

# Crowdsourced Entity Alignment: A Decision Theory Based Approach

Yan Zhuang, Guoliang Li<sup>(✉)</sup>, and Jianhua Feng

Department of Computer Science, Tsinghua University, Beijing, China  
zhuang-y14@mails.tsinghua.edu.cn, {liguoliang, fengjh}@tsinghua.edu.cn

**Abstract.** Crowdsourcing is a new computation paradigm that utilizes the wisdom of the crowd to solve problems which are difficult for computers (e.g., image annotation and entity alignment). In crowdsourced entity alignment tasks, there are usually large numbers of candidate pairs to be verified by the crowd workers, and each pair will be assigned to multiple workers to achieve high quality. Thus, two fundamental problems are raised: (1) question selection – what are the most beneficial questions that should be crowdsourced, and (2) question assignment – which workers should be assigned to answer a selected question? In this paper, we address these two problems by decision theory. Firstly, we define the problems on two budget constraints. The first takes the marginal gain into account, and the second focuses on the limited budget. Then, we formulate the decision-making problems under different budget constraints and build influence diagram to perform result inference. We propose two efficient algorithms to address these two problems. Finally, we conduct extensive experiments to validate the efficiency and effectiveness of our proposed algorithms on both synthetic and real data.

**Keywords:** Entity alignment · Crowdsourcing · Decision theory

## 1 Introduction

Entity alignment which aims at finding entities from two different datasets that refer to the same entity is an important problem for data integration and cleaning. It has been studied extensively for several decades. Recent advances in crowdsourcing technologies enable entity alignment to be performed by online workers. Crowdsourced entity alignment takes advantage of the wisdom of online crowds to improve the quality. In general, crowdsourced entity alignment involves the following steps: (1) creating questions (a.k.a. microtasks) from candidate entity pairs; (2) publishing the questions to a crowdsourcing platform; (3) collecting and aggregating the answers from online workers; (4) rewarding the involved workers. Not surprisingly, the wisdom of the crowd may outperform automatic algorithms in some cases especially for records that are not explicitly identical to refer to the same real-world entity.

There are some challenges to utilize crowdsourcing. Firstly, online workers are not free, and we must pay for their labour. Entity alignment tasks usually ask

many pairs of records for the workers to check. This will involve huge monetary costs, especially for large datasets. Therefore, question selection – how to select the most beneficial questions from the candidate entity pairs for crowdsourcing – is a challenging problem. Secondly, online workers are error-prone, and the answers from them are not 100% correct. To improve the alignment quality, we should assign each task to multiple workers and aggregate the answers from these workers to infer the final answer. So another challenging problem is question assignment, which decides how to assign tasks to appropriate workers so that the alignment performance is maximized.

To address these two problems, in this work, we propose a decision theory based method. To reduce monetary costs, we adopt a partial order based approach to describe the inference power of a candidate pair: if a candidate pair is labeled to be aligned, the pairs preceding this pair based on the partial order are inferred as identical; if a candidate pair is labeled to be not aligned, the pairs succeeding this pair are inferred as different ones. Each candidate pair has its inference power according to the expectation of its inferred pairs and we choose the pairs with larger inference power as candidate microtasks. Then, we propose two budget constraints. The unlimited budget constraint balances the cost and the inference power, and selects questions from candidate microtasks whose marginal gain is larger than a predefined threshold. The bounded budget constraint selects questions to maximize the inference power when the total cost is no more than a given budget. We define two problems under the two constraints: question selection and question assignment. To solve the problems, we devise a utility function for each candidate pair to measure its incremental inference power with different number of workers involved and build influence diagram to find the choices with maximal incremental inference power. Different from the existing works, our utility function focus on the inference power rather than the matching quality. We show in this paper that our method not only finds more aligned pairs with little costs but also improves the alignment quality. We propose two efficient algorithms to solve the two problems.

To summarize, we make the following contributions. (1) We define the crowdsourced entity alignment problem based on the partial order, and propose unlimited and bounded constraints to describe the cases on how to select the questions and workers in crowdsourced entity alignment. (2) We formulate the decision-making problems under the two constraints, and build influence diagram to solve the problems by inferring on the diagram. (3) We devise a utility function to measure the feasibility of a decision. We prove the monotonicity of the utility and propose efficient algorithms to tackle the problems based on the utility function. (4) We prove that the decision-making problems under budget constraint is NP-hard, and provide a greedy algorithm to solve the problem. (5) We conduct experiments to validate the efficiency and effectiveness of our proposed algorithms on both synthetic and real data. The results show that the proposed method can achieve better performance with smaller costs than any other baseline methods.

The rest of this paper is arranged as follows. We describe the data model and the problem definition in Sect. 2. In Sect. 3, we introduce the inference method based on decision theory and formulate the optimization problem. Section 4 presents two efficient algorithms to solve the problems. We report experimental results in Sect. 5. In Sect. 6, we review the related works and Sect. 7 concludes the paper.

## 2 Data Model and Problem Definition

### 2.1 Alignment Model

Given two datasets, entity alignment finds the pairs of entities from different datasets that refer to the same real-world entity. Crowdsourced Entity alignment asks the crowd to address this problem. The goal of our work is to study the task assignment problems on aligning the candidate pairs. In our setting, some pairs will be labeled by the crowd, while others will be deduced using partial order [23]. For any entity alignment problem, we can define a partial order set  $(P, \prec)$  on the entity pairs set  $P$  with a partial order  $\prec$  and  $p_{ij} \in P$ . We define  $pp^k$  as the  $k$ -th property pair of  $p_{ij}$ , and  $s_{ij}^k$  to denote the similarity score on  $pp^k$  ( $s_{ij}^k$  can be calculated by any similarity function). The partial order  $\prec$  can be defined as: Given two pairs  $p_{ij} = (e_i, e_j), p_{i'j'} = (e_{i'}, e_{j'})$ . For each  $pp^k$ , if  $s_{ij}^k \geq s_{i'j'}^k$  and  $\sum_k(s_{ij}^k) > \sum_k(s_{i'j'}^k)$ , then  $p_{ij} \succ p_{i'j'}$ . We say  $p_{ij}$  and  $p_{i'j'}$  are comparable and  $p_{ij}$  precedes  $p_{i'j'}$  or  $p_{i'j'}$  succeeds  $p_{ij}$ . Intuitively, for a pair of entities  $p_{12} = (e_1, e_2) \in P$ , if they refer to the same entity ( $e_1 = e_2$ ), then for any other pair of entities  $p_{34} = (e_3, e_4)$ , if  $p_{34} \succ p_{12}$ , we can infer that the pairs in  $p_{34}$  are identical with high confidence. If  $p_{12}$  refers to different objects (e.g.  $e_1 \neq e_2$ ), then for any other pair of entities  $p_{34}$ , if  $p_{34} \prec p_{12}$ , the pairs in  $p_{34}$  can be inferred as different entities. Based on this approach, we find that each entity pair has its inference power which can be defined as the number of the inferred pairs, and entity pairs with larger inference power should be asked first to save the cost. Since the crowd is not free and we need to pay each worker for labeling a pair, the pairs should be deduced as many as possible to minimize the number of crowdsourced pairs. Based on this idea, we define our problem as below.

**Definition 1** (*Crowdsourced Entity Alignment*). *Given two datasets  $(D, D')$ , it selects a set of questions generated from  $D$  and  $D'$  to ask the online crowd to maximize the number of aligned pairs based on partial order, where each question contains a pair of entities and it asks the crowd to identify whether the two entities refer to the same one.*

### 2.2 Crowdsourcing Model

There are many crowdsourcing platforms, such as AMT and CrowdFlower, which provide facilities for asking online workers to complete the alignment tasks. Since workers may provide wrong answers, we must take the worker quality into consideration, and assign each microtask to multiple workers (i.e., worker redundancy)

to derive a more accurate answer. For ease of presentation, each microtask here is a single choice question (options: YES/NO). The strategy on how to aggregate the distributed opinions of workers is Majority Voting (MV). Note that other voting scheme, such as Half Voting, Weighted MV, or Bayesian voting, can also be easily integrated into our model. In our crowdsourcing model, there are:

- A requestor who publishes a set of entity alignment microtasks  $\mathcal{T} = \{t_1, t_2, \dots, t_n\}$ , each  $t_i$  is associated with a price  $b_i$ .
- Each task associates with an answer  $S_i \in \{0, 1\}$  to denote whether they are identical which is unobserved beforehand.
- Tasks are assigned to workers  $\mathcal{W} = \{w_1, w_2, \dots, w_m\}$ , and each worker  $w_j$  has a quality  $q_j$ .  $\mathcal{K} = \{k_1, k_2, \dots, k_n\}$  denotes the worker redundancy of each task in  $\mathcal{T}$  (the number of workers assigned for the task).  $q_j$  is the probability that worker  $q_j$  correctly answers the entity alignment questions. It can be obtained by the qualification test, provided by most of crowdsourcing platforms. It asks workers to answer some tasks with known answers before they can answer the real questions. We assume that the worker qualities  $\{q_j\}_{j \in [1, \dots, n]}$  are independent and identically distributed (i.i.d.) random variables with a distribution on  $[0, 1]$ .
- Click  $C_{ij}$  is defined as an answer on task  $t_i$  from worker  $w_j$  where  $C_{ij} \in \{0, 1\}$ .

### 2.3 Decision-Making Problems

**Budget Constraint.** In crowdsourced entity alignment tasks, there are usually large amount of candidate pairs to be verified by the workers, and each pair will be assigned to multiple workers (i.e., worker redundancy) to improve the quality. It is infeasible to publish all pairs for verification to the crowdsourcing platform due to the high cost, nor for the unlimited worker redundancies for each task. In order to find the most beneficial questions in the crowdsourced entity alignment, a crucial issue is the definition of the budget constraint. In our settings, we define two budget constraints. The first one only considers the trade-off between cost and inference power, named unlimited budget constraint. Intuitively, as we always choose pairs with maximal inference power as questions, the increment of inference power is slow down with the increase of the cost. That is to say, the marginal gain is reduced. Then the unlimited constraint can be defined as:

**Definition 2** (*Unlimited Budget Constraint*). *While selecting questions from all the candidate pairs, only the pairs whose marginal gain is larger than a predefined threshold  $\tau$  can be selected as questions.*

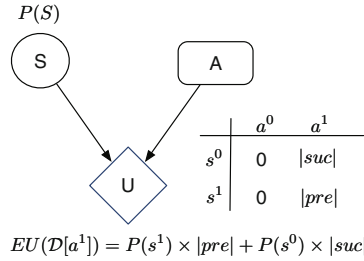
The second one is based on a given bounded budget, named bounded budget constraint. That is, we choose pairs with maximal inference power as microtasks and assign them to the coming workers until the total cost exceeds the given budget. The bounded constraint can be defined as:

**Definition 3** (*Bounded Budget Constraint*). *While selecting questions from all the candidate pairs, and each is associated with a price, it selects questions to*

ask the crowd to maximize the inference power when the total price is no more than a given budget  $\mathcal{B}$ .

**Decision-Making Problems.** Given certain budget constraint, there are two decision-making problems to be addressed in the microtask assignment problem. One is to decide whether a candidate pair should be chosen as questions among all the candidate pairs (i.e., how to select a subset  $\mathcal{T}_h \subset \mathcal{T}$  which have the maximal inference power under given constraint). Another is to decide whether a worker should be chosen to answer a question (i.e., how to choose worker redundancies  $k_i$  for  $t_i$  in order to maximize the inference power under given constraint).

Existing crowdsourcing platform has limitations on task assignment that we can not obtain the worker qualities  $\{q_j\}$  upfront and it is not applicable to assign microtasks to specific worker. To address this challenge, one can design specialized interface to get the worker qualities by communicating with crowdsourcing platform through designated APIs, such as ICROWD [5]. While in this work, we use qualification test to overcome this difficulty. Before each task assignment, we predefine a threshold (e.g. 80%) for the qualification test. Then the quality of workers who passed qualification test will be deemed as 80% for this type of microtask. By this way, we can get the worker qualities  $\hat{q} = 80\%$  before generate real questions. Since the specified threshold is the lower bound of the worker quality, it ensures the alignment quality, meanwhile simplifies the inference process which will be discussed in the next section.



**Fig. 1.** Influence diagram  $\mathcal{I}_s$  of a microtask  $t_i$  for the simple decision-making situation.

## 3 Inference Method

### 3.1 Decision Theory

In a decision-making problem, a decision maker (called agent, referring to the task requestor in our crowdsourced framework) has a set of possible actions that she might pick. Each action may lead to different outcomes, which the agent prefers to different degrees. A simple decision-making situation  $\mathcal{D}$  can be defined as follows [13]:

- A set of possible actions that the agent can take,  $\mathcal{A} = \{a_1, \dots, a_k\}$ ;
- A set of states of the world,  $\mathcal{X} = \{x_1, \dots, x_l\}$ ;
- A probability distribution  $P(\mathcal{X}|\mathcal{A})$ , where  $P(X|a)$  specifies the distribution over state  $X$  given that the action  $a \in \mathcal{A}$  was taken;
- A utility function  $\mathcal{U}(\mathcal{X}, \mathcal{A})$ , where  $\mathcal{U}(X, a)$  specifies the agent's preferences over the joint distribution of state  $x$  and action  $a$ .

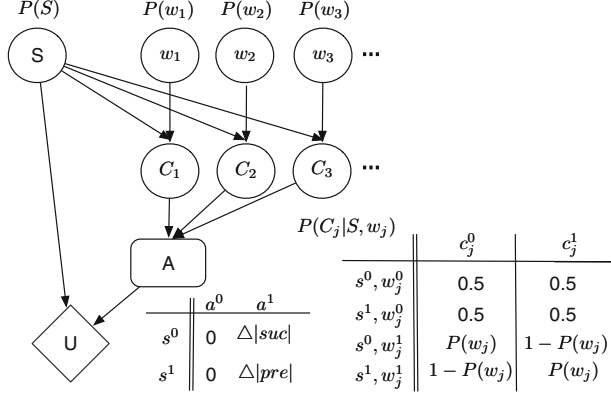
Then the expected utility is:  $EU[D[a]] = \sum_{x_1, x_2, \dots, x_l} P(X_i|a)U(X_i, a)$ . In a decision-making situation  $\mathcal{D}$ , we want to choose the action that maximizes the expected utility:  $a^* = \operatorname{argmax}_a EU[D[a]]$ .

Here, we use influence diagram to represent the different situations that might be encountered by the agent in the context of our decision problems. Influence diagram is a natural extension of the Bayesian network, it uses a directed acyclic graph which containing three types of nodes corresponding to chance variables (i.e.,  $\mathcal{X}$ ), decision variables (i.e.,  $\mathcal{A}$ ), and utility variables (i.e.,  $\mathcal{U}$ ). These different node types are represented as ovals, rectangles, and diamonds, respectively.

For example, Fig. 1 shows the simplest influence diagram  $\mathcal{I}_s$  of the decision situations in an entity alignment problem. Given an entity alignment problem, we first filter the candidate pairs by automatic algorithms to reduce the matching scale. Then, we build partial order to calculate the number of inference pairs of each candidate pair. The chance variable  $S$  is a binary random variable refers to whether the candidate pair is identical ( $s^1$ ) or not ( $s^0$ ).  $P(S)$  is the prior distribution of  $S$  ( $S \in \mathcal{X}$ ) which can be obtained by automatic algorithms beforehand, default is 0.5.  $A$  is the decision variable represents the action whether to chose this candidate pair as microtask ( $a^1$ ) or not ( $a^0$ ). The utility variable  $U$  encodes the utility of the inference power of this candidate pair, which is determined by the utility variable's parents node (i.e.,  $S$  and  $A$ ). It is a real-valued function for each combination of the parent nodes:  $U : Val(S) \times Val(A) \rightarrow \mathbb{R}$ . The table in the right part of Fig. 1 shows the function in our settings. Here  $|suc|$  means the number of inference pairs when  $s^0$ , and  $|pre|$  means the number of inference pairs when  $s^1$ . Obviously  $a^* = a^1$  will maximize the expected utility, and the expected utility is shown in the bottom of the figure. We find that the expected utility we obtain from  $\mathcal{I}_s$  is just the expectation of the inference power of an entity pair. We should choose the candidate pairs with larger expected utility as microtasks.

### 3.2 Inference on Decision Model

In our entity alignment models, it is more complex than what is shown in Fig. 1. As mentioned in previous section, we have to take the quality and choice of workers into account, and calculate not only the subset of candidate pairs, but also the redundancies of workers in order to maximize the expected utility under given constraint. Therefore, suppose a candidate pair was published as a micro-task on the crowdsourcing platform, we add different workers and their choices into the influence diagram  $\mathcal{I}_s$ , and obtain a complex influence diagram  $\mathcal{I}_c$  which



**Fig. 2.** Influence diagram  $\mathcal{I}_c$  of a microtask  $t_i$  for the complex decision-making situation.

is shown in Fig. 2. Then, we can address the two decision-making problems by inferring on the influence diagram.

For any candidate pair, when it is published as microtask  $t_i$  on the crowdsourcing platform, it will be examined by a set of workers  $\{w_j\}$ , and each of them will perform a click  $C_j$  to express the fact whether the candidate pair is identical ( $c_j^1$ ) or not ( $c_j^0$ ) from her perspective. For each click  $C_j$  of  $w_j$ , it is determined not only by the quality distribution of  $w_j$ , but also by the prior distribution of  $S$ . Here  $w_j$  is also a binary chance variable refers to whether the worker can give a right answer ( $w_j^1$ ) or not ( $w_j^0$ ).  $P(w_j)$  can be obtained by the qualification test, and  $P(S)$  can be obtained by automatic algorithm. For the conditional probability distribution of  $C_j$ , we assume low quality workers deciding randomly to our published questions (i.e.,  $P(C_j|S, w_0) = 0.5$ ). When high quality workers come, the probability of giving a right answer is equal to the worker quality  $P(w_j)$ , and  $1 - P(w_j)$  for the wrong answer. This assumption is reasonable when we leave out the malicious workers or spammers into account. The conditional probability table (CPT) of click  $C_j$  is shown in the bottom right of Fig. 2. Since we focus on the increment of the expected utility (i.e., the changes of the inference power under different decisions), the utility function is a little different from what is shown in  $\mathcal{I}_s$ . We use  $\Delta|suc|$  and  $\Delta|pre|$  to present the increment of inference power.  $\Delta|suc| = |suc| - EU[D[a^*]]$  and  $\Delta|pre| = |suc| - EU[D[a^*]]$ .

Given  $\mathcal{I}_c$  and the probability distributions discussed above, we can obtain the expected utility under different choice of workers. We use  $Pa_{X_i}$ ,  $Pa_A$ , and  $Pa_U$  as a shorthand to denote the parents of chance variable  $X_i$ , decision variable  $A$ , and utility variable  $U$  respectively, and  $W = \{X_i\} - C_j$ . Note that the decision rule  $\delta_A$  in the decision network is also a conditional probability distribution (CPD), and all of the variables in our network have a CPD associated with them. Then, we have effectively defined a joint probability distribution over all the variables in the network. Although utilities are not just ordinal values of

probability, but they still can be deemed as a factor in Bayesian network, and we can probabilistically aggregate utilities and compute their expectations over the different possible states. Similar to the chain rule of Bayesian network, we can define the expected utility  $EU[D[\delta_A]]$  of  $I_c$  as follows:

$$\begin{aligned}
EU[D[\delta_A]] &= \sum_{x_1, x_2, \dots, x_l, a} P_{\delta_A}(X_i, a) U(X_i, a) \\
&= \sum_{x_1, x_2, \dots, x_l, A} \left( \left( \prod_i P(X_i | Pa_{X_i}) \right) U(Pa_U) \delta_A(A | Pa_A) \right) \\
&= \sum_{C_j, A} \delta_A(A | C_j) \sum_W \left( \left( \prod_i P(X_i | Pa_{X_i}) \right) U(S, A) \right) \\
&= \sum_{C_j, A} \delta_A(A | C_j) \mu(A, C_j) \tag{1}
\end{aligned}$$

$$\delta_A^*(a | c_j) = \begin{cases} 1 & a = \operatorname{argmax}_A \mu(A, c_j) \\ 0 & \text{otherwise} \end{cases}$$

The algorithms to solve the inference on Bayesian network can also be used to solve our problems. The most commonly-used algorithm is called variable elimination. Generally speaking, variable elimination has exponential time and space complexity in the worst case and the problem is as hard as that of computing the number of satisfying assignments for a propositional logic formula. This means that it is  $\#P$ -hard (i.e., strictly harder than NP-complete problems). Fortunately, the number of workers chosen to answer the question is always bounded within a dozen. Therefore, the complexity of  $EU[D[\delta_A]]$  can be deemed as a constant.

### 3.3 Optimization Problem Formalization

Next, we give the formal representation of the optimizing decision-making problem under given constraint. Our objective is to decide how to choose  $\mathcal{T}_h$ , and worker redundancies  $\{k_i\}$  to maximize the overall expected utility. In order to give a clear answer by the voting strategy, we assume the size of workers for each question is odd as tradition [2, 4]. Suppose the current worker redundancy of a task  $t_i$  is  $k$ , then the next number of worker redundancy is  $k + 2$ . The increment expected utility between two choices is:

$$\Delta EU[D[\delta_A(k)]] = \begin{cases} EU[D[\delta_A(k)]] - EU[D[\delta_A(k-2)]], & k > 1, k \text{ is odd}; \\ EU[D[\delta_A(1)]] - EU[D[\delta_A(0)]], & k = 1; \\ EU[D[\delta_A(0)]], & k = 0. \end{cases} \tag{2}$$

The marginal gain of  $t_i$  under certain worker redundancy  $k$  (denoted by  $\rho_i^k$ ) is a key concept to the decision-making problem under unlimited budget constraint which can be defined as  $\rho_i^k = \frac{\Delta EU[D[\delta_A(k)]]}{EU[D[a^*]]}$ . Here  $EU[D[a^*]]$  is the expected utility from  $\mathcal{I}_s$  which means the expected inference power of a candidate pair.



Then, for unlimited constraint, given the threshold  $\tau$ , the optimizing decision-making problem can be defined as:

$$\begin{aligned}
 & \text{maximize} && \sum_{i=1}^n EU_{t_i}[D[\delta_A(k_i)]]y_i \\
 & \text{s.t.} && \rho_i^k \geq \tau, \quad k = 0 \text{ or } k \text{ is odd.} \\
 & && y_i \in \{0, 1\}, \quad i = 1, 2, \dots, n.
 \end{aligned} \tag{3}$$

For bounded constraint, given a predefined budget  $\mathcal{B}$ , the optimizing decision-making problem can be defined as:

$$\begin{aligned}
 & \text{maximize} && \sum_{i=1}^n EU_{t_i}[D[\delta_A(k_i)]]y_i \\
 & \text{s.t.} && \sum_{i=1}^n k_i b_i \leq \mathcal{B} \\
 & && y_i \in \{0, 1\}, \quad i = 1, 2, \dots, n.
 \end{aligned} \tag{4}$$

## 4 Assignment Algorithms

### 4.1 Monotonicity

The increment expected utility  $\triangle EU[D[\delta_A(k)]]$  satisfies monotonicity as stated in Theorem 1, which is used in our task assignment algorithms<sup>1</sup>. Notice that Theorem 1 is also applicable under the case when we know the different worker qualities in advance (e.g., ICROWD [5]) and add workers in a descending order according to their qualities. Since our task assignment algorithms, on unlimited or bounded constraint, are based on Theorem 1, they can also be used to solve the task assignment problems with different worker qualities for these specially developed systems.

**Theorem 1.**  $\triangle EU[D[\delta_A(k)]] \geq 0$ , and  $\triangle EU[D[\delta_A(k)]]$  decreases monotonically with  $k$  until it is equal to 0 given a fixed value of worker quality  $\hat{q}$ .

### 4.2 Task Assignment on Unlimited Budget Constraint

For unlimited constraint, the goal is to maximize total expected utility, such that,  $\rho_i^k$  is greater than a specified threshold for each task  $t_i$ . The algorithm of unlimited constraint (TAUC) is shown in Algorithm 2 which takes two datasets  $D$  and  $D'$  and several parameters as input, and output a subset  $\mathcal{T}_h \subset \mathcal{T}$  and the worker redundancies  $k_i$  for each  $t_i \in \mathcal{T}_h$ . We first build candidate microtasks set  $\mathcal{T}$  based on the partial order (line 2). The details of this algorithm can be found in [23]. Then, we compute the expected utility for each candidate pairs and put it into a priority queue (lines 3–5). Since  $\triangle EU[D[\delta_A(k)]] \geq 0$ , and it decreases monotonically with  $k$  according to Theorem 1, for each task to be published, we increase the number of workers until  $\rho_i^k < \tau$  (lines 6–14). At last, for the

<sup>1</sup> The proof can be found in our technical report <http://dbgroup.cs.tsinghua.edu.cn/lgl/crowd2.pdf>.

**Algorithm 1.** TASK ASSIGNMENT ON UNLIMITED CONSTRAINT

**Input:** Two datasets:  $D$  and  $D'$ ; the marginal gain threshold:  $\tau$ ; the worker quality:  $\hat{q}$ .

**Output:**  $\mathcal{T}_h$  and  $\mathcal{K}$ .

---

```

1 begin
2   Calculate candidate microtask set  $\mathcal{T}$  from  $D$  and  $D'$  based on partial order;
3   for each  $t_i \in \mathcal{T}$  do
4     compute  $EU_{t_i}[D[a^*]]$ ;  $k_i=0$ ;
5      $Q.push(t_i)$ ; //  $Q$  is a priority queue.
6   for each  $t_i \in Q$  do
7      $t_i = Q.pop()$ ;
8     repeat
9       if  $k_i = 0$  then  $\Delta k = 1$ ;
10      else  $\Delta k = 2$ ;
11      compute  $\Delta EU_{t_i}[D[\delta_A(k)]]$ ;
12       $\rho_i^k = \frac{\Delta EU_{t_i}[D[\delta_A(k)]]}{EU_{t_i}[D[a^*]]}$ ;
13      if  $\rho_i^k \geq \tau$  then  $k_i = k_i + \Delta k$ ;
14    until  $\rho_i^k < \tau$ ;
15    if  $k_i \geq 1$  then  $\mathcal{T}_h.add(t_i)$ ;  $\mathcal{K}.add(k_i)$ ;

```

---

tasks whose  $k_i \geq 1$ , we put it into  $\mathcal{T}_h$  and record their worker redundancies  $k_i$  (line 15).

The key point of this algorithm is the calculation of  $\Delta EU[D[\delta_A(k)]]$  which is exponential to the number of workers. The time complexity for the calculation of each task is  $\mathcal{O}(m2^{m-1})$ . Updating  $Q$  costs  $\mathcal{O}(\log^n)$  time, and there are  $n$  tasks, hence the time complexity of this part is  $\mathcal{O}(n(m2^{m-1} + \log^n))$ . As mentioned above, the number of workers is always bounded within a dozen, and the exponential part can be deemed as a constant. Therefore, the time complexity is  $\mathcal{O}(n \log^n)$ .

### 4.3 Task Assignment on Bounded Budget Constraint

For bounded constraint, our goal is to maximize  $\sum_{i=1}^n EU_{t_i}[D[\delta_A(k_i)]]$ , such that, the summation of the cost of each task is less than a specified total budget  $\mathcal{B}$ . We first prove that the task assignment problem on bounded constraint is NP-hard, and then propose a greedy algorithm to solve it effectively.

**Theorem 2.** *The task assignment problem on bounded constraint is NP-hard.*

*Proof* (Sketch). We first proof when considering the increment expected utility, the Eq. 4 can be rewritten as:

$$\begin{aligned}
 & \text{maximize} && \sum_{i=1}^n \sum_{k=0}^{\mathcal{B}/2b_i} \Delta EU_{t_i}[D[\delta_A(k)]] y_{ik} \\
 & \text{s.t.} && \sum_{i=1}^n \sum_{k=0}^{\mathcal{B}/2b_i} r_{ik} y_{ik} \leq \mathcal{B} \\
 & && y_{ik} \in \{0, 1\}, \quad i = 1, 2, \dots, n.
 \end{aligned} \tag{5}$$

where

$$r_{ik} = \begin{cases} b_i, & k = 0; \\ 2b_i, & \text{otherwise.} \end{cases}$$

---

**Algorithm 2.** TASK ASSIGNMENT ON BOUNDED CONSTRAINT
 

---

**Input:** Two datasets:  $D$  and  $D'$ ; budget:  $\mathcal{B}$ ; price of a question:  $\{b_i\}$ ;  
worker quality:  $\hat{q}$ .

**Output:**  $\mathcal{T}_h$  and  $\mathcal{K}$ .

```

1 begin
2   Calculate candidate microtask set  $\mathcal{T}$  from  $D$  and  $D'$  based on partial
   order;
3    $\delta = \mathcal{B}$ ;
4   for each  $t_i \in \mathcal{T}$  do
5     compute  $EU_{t_i}[D[a^*]]$ ;  $k_i=0$ ;
6     Q.push( $t_i$ ); // Q is a priority queue.
7   while  $\delta > 0$  and !Q.isEmpty() do
8      $t_i = \text{Q.pop}()$ ;
9     if  $k_i = 0$  then  $\Delta k = 1$ ;
10    else  $\Delta k = 2$ ;
11    if  $\Delta k \times b_i < \delta$  then
12       $k_i = k_i + \Delta k$ ;  $\delta = \delta - \Delta k \times b_i$ ;
13      compute  $\Delta EU_{t_i}[D[\delta_A(k)]]$ ;
14      Q.push( $t_i$ );
15    if  $k_i \geq 1$  then  $\mathcal{T}_h.add(t_i)$ ;  $\mathcal{K}.add(k_i)$ ;

```

---

For each task, there are at most  $\mathcal{B}/2b_i$  workers can be added to the task ( $\mathcal{B}/2b_i \geq k_i$ ). According to Theorem 1,  $\Delta EU[D[\delta_A(k)]]$  decreases monotonically with  $k$  in each task. Therefore, if we choose these candidate pairs as tasks, we can only add the workers with larger increment expected utility (i.e., for every  $k \in (k_i, \mathcal{B}/2b_i]$ ,  $y_{ik} = 0$ ). According to Eq. 2, we have:  $\sum_{k=0}^{\mathcal{B}/2b_i} \Delta EU_{t_i}[D[\delta_A(k)]] = \sum_{k=0}^{k_i} \Delta EU_{t_i}[D[\delta_A(k)]] = EU[D[\delta_A(k_i)]] - EU[D[\delta_A(k_i - 2)]] + \dots + EU[D[\delta_A(0)]] = EU_{t_i}[D[\delta_A(k_i)]]$ . Similarly,  $\sum_{k=0}^{\mathcal{B}/2b_i} r_{ik} y_{ik} = k_i b_i$ , Eqs. 4 and 5 are equivalence. It is easy to prove that the optimization problem of Eq. 5 is NP-hard by a reduction from the 0-1 Knapsack Problem, in which the

$\sum_{k=0}^{\mathcal{B}/2b_i} \Delta EU_{t_i}[D[\delta_A(k)]]$  is the profit, the  $\sum_{k=0}^{\mathcal{B}/2b_i} r_{ik}y_{ik}$  is the weight, and the budget  $\mathcal{B}$  is the capacity of the knapsack.

We follow the algorithm developed to solve the 0–1 knapsack problem from [12] and also propose a greedy algorithm with 0.5-approximation ratio. The algorithm on bounded constraint (TABC) is shown in Algorithm 2. After the generation of candidate microtasks set (line 2), we initialize the budget,  $k_i$ , and  $Q$  (lines 3–6). Next, in each iteration, we add workers to the task with the largest increment expected utility until the budget is exhausted or  $Q$  is empty (lines 7–14). For the tasks whose  $k_i \geq 1$ , we put it into  $\mathcal{T}_h$  and record their worker redundancies  $k_i$  as the final answer (line 15). The time complexity of TABC is similar to TAUC. The difference is the calculation of  $\Delta EU[D[\delta_A(k)]]$  will at most repeat  $\mathcal{B}$  times and  $Q$  will be updated in each iteration. Hence the time complexity of TABC is  $\mathcal{O}((\mathcal{B} + n)\log^n)$ .

## 5 Experiments

### 5.1 Experiment Setup

**Datasets:** We choose the IIMB benchmark of OAEI<sup>2</sup> and the large-scale *Yago-DBPedia* dataset<sup>3</sup>. The IIMB benchmark provides *OWL/RDF* data about films, actors, and locations. *YAGO* and *DBPedia* are two famous large-scale *KBs* with a rich schema structure. Both *YAGO* [14] and *DBPedia* [8] are available as lists of triples from their respective websites. Table 1 presents the statistics information of the two datasets.

Table 1. Datasets

Dataset	# Instance	# Classes	# Properties
YAGO	3.03 M	360 K	70
DBPedia	2.49 M	0.32 K	1.2 K
IIMB	12.6 K	0.2 K	24

**Competitors.** We consider two heuristic worker assignment methods as baselines. The first one is the average strategy which assigns each task the same number of workers. We vary the number of workers (e.g., 3, 5, 7, ... and denoted by Ave(3), Ave(5), Ave(7), ...) for each task as comparisons. The second one is the maximal strategy (denoted by Max) which assign the tasks in a descending order according to their expected utility, and assign each task as many as possible workers until the marginal gain is less than the predefined threshold before moving to the next task. For the bounded constraint, the stop criterion of the Max strategy is when the given budget is exhausted. For different methods, we

<sup>2</sup> <http://oaei.ontologymatching.org/>.

<sup>3</sup> The dataset comes from <http://webdam.inria.fr/paris> with ground truth.

compare the alignment quality, the number of generated questions, and the number of alignment pairs. We evaluate the alignment quality using the standard metrics of precision, recall, and F1-score.

**Crowdsourcing Simulation.** The task assignment algorithm can be executed before the generated questions published to the crowdsourcing platform. To test the alignment result, we should obtain and aggregate the returned answers from the platform. Because the ground truth is known, we can simulate to generate workers with quality in 70%, 80%, and 90% respectively. We repeat each experiment 10 times and record the average result. By this way, we can conduct extensive experiments to test the effectiveness of our proposed methods, and reduce the impact of the randomness of the real workers.

## 5.2 Synthetic Data

To evaluate the performance of the proposed algorithms, in this section we use synthetic datasets to simulate the process of the task assignment and evaluate the number of generated questions and alignment quality by varying different parameters. Alignment results are investigated on unlimited constraint and bounded constraint respectively.

**Unlimited Constraint.** The experiments are conducted under different number of candidate microtasks  $n$  (i.e., the number of  $\mathcal{T}$ ) which are sorted by the expected inference power of the pairs in a descending order.  $n$  varies from 50 to 250 as the value of the inference power is too small to impact the alignment result after 250. In practice, we can use question selection algorithms [23] without worker involved to select the candidate microtasks according to their expected inference power as demonstrated in Algorithm 1.

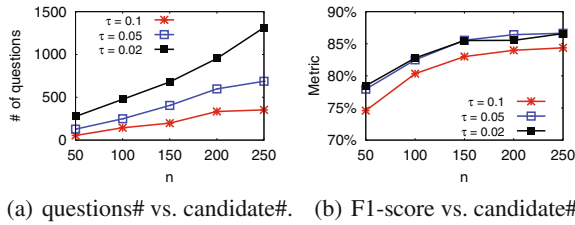


Fig. 3. Parameter tuning on unlimited constraint (synthetic data).

**Effect of the Marginal gain Threshold  $\tau$ .** Figure 3(a) shows the number of generated questions under different number of candidate tasks  $n$  with the marginal gain threshold  $\tau$  which is equal to 0.1, 0.05 and 0.02 respectively. For all the value of  $\tau$ , the number of questions increases with  $n$ . For all the values of  $n$ , the number of questions decreases with  $\tau$ , and the increase rate also drops with  $\tau$ . This is because larger  $\tau$  will stop adding workers to each task earlier, and

with the increase of  $n$  the effect becomes larger. Hence, to save cost we should adopt larger  $\tau$ . Figure 3(b) shows the alignment quality under different number of  $n$ . No surprise, the total quality increases with  $n$ . We can see in the figure that  $\tau = 0.02$  and  $\tau = 0.05$  have the competitive quality for all the values of  $n$ , while the alignment quality of  $\tau = 0.1$  is significantly lower than them. Considering the trade-off between quality and cost, in our settings  $\tau = 0.05$  is a better choice, and we adopt  $\tau = 0.05$  in the following experiments on the synthetic data.

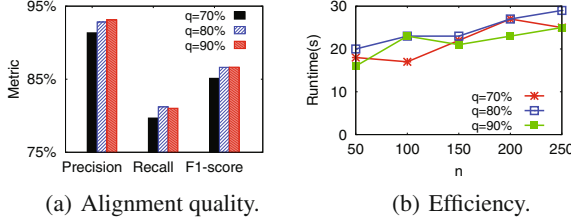


Fig. 4. Alignment comparison by varying worker quality (synthetic data).

**Effect of Worker Qualities.** Figure 4(a) shows the comparison of alignment quality of the TAUC with different worker qualities when  $n = 250$ . We can find from the figure that the increment of worker quality from 80% to 90% does not improve the performance visibly. This shows that our algorithm does not require very high quality workers. Figure 4(b) shows the runtime of TAUC under different values of  $n$ . Here, we exclude the runtime of partial order construction and only evaluate the efficiency of TAUC. The result presents certain randomness while the overall trend is increasing gently, which shows the good efficiency of TAUC. This proves that the exponential complexity of the inference on Bayesian network makes little effect on our algorithm due to the limited number of workers.

**Bounded Constraint.** The experiments on bounded constraint are conducted under different numbers of predefined budget. We compare the alignment results under different budgets to show the effectiveness of the proposed TABC algorithm with baseline methods. As mentioned above, the default threshold  $\tau$  for Max algorithm is 0.05, and the default worker quality is 80%. Budget varies from 100 to 900 because we want to evaluate the performance under a rather limited budget to a relative high budget. To ease of evaluation, we assume the price of each question consumes unit budget (i.e.,  $b_i = 1$ ) in our experiments. Notice both TABC and other baseline methods can support the settings of different  $b_i$ . The alignment result is shown in Fig. 5. Figure 5(a) shows that the number of alignment pairs increases with the budget  $\mathcal{B}$ , and the increase rate drops with  $\mathcal{B}$ . This is because the inference power becomes lower with the increment of the questions. There is a large gap among all the five methods when  $\mathcal{B} = 100$ . This is because when the budget is very limited, a better assignment algorithm will embody its power more obviously. With the increase of  $\mathcal{B}$ , the gap becomes smaller and smaller, while our TABC outperforms all the other methods before  $\mathcal{B} = 900$ . This shows that TABC

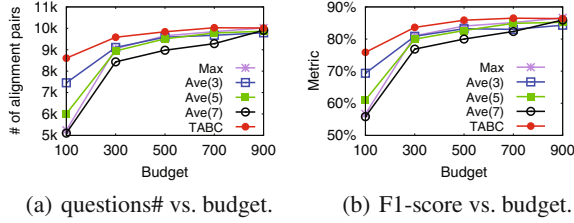


Fig. 5. Alignment comparison on bounded constraint (synthetic data).

is especially suitable for the case of limited budget. The number of alignment pairs of Max algorithm is small at the beginning and a little larger than other methods when  $\mathcal{B} = 900$ . This is because although Max can obtain the maximal expected utility for each task but it can only select few tasks under limited budget. The alignment quality which is shown in Fig. 5(b) present the same regularity with Fig. 5(a) because the number of alignment pairs can reflect the alignment quality to some extent.

### 5.3 Real Data

To evaluate the proposed algorithms without bias, the datasets we use in this section are two real world knowledge base, *YAGO* and *DBPedia*. Since there are several millions of entities in the datasets, this experiment is a time-consuming process. Therefore, we only evaluate the performance with 80% worker quality. Alignment results are also investigated on unlimited constraint and bounded constraint respectively.

**Unlimited Constraint.** As stated in the synthetic data experiment, we first evaluate the value of marginal gain threshold  $\tau$  in this dataset. The experiments are conducted under candidate microtasks  $n = 500$ . This is because there are so many candidate entity pairs in this dataset that it is intractable to enumerate all the candidate pairs. We only select the first 500 candidate entity pairs with the largest expected inference power as candidate tasks. Figure 6(a) shows the alignment quality under different value of  $\tau$ . We can see from the figure there is not much difference between them. While with the decrease of  $\tau$ , the number of generated questions grows very fast (increases from 1600 to 3400 with  $\tau$  drops from 0.1 to 0.02), which means much larger cost will be paid. Therefore, in the following experiments we adopt  $\tau = 0.1$ . Figure 6(b) shows the alignment quality comparison between different methods under  $n = 500$ . Our TAUC algorithm achieves similar performance as Ave(7) which is much better than other two baseline methods, while the number of questions needed is 2.2 times lower than Ave(7). This means our method can assign workers more effectively than the average strategy.

**Bounded Constraint.** The settings of the experiment on bounded constraint are similar to that of the synthetic data. The default threshold  $\tau$  for Max

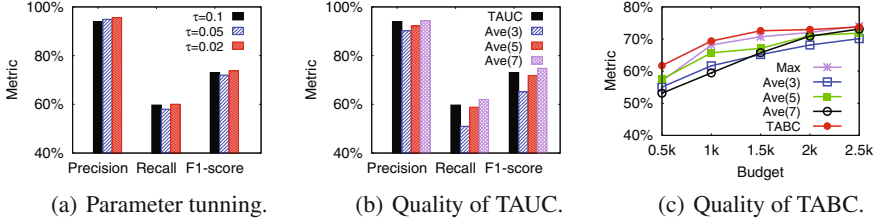


Fig. 6. Alignment comparison on real data.

algorithm is 0.1, and the default worker quality is 80%. The alignment result is also similar which is shown in Fig. 6(c). The alignment quality increases with the budget  $\mathcal{B}$ , and the increase rate drops with  $\mathcal{B}$ . TABC algorithm outperforms other baseline methods especially when budget is limited. The difference between Figs. 6(c) and 5(b) is that (1) the gap among all the five methods at the beginning is not too large because the initial budget is larger than that of Fig. 5(b), and (2) the quality of Ave(3) is lower than Ave(5) when  $\mathcal{B} = 500$  because the real world datasets are more complex than the synthetic one and only assigning three workers to a task can not achieve satisfactory alignment quality.

## 6 Related Work

Entity alignment (a.k.a. entity reconciliation/resolution) has been studied extensively in these years (see [6] for a survey). Some existing works have investigated how to use active learning in this problem. Arasu et al. [1] proposed a scalable active learning methods for matching large datasets and provided probabilistic guarantees on the quality of the results. Gokhale et al. [7] developed a solution using crowdsourcing to train a learning-based matcher, and used active learning to minimize crowdsourcing costs. All these methods focus on interactive methods of entity alignment whereas we consider the impact of different inference powers and worker redundancies of this problem.

Many studies focus on leveraging crowdsourcing to improve the alignment quality and reduce the cost [9–11, 21, 22]. Wang et al. [17] and Vedapunt et al. [15] studied how to utilize the transitivity to reduce the number of questions. Obviously the transitivity can reduce the cost, but it may introduce negative effects on quality. ACD [18] and GCER [19] are another two crowdsourcing ER frameworks which mostly focus on matching quality, but they achieve high quality at high monetary budget. CROWDER [16] and POWER [3] adopt a two-step framework, in which they first generated a candidate set of matching pairs by automatic methods, and then introduced crowdsourcing to check the matching results. However, these methods use average strategy to assign workers which is proved to be not efficient and economic by this work. Recently, there are a few works take worker redundancy problems into consideration. Cao et al. [2] define a Jury Selection Problem (JSP) by utilizing crowdsourcing to assign different workers to a task under a limited budget. Zheng et al. [20] go beyond [2]



and claim Bayesian Voting can optimally solve JSP. They assume the worker's qualities are known before assignment which is usually not applicable in existing systems. For this problem, Mo et al. [4] assume each microtask has a worker accuracy which can be calculated in qualification test, and propose several algorithms to solve the problem. These methods focus only on how to improve alignment quality (i.e., accuracy) by crowdsourcing while our method concentrates on finding more aligned entity pairs with high quality.

## 7 Conclusion

In this paper, we define two decision-making problems under the different budget constraints for entity alignment. We devise a utility function to measure the feasibility of an action based on decision theory. We formulate the decision-making problems as optimization problems and propose two efficient algorithms. The experiment results on synthetic and real datasets show that the proposed method can achieve better performance than baseline methods.

**Acknowledgement.** This work was supported by 973 Program of China (2015CB358700), NSF of China (61632016, 61373024, 61602488, 61422205, 61472198), FDCT/007/2016/AFJ, and Key Projects of Military Logistics Research (BHJ14L010).

## References

1. Arasu, A., Götz, M., Kaushik, R.: On active learning of record matching packages. In: SIGMOD 2010, Indianapolis, Indiana, USA, 6 June–10 June 2010, pp. 783–794 (2010)
2. Cao, C.C., She, J., Tong, Y., Chen, L.: Whom to ask? jury selection for decision making tasks on micro-blog services. PVLDB **5**(11), 1495–1506 (2012)
3. Chai, C., Li, G., Li, J., Deng, D., Feng, J.: Cost-effective crowdsourced entity resolution: a partial-order approach. In: SIGMOD 2016, San Francisco, CA, USA, 26 June–01 July 2016 (2016)
4. Mo, L., et al.: Optimizing plurality for human intelligence tasks. In: CIKM 2013, San Francisco, CA, USA, 27 October–1 November 2013, pp. 1929–1938 (2013)
5. Fan, J., Li, G., Ooi, B.C., Tan, K., Feng, J.: iCrowd: an adaptive crowdsourcing framework. In: SIGMOD, Melbourne, Victoria, Australia, 31 May–4 June 2015, pp. 1015–1030 (2015)
6. Getoor, L., Machanavajjhala, A.: Entity resolution for big data. In: KDD 2013, Chicago, IL, USA, 11 August–14 August 2013, p. 1527 (2013)
7. Gokhale, C., Das, S., Doan, A., Naughton, J.F., Rampalli, N., Shavlik, J.W., Zhu, X.: Corleone: hands-off crowdsourcing for entity matching. In: SIGMOD 2014, Snowbird, UT, USA, 22 June–27 June 2014, pp. 601–612 (2014)
8. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBPedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semant. Web **6**(2), 167–195 (2015)
9. Li, G.: Human-in-the-loop data integration. PVLDB **10**(12), 2006–2017 (2017)

10. Li, G., Chai, C., Fan, J., Weng, X., Li, J., Zheng, Y., Li, Y., Yu, X., Zhang, X., Yuan, H.: CDB: optimizing queries with crowd-based selections and joins. In: SIGMOD (2017)
11. Li, G., Wang, J., Zheng, Y., Franklin, M.J.: Crowdsourced data management: a survey. *IEEE Trans. Knowl. Data Eng.* **28**(9), 2296–2319 (2016)
12. Martello, S., Toth, P.: *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, New York (1990)
13. Russell, S.J., Norvig, P.: *Artificial Intelligence - A Modern Approach*. Pearson Education, London (2010). (3. internat. ed.)
14. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: a large ontology from wikipedia and wordnet. *J. Web Sem.* **6**(3), 203–217 (2008)
15. Vesdapunt, N., Bellare, K., Dalvi, N.N.: Crowdsourcing algorithms for entity resolution. *PVLDB* **7**(12), 1071–1082 (2014)
16. Wang, J., Kraska, T., Franklin, M.J., Feng, J.: Crowder: crowdsourcing entity resolution. *PVLDB* **5**(11), 1483–1494 (2012)
17. Wang, J., Li, G., Kraska, T., Franklin, M.J., Feng, J.: Leveraging transitive relations for crowdsourced joins. In: SIGMOD 2013, New York, NY, USA, 22 June–27 June 2013 (2013)
18. Wang, S., Xiao, X., Lee, C.: Crowd-based deduplication: an adaptive approach. In: SIGMOD 2015, Melbourne, Victoria, Australia, 31 May–June 4 2015, pp. 1263–1277 (2015)
19. Whang, S.E., Lofgren, P., Garcia-Molina, H.: Question selection for crowd entity resolution. *PVLDB* **6**(6), 349–360 (2013)
20. Zheng, Y., Cheng, R., Maniu, S., Mo, L.: On optimality of jury selection in crowdsourcing. In: EDBT 2015, Brussels, Belgium, 23 March–27 March 2015, pp. 193–204 (2015)
21. Zheng, Y., Li, G., Li, Y., Shan, C., Cheng, R.: Truth inference in crowdsourcing: is the problem solved? *PVLDB* **10**(5), 541–552 (2017)
22. Zheng, Y., Wang, J., Li, G., Cheng, R., Feng, J.: QASCA: a quality-aware task assignment system for crowdsourcing applications. In: SIGMOD, pp. 1031–1046 (2015)
23. Zhuang, Y., Li, G., Zhong, Z., Feng, J.: Crowd-based large knowledge bases alignment. Technical report (2016). <http://dbgroup.cs.tsinghua.edu.cn/ligl/crowdalign.pdf>

Web Information Systems Engineering – WISE 2017

18th International Conference, Puschino, Russia,

October 7-11, 2017, Proceedings, Part II

Bouguettaya, A.; Gao, Y.; Klimenko, A.; Chen, L.; Zhang,

X.; Dzerzhinskiy, F.; Jia, W.; Klimenko, S.V.; Li, Q. (Eds.)

2017, XXII, 576 p. 167 illus., Softcover

ISBN: 978-3-319-68785-8