wang et al., 2017, Sameki et al., 2019, Liu and Xu, 2020, Tu et al., 2020, Yu et al., 2020] focus on instance assignments to workers. These studies jointly learn worker-instance distribution or the difficulty level of the instances to make a knowledgeable decision on which worker to assign. Another line of studies [Zhang et al., 2015, Gan et al., 2017, Miao et al., 2022, Dizaji et al., 2020, Yin and Chen, 2015] focuses on pricing for each worker label. Specifically, Zhang et al. [2015], Gan et al. [2017] design an online platform as a reverse auction where workers can bid on a task. Zhang et al. [2015] consider a binary labeling task whereas Gan et al. [2017] consider multi-class labeling task. Miao et al. [2022], Yin and Chen [2015] propose a dynamic pricing mechanism to incentivize workers to perform well.

Focusing on how to select instances to query worker labels, Zhou et al. [2014] proposes to use the aggregate regret to select K arms with the highest expected rewards in a stochastic n-armed bandit game. However, this approach does not perform sequential instance selection. More related to our work, Sheng et al. [2008], Li et al. [2016], Frazier et al. [2008], Chen et al. [2013], Raykar and Agrawal [2014] aim to learn a budget allocation policy for the sequence of instance selection. Sheng et al. [2008], Li et al. [2016] aim to maximize the number of labeled instances while maintaining the quality requirements under the given budget. Sheng et al. [2008] assumes that data quality is the same in all instances, whereas Li et al. [2016] assumes that data quality would be higher for easy instances. Raykar and Agrawal [2014] aim to maximize a utility function with consideration of the pull market (i.e., workers may not accept jobs from requesters). Frazier et al. [2008], Chen et al. [2013] have similar goals to ours but consider each instance as i.i.d.. Frazier et al. [2008] propose a knowledge gradient policy to sequentially select instances to label. Chen et al. [2013] show that the knowledge gradient policy is not consistent and propose an optimistic knowledge gradient policy and show that it is a consistent policy when the budget is infinite.

Unlike prior works, our goal is to obtain an optimal budget allocation policy for a graph of instances where edge connects dependent instances. Our proposed policy considers dependency between instances and the influence of each instance on other instances of the graph to obtain a policy that maximizes the overall label accuracy under budget constraints. To the best of our knowledge, we are the first to consider budget allocation policy for a graph of instances.

3 PRELIMINARIES

3.1 PROBLEM FORMULATION

Consider a graph $G = (V, E)$, each vertex $v_i \in V$ for $1 \leq i \leq N$ of the graph represents an instance whose true label is unknown, and the edge set E contains edges connecting dependent instances. Since we consider binary labeling tasks, for each edge $e = (v_i, v_j) \in E$, let C_{ij} be a 2×2 matrix representing the pairwise vertex dependency between the vertices v_i and v_j. Each vertex $v_i \in V$ is associated with a true label $l_i \in \{+1, -1\}$ for $1 \leq i \leq N$ and is characterized with the probability of being in class $+1$ denoted by $\theta_v \in [0, 1]$. The label provided by a worker for any given vertex $v \in V$ at any given timestamp t denoted by y_{v_t} is drawn from the underlying label distribution, $y_{v_t} \sim Bernoulli(\theta_v)$. A label costs one unit of budget. Given a budget T, the goal is to maximize the overall label accuracy, which is measured on the inference of true labels of the vertices given the worker labels. Intuitively, with a larger budget, we can estimate θ_v more accurately for each vertex $v \in V$ and thus achieving better overall label accuracy.

3.2 BELIEF PROPAGATION

Belief propagation (BP) [Pearl, 2022] is a message-passing algorithm. For each edge (v_i, v_j), two messages, $\underset{v_i \to v_j}{\mu}$ and $\underset{v_j \to v_i}{\mu}$, are propagated, one in each direction. A message from vertex v_i to vertex v_j essentially contains all the information from the subtree rooted at v_i.

We convert the input graph G into a factor graph FG to apply belief propagation. A factor graph $FG = (V \cup F, E')$ is a bipartite graph with variables V and factors F as vertices and edges E' connecting variables and factors. The variables V of the factor graph are the N instances of G, and for each edge $e = (v_i, v_j) \in E$, we add a factor vertex F_e. Each factor vertex $f \in F$ has a function ϕ_f that models the pairwise vertex dependency matrix C_{ij}. The factor vertex F_e is connected to the variable vertices v_i and v_j using undirected edges.

For a factor graph FG, messages μ are passed between variable vertex $v \in V$ and factor vertex $f \in F$. The messages are computed differently depending on whether the vertex receiving the message is a variable vertex or a factor vertex.

$$\underset{v \to f}{\mu}(x_v) = \prod_{f^* \in \mathcal{N}(v) \setminus \{f\}} \underset{f^* \to v}{\mu}(x_v), \tag{1}$$

$$\underset{f \to v}{\mu}(x_v) = \sum_{x'_f = x_v, x'_v = x_v} \left(\phi_f(x'_f) \prod_{v^* \in \mathcal{N}(f) \setminus \{v\}} \underset{v^* \to f}{\mu}(x'_{v^*}) \right), \tag{2}$$

where $\forall\, v \in V$, $x_v \in \{+1, -1\}$ represents the labels that variable vertex v can take, $\mathcal{N}(v)$ and $\mathcal{N}(f)$ represent the sets of neighboring vertices of v and f, respectively.

In each iteration, an arbitrary vertex is chosen as a *root*, and then messages are passed from leaf vertices in the graph FG to the root (*forward propagation*) and then back to the leaf vertices (*backward propagation*). In both *forward* and *backward* propagations, the messages are initiated from variable vertices $v \in V$. The message from each vertex $v \in V$ is initialized with its prior/posterior probability ω_v. Following *forward* and *backward* propagation, the message between adjacent vertices is updated iteratively as per Eq. (1) and Eq. (2) until convergence. Furthermore, the messages are normalized in each step to avoid underflow. Upon convergence, the marginal probability of each variable vertex $v \in V$ is:

$$P_v(x_v) \propto \omega_v(x_v) \prod_{j \in \mathcal{N}(v)} \underset{j \to v}{\mu}(x_v), \tag{3}$$

where $\omega_v(x_v)$ is the prior/posterior probability of x_v.

3.3 KG AND OPTKG POLICY

Knowledge Gradient (KG) [Frazier et al., 2008] and Optimistic Knowledge Gradient (OPTKG) [Chen et al., 2013] provide policies to sequentially select instances to obtain worker labels. These methods consider each instance as i.i.d and formulate the budget allocation problem as an optimization problem. To find an optimal policy, these methods define a stage-wise reward function. At each timestamp, they select the next instance that maximizes the reward. Specifically, KG is a single-step look-ahead policy that greedily selects the next instance with the largest expected reward:

$$v_t = \underset{v}{\operatorname{argmax}} \left(R(S^t, v) \doteq p_1 * R_1(a_v^t, b_v^t) + p_2 * R_2(a_v^t, b_v^t) \right), \tag{4}$$

where a_v^t and b_v^t represent the number of labels belonging to positive and negative classes, respectively, of vertex v at timestamp t. $p_1 = \frac{a_v^t}{a_v^t + b_v^t}$ and $p_2 = \frac{b_v^t}{a_v^t + b_v^t}$ are posterior probabilities of v_t, and $R_1(a_v^t, b_v^t)$, $R_2(a_v^t, b_v^t)$ are the rewards of getting label $+1$ and -1, respectively, for vertex v.

OPTKG selects the next instance based on the optimistic outcome of the reward:

$$v_t = \underset{v}{\operatorname{argmax}} \left(R^+(S^t, v) \doteq \max(R_1(a_v^t, b_v^t), R_2(a_v^t, b_v^t)) \right). \tag{5}$$

Computationally, both KG and OPTKG have the time complexity $O(NT)$ and space complexity $O(N)$. However, their reward estimation considers each instance separately since instances are considered i.i.d.

4 METHODOLOGY

Our goal is to find an optimal budget allocation policy for graph datasets. The policy should properly estimate the underlying true label of each vertex, considering the dependency between vertices and the influence of each vertex on other vertices in the graph. We formulate our problem in the Bayesian setting. A detailed discussion is provided in Section 4.1. According to the Bayesian setup, we define our task as an optimization problem that maximizes the overall label accuracy in the given budget T, which we discuss in Section 4.2. The optimization problem is formulated into a Markov Decision Process to find the optimal policy. Then, we define our stage-wise expected reward that considers the probability distribution of every vertex in the graph after each iteration in Section 4.3. We define two approximate policies for the problem and theoretically prove that the policies are consistent. A detailed discussion is provided in Section 4.4.

4.1 BAYESIAN SETUP

Since the true labels of the vertices in the graph are unknown, we initialize θ_v with a Beta prior distribution Beta(α, β). Specifically, α and β values are set to 0.1. This initialization can be interpreted as having α positive and β negative pseudo-labels for the vertex v at the initial stage.

We aim to model each worker label's effect on the whole graph. A worker label can update the marginal probabilities of vertices in the graph. Therefore, we define the state matrix S^t as a $N \times 2$ matrix representing the marginal probabilities of the vertices in the graph. At each timestamp, depending on the choice of vertex and the label obtained, the marginal probabilities of vertices are updated, and we transition to the new state S^{t+1}.

We can observe that S^t is a Markovian process because S^{t+1} is completely determined by the current state S^t, the action v_t and the obtained label y_{v_t}. Specifically, the change in marginal probability of vertices in graph FG between timestamps is only due to the action v_t and obtained label y_{v_t}. Moreover, suppose we choose v_t to obtain a worker label in the current state S^t. In that case, we can calculate the state transition probability $Pr(y_{v_t}|S^t, v_t)$, which is the posterior probability that we are in the next state S^{t+1} since each worker label at any given timestamp t is drawn from the underlying label distribution.

$$Pr(y_{v_t} = +1|S^t, v_t) = \mathbb{E}(\theta_{v_t}|S^t) = \frac{\alpha + a_v^t}{\alpha + a_v^t + \beta + b_v^t}, \tag{6}$$

where a_v^t and b_v^t are the number of positive and negative worker labels obtained for vertex v till timestamp t and $Pr(y_{v_t} = -1|S^t, v_t) = 1 - Pr(y_{v_t} = +1|S^t, v_t)$. Following the above labeling process, a filtration $\{\mathcal{F}_t\}_{t=0}^{T-1}$ is defined, where $\mathcal{F}_t$ is the σ-algebra generated by the sample path $(v_0, y_{v_0}, ..., v_{t-1}, y_{v_{t-1}})$. The choice of the next vertex to label v_t at timestamp t is done after observing the historical labeling results up to the timestamp $t-1$. Therefore, v_t is $\mathcal{F}_t$-measurable. Hence, the process of budget allocation is defined as a sequence of choices $\pi = (v_0, ..., v_{T-1})$.

4.2 OBJECTIVE FUNCTION

Our goal is to maximize the overall prediction accuracy once the budget is exhausted at timestamp T. The true label of each variable vertex $v \in V$ is inferred based on their marginal probability at timestamp T. Since the task is binary, we need to determine the positive set H_T that maximizes the *conditional* expected accuracy conditioning on $\mathcal{F}_T$:

$$H_T = \underset{H \subset \{1,...,N\}}{\operatorname{argmax}} \ \mathbb{E}\left(\sum_{v \in H} \mathbf{1}(v \in H^*) + \sum_{v \notin H} \mathbf{1}(v \notin H^*)|\mathcal{F}_T\right), \tag{7}$$

where H^* refers to the set of vertices with ground truth true labels $+1$, H refers to the set of vertices with the estimated label $+1$, H_T is the set H that maximizes Eq. (7), and $\mathbf{1}(.)$ is the indicator function. For $0 \leq t < T$, the conditional distribution $\theta_v|\mathcal{F}_t$ is exactly the marginal probability calculated using Eq. (3) that depends on the historical sampling results only through S^t. Therefore, we define

$$P_v^t(+1) = Pr(v \in H^*|\mathcal{F}_t) = Pr(\theta_v \geq 0.5|S^t). \tag{8}$$

Xie and Frazier [2012] show that the final positive set H_T can be determined by the Bayes decision rule.

Similar to Chen et al. [2013], we define the following proposition to solve Eq. (7).

Proposition 1 *$H_T = \{v : P_v^T(+1) \geq 0.5\}$ solves Eq. (7) and the expected accuracy on RHS of Eq. (7) can be written as $\sum_{v=1}^{N} h(P_v^T(+1))$, where $h(z) = \max(z, 1-z)$.*

In order to find the optimal policy that maximizes the expected accuracy, the following optimization problem should be solved:

$$\begin{aligned} V(S^0) \doteq \sup_{\pi} \mathbb{E}^{\pi} &\left[\mathbb{E}\left(\sum_{v \in H_T} \mathbf{1}(v \in H^*) + \sum_{v \notin H_T} \mathbf{1}(v \notin H^*)|\mathcal{F}_T\right)\right] \\ &= \sup_{\pi} \mathbb{E}^{\pi}\left(\sum_{v=1}^{N} h(P_v^T(+1))\right), \end{aligned} \tag{9}$$

where $\mathbb{E}^{\pi}$ represents the expectation taken over the sample paths $(v_0, y_{v_0}, ..., v_{T-1}, y_{v_{T-1}})$ generated by a policy π. The second equality is due to Proposition 1 and $V(S^0)$ is the value function at the initial state S^0. Any policy π that attains the supremum in Eq. (9) is the optimal policy π^*.

4.3 OPTIMAL POLICY

To obtain the optimal policy π^*, we formulate the optimization problem in Eq. (9) into a Markov Decision Process (MDP). The final expected accuracy is decomposed as a sum of *stage-wise rewards* following the technique proposed in Xie and Frazier [2012]. The problem discussed by Xie and Frazier [2012] is an *infinite-horizon* one which optimizes the stopping time, but Chen et al. [2013] shows that the technique can also be applied for *finite-horizon* problem. We consider the marginal probability of every vertex in the graph by taking the sum of marginal probabilities at each timestamp. Then, we define the reward function as the change in the sum of marginal probabilities between two timestamps.

Proposition 2 *The stage-wise expected reward is defined as:*

$$R(S^t, v_t) = \mathbb{E}(\sum_{k=1}^{N} h(P_k^{t+1}(+1)) - \sum_{k=1}^{N} h(P_k^t(+1))|S^t, v_t), \tag{10}$$

then the value function in Eq. (9) becomes:

$$V(S^0) = G_0(S^0) + \sup_{\pi} \mathbb{E}^{\pi} \left(\sum_{t=0}^{T-1} R(S^t, v_t) \right), \tag{11}$$

where $G_0(S^0) = \sum_{k=1}^{N} h(P_k^0(+1))$ and any policy π that attains the supremum is the optimal policy π^*.

The detailed steps of the derivation are provided in Appendix A. Using Proposition 2, the maximization problem in Eq. (9) is formulated as a T-stage MDP (Eq. (10)), which is associated with a tuple $\{T, \{\mathcal{S}^t\}, \mathcal{A}, \mathcal{P}^t, R(S^t, v_t)\}$. Here, $\mathcal{S}^t$, the state space at stage t, is all possible states that can be reached at stage t. Once a label y_{v_t} is obtained for a variable vertex v at timestamp t, the marginal probability of more than one variable vertex $v' \in V$ can change. Therefore, we have

$$\mathcal{S}^t = \left\{ \{p_{1_v}^t, p_{2_v}^t\}_{v=1}^N : p_{1_v}^t, p_{2_v}^t \in [0,1], p_{1_v}^t + p_{2_v}^t = 1 \right\}. \tag{12}$$

The action space $\mathcal{A} = \{1, 2, ..., N\}$ is the set of instances that could be labeled next. $\mathcal{P}^t = \{P_1^t, P_2^t, ..., P_N^t\}$ is the set of marginal probabilities at timestamp t of each variable vertex $v \in V$ and $R(S^t, v_t)$ is the expected reward defined in Eq. (13). Moreover, due to the Markovian property of $\{S^t\}$, it is sufficient to consider a Markovian policy [Powell, 2007], where v_t is chosen only based on the state S^t.

4.4 EFFICIENT APPROXIMATE POLICY

Our goal is to choose the next vertex to obtain a worker label. Since our problem is an optimization problem and we model it as T-stage MDP (Eq. (10)), at each timestamp, we need to select the vertex that has the maximum stage-wise expected reward as the next vertex. At any given state S^t at timestamp t, if any vertex $v \in V$ is chosen to obtain a worker label, it can get a label of $+1$ or -1. Therefore, to compute the stage-wise expected reward, we need to consider both possibilities. Let $R_1(S^t, v_t)$, $R_2(S^t, v_t)$ be the reward of getting label $+1$ and -1, respectively. Then, the expected reward is:

$$R(S^t, v_t) = p_1 * R_1(S^t, v_t) + p_2 * R_2(S^t, v_t), \tag{13}$$

where $p_1 = \frac{\alpha + a_v^t}{\alpha + a_v^t + \beta + b_v^t}$ and $p_2 = \frac{\beta + b_v^t}{\alpha + a_v^t + \beta + b_v^t}$ are posterior probabilities of v_t. Therefore, the next vertex is the one that has the maximum expected reward:

$$v_t = \underset{v}{\operatorname{argmax}} \left(R(S^t, v_t) \doteq p_1 * R_1(S^t, v_t) + p_2 * R_2(S^t, v_t) \right). \tag{14}$$

Following Eq. (14), we can find the next vertex at each timestamp $0 \leq t < T$ and obtain the policy $\hat{\pi} = (v_0, ..., v_{T-1})$ which we call GraphOBA-EXP. Furthermore, similar to Eq. (5), we can also choose the next vertex based on the optimistic outcome of the reward:

$$v_t = \underset{v}{\operatorname{argmax}} \left(R^+(S^t, v_t) \doteq \max(R_1(S^t, v_t), R_2(S^t, v_t)) \right). \tag{15}$$

We call the policy $\pi^o = (v_0, ..., v_{T-1})$ obtained following Eq. (15) GraphOBA-OPT.

GraphOBA-EXP and GraphOBA-OPT require computation of $R_1(S^t, v_t)$ and $R_2(S^t, v_t)$. Therefore, we need to compute the change in the sum of marginal probabilities due to a new label $+1$ and -1, respectively.

For the computation, we utilize the belief propagation (BP) algorithm to propagate labeling information throughout the graph. Each factor vertex $f \in F$ in graph FG has a function ϕ_f that models the provided pairwise vertex dependency C_{ij}. To compute $R_1(S^t, v_t)$[3], assuming that BP converged at timestamp t, we first compute marginal probabilities of each variable vertex $v \in V$ following Eq. (3). Then, $\sum_{v=1}^{N} h(P_v^t(+1))$ is computed using the marginal probabilities.

Following *forward propagation*, the messages are passed from leaf vertices of the factor graph FG to the variable vertex v_t. Then, v_t is assigned label $+1$ and the current posterior distribution Beta$(\alpha + a_v^t, \beta + b_v^t)$ of the variable vertex v_t is updated. Since Beta is the conjugate prior of the Bernoulli, the posterior of $\theta_{v_t}(\omega_v)$ at the timestamp $t+1$ will be updated as Beta$(\alpha + a_v^{t+1}, \beta + b_v^{t+1})$ =

[3]To compute the expected reward of each vertex $v \in V$, different temporary parallel environments similar to the main environment are created so that the main environment is not affected.

Table 1: Statistics of the Datasets

Dataset	#Vertex	#Pos	#Neg	#Train	#Test
Cora	2708	1296	1412	2166	542
Citeseer	3312	1618	1694	2650	662
Pubmed	19717	7875	11842	15774	3943
WebKB	877	415	462	4705	1176
Bitcoin	5881	2914	2967	702	175

Beta($\alpha + a_v^t + 1, \beta + b_v^t$). Once ω_v is updated, the messages are *backward propagated* from variable vertex v_t to the leaf vertices of FG. BP may not converge in one iteration; therefore, *forward* and *backward* propagation steps are run multiple times but without assigning any new label to v_t. Upon convergence, Eq. (3) is used to compute the updated marginal probability for each variable vertex $v \in V$ and compute $\sum_{v=1}^{N} h(P_v^{t+1}(+1))$. The difference between $\sum_{v=1}^{N} h(P_v^{t+1}(+1))$ and $\sum_{v=1}^{N} h(P_v^{t}(+1))$ is the reward $R_1(S^t, v_t)$. Similarly, $R_2(S^t, v_t)$ is computed where the assigned label is -1.

R_1 and R_2 are computed for all vertices $v \in V$ and the next vertex is chosen following Eq. (14) if GraphOBA-EXP is followed and Eq. (15) if GraphOBA-OPT is followed. Once the vertex is chosen, a worker label is obtained for the vertex. The marginal probabilities of each vertex of the main environment are updated by propagating labeling information using belief propagation.

Given the pairwise vertex dependency (C_{ij}) among all pairs of adjacent variable vertices v_i and v_j, the next theorem shows that the policies $\hat{\pi}$ and π^o are consistent for the problem.

Theorem 1 *Given the pairwise vertex dependency (C_{ij}) among all pairs of adjacent variable vertices v_i and v_j and $\alpha, \beta > 0$, the policies $\hat{\pi}$ and π^o are consistent, i.e., as the budget T goes to infinity, the accuracy will be 100% almost surely (i.e., $H_T = H^*$).*

To prove the theorem, we show that the marginal probability of each vertex $v \in V$ is updated only due to its posterior probability and posterior probabilities of leaf vertices in the factor graph FG. Then, we show that the reward function is proportional to the change in the posterior probability of chosen vertex v_t, and both GraphOBA-EXP and GraphOBA-OPT will label each vertex infinitely many times as the budget goes to infinity. Since we consider workers reliable, if we label each vertex infinitely many times, we will converge to θ_v for each $v \in V$. Therefore, the accuracy will be 100%, almost surely implying that $\hat{\pi}$ and π^o are consistent policies. The theorem is proved in Appendix B.

5 EXPERIMENTS

In this section, we evaluate two versions of our proposed approach, GraphOBA-EXP that chooses the next vertex to label following Eq. (14) and GraphOBA-OPT that chooses the next vertex to label following Eq. (15). We compare our proposed approaches with baselines on five benchmark graph datasets with different statistics and from different domains. More studies can be found in the Appendix C.

5.1 DATASET

The performance of GraphOBA is evaluated across five graph datasets. Three of the datasets, Cora, Citeseer, Pubmed Bojchevski and Günnemann [2017] are citation networks, Bitcoin [Kumar et al., 2016, 2018] is a trust network between Bitcoin users, and WebKB [Craven et al., 1998] is a dataset that includes web pages from computer science departments of various universities. The datasets are multi-class, so we combine the classes to convert the datasets into binary-class datasets. Each dataset is split randomly into train and test sets in an 8:2 ratio. All policies can only label vertex in the train set. The labeling information of vertices in the train set is propagated to vertices in the test set using belief propagation. The statistics of the datasets can be found in Table 1, and the pre-processing steps can be found in Appendix D.

5.2 EVALUATION METRICS

Since the goal of the proposed method is to maximize accuracy under budget constraints. We compare with the baselines using *accuracy* as the evaluation metric.

5.3 BASELINE METHODS

We obtain instances to label following the baseline policies. Once the policies are obtained, the baselines are compared under two settings: (1) without BP and (2) with BP.

When the budget is lower than two times the number of instances, KG and OPTKG policies follow a round-robin policy and are equivalent. Since the budget for all our experiments is lower than two times the number of instances, we only compare with OPTKG. The following are the baselines[4]. We compare with the following:

1. *Uniform:* Randomly sample one vertex from the train set of the graph.
2. *OPTKG*: Optimistic Knowledge Gradient [Chen et al., 2013] policy that follows Eq. (5).

[4]Though Gittins et al. [2011] and Nino-Mora [2011] can be used for the problem, they are computationally expensive. The calibration method Gittins et al. [2011] and Nino-Mora [2011] and state-of-the-art exact method Nino-Mora [2011] require $O(T^3)$ and $O(T^6)$ time and space complexity, respectively. Therefore, we do not compare them in our experiments.

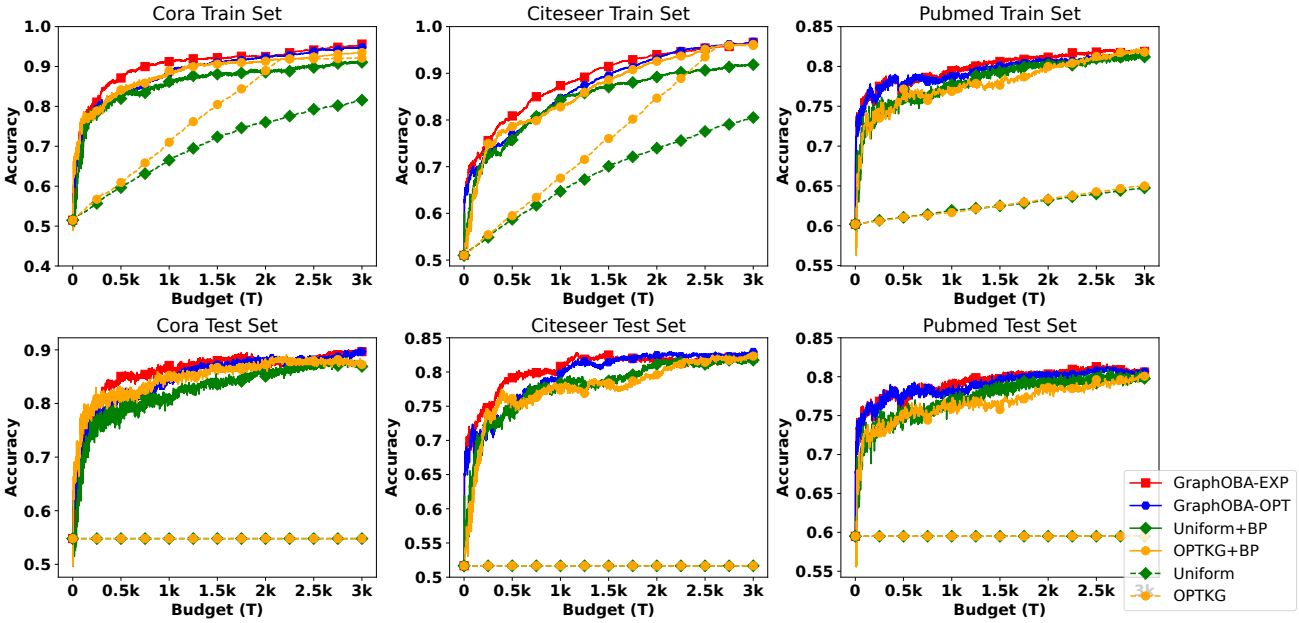

Figure 1: Performance comparison on datasets that follow homophily setting. The top three plots show the performance on the train set, and the bottom three plots show the performance on the test set due to the train set labeling information propagation using BP.

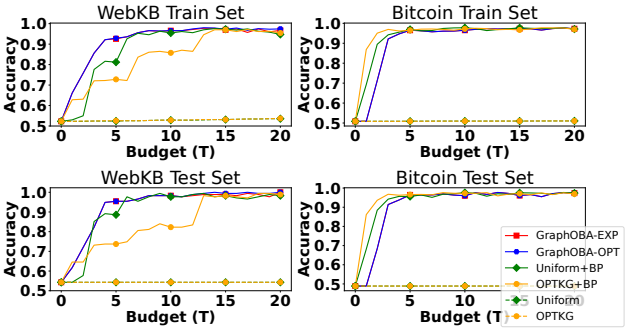

Figure 2: Performance comparison on WebKB and Bitcoin datasets. The top two plots show the performance on the train set, and the bottom two plots show the performance on the test set due to the train set labeling information propagation using BP.

3. *Uniform+BP*: Uniform policy and then apply belief propagation to propagate labeling information.
4. *OPTKG+BP*: Optimistic Knowledge Gradient [Chen et al., 2013] policy that follows Eq. (5). Belief propagation is applied to propagate labeling information.

5.4 EXPERIMENTAL SETTINGS

For each dataset, we simulate reliable workers for the experiments. Since Cora, Citeseer, Pubmed, and WebKB datasets have features for vertices, a logistic regression model is trained on the whole dataset with the vertex features as input. Then, for each vertex v in the dataset, the trained model provides a probability of being in class $+1$, which is used as θ_v. Bitcoin dataset does not have features for vertices. Therefore, we use ground truth to decide θ_v for each vertex v. If the ground truth label of the vertex is $+1$, θ_v is set to 0.93; otherwise, it is set to 0.07. Finally, to obtain labels provided by reliable workers for each vertex v, random samples are drawn from $Bernoulli(\theta_v)$.

The pairwise vertex dependency among adjacent vertices follows a homophily setting in Cora, Citeseer, and Pubmed datasets. Therefore, each factor vertex for these datasets is initialized with the same value following the homophily setting (probability of connect vertices having the same label is 0.51 and different label is 0.49).

WebKB and Bitcoin datasets do not follow the homophily setting. Therefore ground truth labels are used to infer pairwise vertex dependency. For a pair of adjacent vertices, the probability of both vertices having the same label is set to 0.9 if the ground truth labels match; otherwise, it is set to 0.1. The experiments on these datasets represent the scenario where we know the entire factor graph and the workers are reliable.

Ideally, as per Eq. (14) and Eq. (15), we should compute the expected reward for all vertices in the train set at each timestamp and choose the vertex with the maximum expected reward. However, belief propagation (BP) is computationally expensive since each iteration of BP has a time complexity of $O(|V \cup F|^2)$ on the factor graph FG, and BP can take several iterations to converge. Therefore, at each timestamp, we uniformly sample 10 vertices from the train set to compute the expected reward and choose the vertex based on the policy. All the experiments are conducted with a random seed value of 11, and the value of α and β in the Beta prior distribution Beta(α, β) is set to 0.1. We provide pseudo code for optimal policy π^* computation for GraphOBA-OPT and

GraphOBA-EXP in Algorithm 1 in Appendix C.

5.5 RESULTS AND DISCUSSION

In Figure 1, we compare the two versions of our proposed approach with the baselines on datasets that follow a homophily setting. Considering the results on the train set for different datasets, we can observe that since OPTKG follows a round-robin policy, its performance grows linearly with the budget till all the vertices in the train set are labeled, whereas the performance growth of Uniform policy follows near logarithmic curve.

We can also observe that applying belief propagation significantly improves the performance of both OPTKG and Uniform policies for all three datasets. The results suggest that propagating labeling information by considering dependency among adjacent vertices can help achieve significantly higher performance when the budget is low.

Comparing baselines with the proposed approaches, we can observe that GraphOBA-EXP outperforms the baselines for all three datasets, whereas GraphOBA-OPT comes in second. The performance improvement of GraphOBA-EXP and GraphOBA-OPT is significant when the budget is low. The results suggest that the vertices chosen by the proposed reward function are influential in the graph.

Considering the results on the test set for different datasets, since OPTKG and Uniform policies do not propagate labeling information, their results correspond to the initial prior distribution. Comparing the remaining baselines, we can observe that GraphOBA-EXP outperforms all the baselines for all three datasets, whereas GraphOBA-OPT comes second. The observation is similar to our observation for the train set, suggesting the importance of choosing the right vertex at each timestamp in labeling information propagation. Choosing the right vertex can result in a larger change in the marginal probabilities of vertices in the graph resulting in faster convergence to the true label distribution.

Figure 2 compares our proposed approaches with the baselines on WebKB and Bitcoin datasets. The results show that the proposed setup can achieve near 100% accuracy with very little budget. Empirically, this suggests that when the ground truth factor graph is known, applying belief propagation can achieve nearly 100% accuracy. The results suggest that knowing the dependency among adjacent vertices in the graph is important. From the figure, we can observe that the proposed approaches outperform the baselines for the WebKB dataset and achieve comparable performance on the Bitcoin dataset for both train and test sets.

We also conduct experiments without splitting the dataset into train and test sets. The results, along with the discussion, can be found in Appendix C.

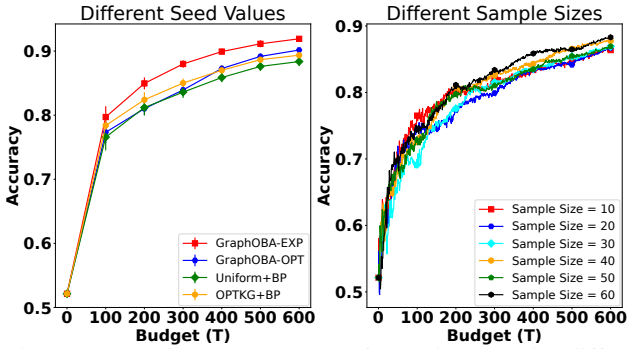

Figure 3: Ablation study results of experiments with different seed values (left plot) and sample sizes (right plot) on the Cora dataset. We plot the means and standard deviations for experiments obtained from different seed values (left plot), and for experiments with different sample sizes (right plot), we report the performance of GraphOBA-EXP.

6 ABLATION STUDIES

We conduct ablation studies to explore the importance of various hyperparameters. All the experiments are conducted on the entire Cora dataset without splitting it into train and test sets. More studies can be found in the Appendix C.

The experiments in Figure 1 and 2 are conducted with a random seed value of 11. However, different seed values can result in different results. Therefore, we conduct experiments with different seed values $(1, 11, 42, 78, 96, 111)$, and the mean and standard deviation are shown in Figure 3. We observe from the results that the proposed approaches, including the baselines Uniform+BP and OPTKG+BP, have a larger variance when the budget is low, and the variance gradually reduces as the budget increases. However, compared to the baselines, the proposed approaches have lower variance, suggesting that the proposed approaches are more robust. The plot of average results of different seed initialization is similar to Figure 1, suggesting that GraphOBA-EXP and GraphOBA-OPT outperform baselines for different seed initialization.

The experiments in Figure 1 and 2 are conducted with a sample size of 10. Intuitively, one may expect to achieve better performance with a larger sample size since there are more candidate vertices to choose from. Therefore, we conduct experiments with different sample sizes $(10, 20, 30, 40, 50, 60)$ using GraphOBA-EXP, and the results are shown in Figure 3. The results confirm that policies with larger sample sizes tend to perform better, but all the policies converge to similar performance when the budget is sufficient.

We also conduct ablation studies with different initialization of α and β and different initialization for factors on the Cora dataset. Results and detailed discussions can be found in Appendix C.

7 CONCLUSION

This work addresses the budget allocation problem in the crowdsourcing tasks on a graph of instances where each edge connects dependent instances. We formulate the problem as an MDP and define the task as an optimization problem that maximizes the overall label accuracy under budget constraints. We propose a novel stage-wise reward function to take advantage of the graph structure and dependency among vertices. We propose two optimal policies using this reward function and theoretically prove that the policies are consistent when the budget is infinite. To propagate labeling information throughout the graph, we convert the input graph into a factor graph and apply belief propagation. The results on five real-world graph datasets demonstrate the effectiveness of the proposed approach.

8 ACKNOWLEDGEMENTS

Adithya, Mohna, and Qi were supported in part by the National Science Foundation under NSF grant IIS-2007941. Sihong was supported in part by the National Science Foundation under NSF Grants IIS-1909879, CNS-1931042, IIS-2008155, and IIS-2145922.

References

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

Xi Chen, Qihang Lin, and Dengyong Zhou. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *International conference on machine learning*, pages 64–72. PMLR, 2013.

Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag. Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-mellon univ pittsburgh pa school of computer Science, 1998.

Kamran Ghasedi Dizaji, Hongchang Gao, Yanhua Yang, Heng Huang, and Cheng Deng. Robust cumulative crowdsourcing framework using new incentive payment function and joint aggregation model. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4610–4621, 2020.

Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47 (5):2410–2439, 2008.

Xiaoying Gan, Xiong Wang, Wenhao Niu, Gai Hang, Xiaohua Tian, Xinbing Wang, and Jun Xu. Incentivize multi-class crowd labeling under budget constraint. *IEEE Journal on Selected Areas in Communications*, 35(4): 893–905, 2017.

John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

David R Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.

Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 221–230. IEEE, 2016.

Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341. ACM, 2018.

Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

Qi Li, Fenglong Ma, Jing Gao, Lu Su, and Christopher J Quinn. Crowdsourcing high quality labels with a tight budget. In *Proceedings of the ninth acm international conference on web search and data mining*, pages 237–246, 2016.

Jia-Xu Liu and Ke Xu. Budget-aware online task assignment in spatial crowdsourcing. *World Wide Web*, 23:289–311, 2020.

Xiaoye Miao, Huanhuan Peng, Yunjun Gao, Zongfu Zhang, and Jianwei Yin. On dynamically pricing crowdsourcing tasks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2022.

José Nino-Mora. Computing a classic index for finite-horizon bandits. *INFORMS Journal on Computing*, 23 (2):254–267, 2011.

Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 129–138. 2022.

Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

Vikas Raykar and Priyanka Agrawal. Sequential crowdsourced labeling as an epsilon-greedy exploration in a markov decision process. In *Artificial intelligence and statistics*, pages 832–840. PMLR, 2014.

Mehrnoosh Sameki, Sha Lai, Kate K Mays, Lei Guo, Prakash Ishwar, and Margrit Betke. Buoca: budget-optimized crowd worker allocation. *arXiv preprint arXiv:1901.06237*, 2019.

Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622, 2008.

Jinzheng Tu, Guoxian Yu, Jun Wang, Carlotta Domeniconi, Maozu Guo, and Xiangliang Zhang. Crowdwt: Crowdsourcing via joint modeling of workers and tasks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(1):1–24, 2020.

Wei Wang, Xiang-Yu Guo, Shao-Yuan Li, Yuan Jiang, and Zhi-Hua Zhou. Obtaining high-quality label by distinguishing between easy and hard items in crowdsourcing. In *IJCAI*, volume 17, pages 2964–2970, 2017.

Jing Xie and Peter I Frazier. Sequential bayes-optimal policies for multiple comparions with a control. Technical report, Technical report, Cornell University, 2012.

Ming Yin and Yiling Chen. Bonus or not? learn to reward in crowdsourcing. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Guoxian Yu, Jinzheng Tu, Jun Wang, Carlotta Domeniconi, and Xiangliang Zhang. Active multilabel crowd consensus. *IEEE Transactions on Neural Networks and Learning Systems*, 32(4):1448–1459, 2020.

Hao Zhang and Masashi Sugiyama. Task selection for bandit-based task assignment in heterogeneous crowdsourcing. In *2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 164–171. IEEE, 2015.

Qi Zhang, Yutian Wen, Xiaohua Tian, Xiaoying Gan, and Xinbing Wang. Incentivize crowd labeling under budget constraint. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2812–2820. IEEE, 2015.

Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. Qasca: A quality-aware task assignment system for crowdsourcing applications. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1031–1046, 2015.

Yuan Zhou, Xi Chen, and Jian Li. Optimal pac multiple arm identification with applications to crowdsourcing. In *International Conference on Machine Learning*, pages 217–225. PMLR, 2014.