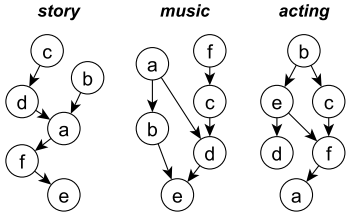
Crowdsourcing Pareto-Optimal Object Finding by Pairwise Comparisons

ABSTRACT

This is the first study on crowdsourcing Pareto-optimal object finding, which has applications in public opinion collection, group decision making, and information exploration. Departing from prior studies on crowdsourcing skyline and ranking queries, it considers the case where objects do not have explicit attributes and preference relations on objects are strict partial orders. The partial orders are derived by aggregating crowdsourcers' responses to pairwise comparison questions. The goal is to find all Pareto-optimal objects by the fewest possible questions. It employs an iterative question-selection framework. Guided by the principle of eagerly identifying non-Pareto optimal objects, the framework only chooses candidate questions which must satisfy three conditions. This design is both sufficient and efficient, as it is proven to find a short terminal question sequence. The framework is further steered by two ideas—macro-ordering and micro-ordering. By different micro-ordering heuristics, the framework is instantiated into several algorithms with varying power in pruning questions. Experiment results using both real crowdsourcing marketplace and simulations exhibited not only orders of magnitude reductions in questions when compared with a brute-force approach, but also close-to-optimal performance from the most efficient instantiation.

1. INTRODUCTION

The growth of user engagement and functionality in crowdsourcing platforms has made computationally challenging tasks unprecedentedly convenient. The subject of our study is one such task—crowdsourcing *Pareto-optimal object finding*. The concept of Pareto-optimal objects resembles that of *skyline objects* [5], but there are several critical differences which shall be discussed later in this section, Section 2, and Table 1. Consider a set of objects O and a set of criteria C for comparing the objects. An object $\mathsf{x}\in O$ is *Pareto-optimal* if and only if x is not dominated by any other object, i.e., $\nexists \mathsf{y}\in O$ such that $\mathsf{y}\succ\mathsf{x}$. An object y dominates x (denoted $\mathsf{y}\succ\mathsf{x}$) if and only if x is not better than y by any criterion and y is better than x by at least one criterion, i.e., $\forall c\in C : \mathsf{x}\not\succ_c\mathsf{y}$ and $\exists c\in C : \mathsf{y}\succ_c\mathsf{x}$. If x and y do not dominate each other (i.e., $\mathsf{x}\not\succ\mathsf{y}$ and $\mathsf{y}\not\succ\mathsf{x}$), we denote it by $\mathsf{x}\sim\mathsf{y}$. The *preference (better-than) relation* P_c (also denoted $\succ_c$) for each $c\in C$ is a binary relation subsumed by $O\times O$, in which a tuple $(\mathsf{x},\mathsf{y})\in P_c$ (also denoted $\mathsf{x}\succ_c\mathsf{y}$) is interpreted as "$\mathsf{x}$ is better than (preferred over) y with regard to criterion c". Hence, if $(\mathsf{x},\mathsf{y})\notin P_c$ (also denoted $\mathsf{x}\not\succ_c\mathsf{y}$), x is not better than y by criterion c. We say x and y are *indifferent* regarding c (denoted $\mathsf{x}\sim_c\mathsf{y}$), if $(\mathsf{x},\mathsf{y})\notin P_c \wedge (\mathsf{y},\mathsf{x})\notin P_c$. We consider the setting where each P_c is a *strict partial order* as opposed to a bucket order [15] or a total order, i.e., P_c is irreflexive ($\forall \mathsf{x} : (\mathsf{x},\mathsf{x}) \notin P_c$) and transitive ($\forall \mathsf{x},\mathsf{y} : (\mathsf{x},\mathsf{y})\in P_c\wedge(\mathsf{y},\mathsf{z})\in P_c\Rightarrow(\mathsf{x},\mathsf{z})\in P_c$), which together imply asymmetry ($\forall \mathsf{x},\mathsf{y} : (\mathsf{x},\mathsf{y})\in P_c \Rightarrow (\mathsf{y},\mathsf{x})\notin P_c$). We note that such definition of better-than relation has been widely used in modeling preferences (e.g., [19, 13, 27]).



(a) Preference relations (i.e., strict partial orders) on three criteria.

QUESTION	ANSWER $\succ$	ANSWER $\sim$	ANSWER $\prec$	OUTCOME
$\mathsf{a}?_s\mathsf{b}$	1	0	4	$\mathsf{b}\succ_s\mathsf{a}$
$\mathsf{a}?_s\mathsf{c}$	0	0	5	$\mathsf{c}\succ_s\mathsf{a}$
$\mathsf{a}?_s\mathsf{d}$	0	2	3	$\mathsf{d}\succ_s\mathsf{a}$
$\mathsf{a}?_s\mathsf{e}$	4	0	1	$\mathsf{a}\succ_s\mathsf{e}$
$\mathsf{a}?_s\mathsf{f}$	3	1	1	$\mathsf{a}\succ_s\mathsf{f}$
$\mathsf{b}?_s\mathsf{c}$	1	2	2	$\mathsf{b}\sim_s\mathsf{c}$
$\mathsf{b}?_s\mathsf{d}$	1	3	1	$\mathsf{b}\sim_s\mathsf{d}$
$\mathsf{b}?_s\mathsf{e}$	5	0	0	$\mathsf{b}\succ_s\mathsf{e}$
$\mathsf{b}?_s\mathsf{f}$	4	1	0	$\mathsf{b}\succ_s\mathsf{f}$
$\mathsf{c}?_s\mathsf{d}$	3	2	0	$\mathsf{c}\succ_s\mathsf{d}$
$\mathsf{c}?_s\mathsf{e}$	4	0	1	$\mathsf{c}\succ_s\mathsf{e}$
$\mathsf{c}?_s\mathsf{f}$	3	1	1	$\mathsf{c}\succ_s\mathsf{f}$
$\mathsf{d}?_s\mathsf{e}$	3	0	2	$\mathsf{d}\succ_s\mathsf{e}$
$\mathsf{d}?_s\mathsf{f}$	3	2	0	$\mathsf{d}\succ_s\mathsf{f}$
$\mathsf{e}?_s\mathsf{f}$	1	1	3	$\mathsf{f}\succ_s\mathsf{e}$

(b) Deriving the preference relation for criterion *story* by pairwise comparisons. Each comparison is performed by 5 workers. $\theta = 60\%$.

Figure 1: Finding Pareto-optimal movies by *story*, *music*, *acting*.

Pareto-optimal object finding lends itself to applications in several areas, including public opinion collection, group decision making, and information exploration, exemplified by the following motivating examples.

Example 1 (Collecting Public Opinion and Group Decision Making)**.** Consider a set of movies O={a,b,c,d,e,f} and a set of criteria C={*story, music, acting*} (denoted by s, m, a in the ensuing discussion). Fig.1a shows the individual preference relations (i.e., strict partial orders), one per criterion. Each strict partial order is graphically represented as a directed acyclic graph (DAG), more

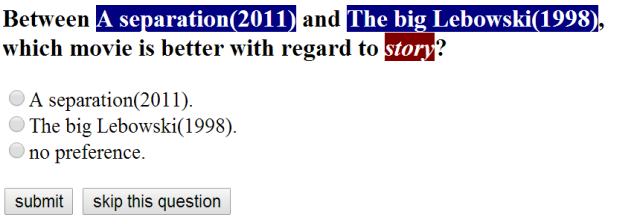


Figure 2: A question that asks to compare two movies by *story*.

specifically a Hasse diagram. The existence of a simple path from x to y in the DAG means x is better than (preferred to) y by the corresponding criterion. For example, $(\mathsf{a},\mathsf{e})\in P_m$ ($\mathsf{a}\succ_m\mathsf{e}$), i.e., a is better than e by *music*. $(\mathsf{b},\mathsf{d})\notin P_s$ and $(\mathsf{d},\mathsf{b})\notin P_s$; hence $\mathsf{b}\sim_s\mathsf{d}$. The partial orders define the dominance relation between objects. For instance, movie c dominates d ($\mathsf{c}\succ\mathsf{d}$), because c is preferred than d on *story* and *music* and they are indifferent on *acting*, i.e., $\mathsf{c}\succ_s\mathsf{d}$, $\mathsf{c}\succ_m\mathsf{d}$, and $\mathsf{c}\sim_a\mathsf{d}$; a and b do not dominate each other ($\mathsf{a}\sim\mathsf{b}$), since $\mathsf{b}\succ_s\mathsf{a}$, $\mathsf{a}\succ_m\mathsf{b}$ and $\mathsf{b}\succ_a\mathsf{a}$. Based on the three partial orders, b is the only Pareto-optimal object, since no other objects dominate it and every other object is dominated by some object.

Note that tasks such as the above one may be used in both understanding the public's preference (i.e., the preference relations are collected from a large, anonymous crowd) and making decisions for a target group (i.e., the preference relations are from a small group of people). □

Example 2 (Information Exploration)**.** Consider a photography enthusiast, *Amy*, who is drown in a large number of photos she has taken and wants to select a subset of the better ones. She resorts to crowdsourcing for the task, as it has been exploited by many for similar tasks such as photo tagging, location/face identification, sorting photos by (guessed) date, and so on. Particularly, she would like to choose Pareto-optimal photos with regard to *color*, *sharpness* and *landscape*. □

By definition, the crux of finding Pareto-optimal objects lies in obtaining the preference relations, i.e., the strict partial orders on individual criteria. Through crowdsourcing, the preference relations are derived by aggregating the crowd's responses to *pairwise comparison* tasks. Each such comparison between objects x and y by criterion c is a question, denoted $\mathsf{x}?_c\mathsf{y}$, which has three possible outcomes—$\mathsf{x}\succ_c\mathsf{y}$, $\mathsf{y}\succ_c\mathsf{x}$, and $\mathsf{x}\sim_c\mathsf{y}$, based on the crowd's answers. An example is as follows.

Example 3 (Deriving Preference Relations from Pairwise Comparisons by the Crowd)**.** Fig.1b shows the hypothetical results of all 15 pairwise comparisons between the 6 movies in Example 1, by criterion s=*story*. The outcomes of all comparisons form the crowd's preference relation on *story* (the leftmost DAG in Fig.1a). Fig.2 is the screenshot of a question form designed for one such comparison. A crowdsourcer, when facing this question, would make a choice among the three possible answers or skip a question if they do not have enough confidence or knowledge to answer it. Fig.1b shows how many crowdsourcers have selected each answer. For instance, for question $\mathsf{a}?_s\mathsf{f}$, three people preferred movie a, one person preferred f, and one person is indifferent. By aggregating these answers, it is derived that a is better than f with regard to *story*, since 60% of the crowdsourcers who responded to the question chose this answer. For question $\mathsf{b}?_s\mathsf{c}$, the result is $\mathsf{b}\sim_s\mathsf{c}$, since neither $\mathsf{b}\succ_s\mathsf{c}$ nor $\mathsf{b}\prec_s\mathsf{c}$ received enough votes. (Assuming a threshold θ=60%, i.e., either $\mathsf{b}\succ_s\mathsf{c}$ or $\mathsf{b}\prec_s\mathsf{c}$ should have at least 60% of votes, in order to not declare $\mathsf{b}\sim_s\mathsf{c}$.) □

To the best of our knowledge, this paper is the first work on crowdsourcing Pareto-optimal object finding. The definition of Pareto-optimal objects follows the concept of *Pareto composition* of preference relations in [13]. It also resembles the definition of *skyline* objects on totally-ordered attribute domains (pioneered by [5]) and partially-ordered domains [11, 27, 28, 31]. However, except for [22], previous studies on preference and skyline queries do not use the crowd; they focus on query processing on *existing* data. On the contrary, we examine how to ask the crowd as few questions as possible in obtaining sufficient data for determining Pareto-optimal objects. Furthermore, our work differs from preference and skyline queries (including [22]) in several radical ways:

- The preference relation for a criterion is *not* governed by explicit scores or values on object attributes (e.g., sizes of houses, prices of hotels), while preference and skyline queries on both totally- and partially-ordered domains assumed explicit attribute representation. For many comparison criteria, it is difficult to model objects by explicit attributes, not to mention asking people to provide such values or scores; people's preferences are rather based on complex, subtle perceptions, as demonstrated in Examples 1 and 2.
- Due to the above reason, we request crowdsourcers to perform pairwise comparisons instead of directly providing attribute values or scores. On the contrary, [22] uses the crowd to obtain missing attribute values. Pairwise comparison is extensively studied in social choice and welfare, preferences, and voting. It is known that people are more comfortable and confident with comparing objects than directly scoring them, since it is easier, faster, and less error-prone [29].
- The crowd's preference relations are modeled as strict partial orders, as opposed to bucket orders or full orders. This is not only a direct effect of using pairwise comparisons instead of numeric scores or explicit attribute values, but also a reflection of the psychological nature of human's preferences [19, 13], since it is not always natural to enforce a total or bucket order. Most studies on skyline queries assume total/bucket orders, except for [11, 27, 28, 31] which consider partial orders.

Our objective is to find all Pareto-optimal objects with as few questions as possible. A brute-force approach will obtain complete preference relations via pairwise comparisons of all object pairs by every criterion. However, without such exhaustive comparisons, incomplete knowledge collected from a small set of questions may suffice in discerning all Pareto-optimal objects. Toward this end, it may appear that we can take advantage of the transitivity of object dominance—a cost-saving property often exploited in skyline query algorithms (e.g., [5]) to exclude dominated objects from participating in any future comparison once they are detected. But, we shall prove that object dominance in our case is *not* transitive (Property 1), due to the lack of explicit attribute representation. Hence, the aforementioned cost-saving technique is inapplicable.

Aiming at Pareto-optimal object finding by a short sequence of questions, we introduce a general, iterative algorithm framework (Sec.3). Each iteration goes through four steps—*question selection*, *outcome derivation*, *contradiction resolution*, and *termination test*. In the i-th iteration, a question $q_i=\mathsf{x}?_c\mathsf{y}$ is selected and its outcome is determined based on crowdsourcers' answers. On unusual occasions, if the outcome presents a contradiction to the obtained outcomes of other questions, it is changed to the closest outcome such that the contradiction is resolved. Based on the transitive closure of the outcomes to the questions so far, the objects O are partitioned into three sets—$O_\surd$ (objects that must be Pareto-optimal), $O_\times$ (objects that must be non-Pareto optimal), and $O_?$ (objects whose Pareto-optimality cannot be fully discerned by the incomplete knowledge so far). When $O_?$ becomes empty, $O_\surd$ contains all Pareto-optimal objects and the algorithm terminates. The question sequence so far is thus a *terminal sequence*.

	Task	Question type	Multiple attributes	Order among objects (on each attribute)	Explicit attribute representation
[12]	full ranking	pairwise comparison	no	bucket/total order	no
[24]	top-*k* ranking	rank subsets of objects	no	bucket/total order	no
[14]	top-*k* ranking and grouping	pairwise comparison	no	bucket/total order	no
[22]	skyline queries	missing value inquiry	yes	bucket/total order	yes
This work	Pareto-optimal object finding	pairwise comparison	yes	strict partial order	no

Table 1: Related work comparison.

There are a vast number of terminal sequences. Our goal is to find one that is as short as possible. We observe that, for a non-Pareto optimal object, knowing that it is dominated by at least one object is sufficient, and we do not need to find all its dominating objects. It follows that we do not really care about the dominance relation between non-Pareto optimal objects and we can skip their comparisons. Hence, the overriding principle of our question selection strategy is to identify non-Pareto optimal objects as early as possible. Guided by this principle, the framework only chooses from *candidate questions* which must satisfy three conditions (Sec.3.1). This design is sufficient, as we prove that an empty candidate question set implies a terminal sequence, and vice versa (Proporty 2). The design is also efficient, as we further prove that, if a question sequence contains non-candidate questions, there exists a shorter or equally long sequence with only candidate questions that produces the same $O_\times$, matching the principle of eagerly finding non-Pareto optimal objects (Theorem 1). Moreover, by the aforementioned principle, the framework selects in every iteration such a candidate question $x?_c y$ that x is more likely to be dominated by y. The selection is steered by two ideas—*macro-ordering* and *micro-ordering*. By using different micro-ordering heuristics, the framework is instantiated into several algorithms with varying power in pruning questions (Sec.4). We also derive a lower bound on the number of questions required for finding all Pareto-optimal objects (Theorem 2).

In summary, this paper makes the following contributions:

- This is the first work on crowdsourcing Pareto-optimal object finding. Prior studies on crowdsourcing skyline queries [22] assumes explicit attribute representation and uses crowd to obtain missing attribute values. We define preference relations purely based on pairwise comparisons and we aim to find all Pareto-optimal objects by as few comparisons as possible.
- We propose a general, iterative algorithm framework (Sec.3) which follows the strategy of choosing only candidate questions that must satisfy three conditions. We prove important properties that establish the advantage of the strategy (Sec.3.1).
- We design macro-ordering and micro-ordering heuristics for finding a short terminal question sequence. Based on the heuristics, the generic framework is instantiated into several algorithms (RandomQ, RandomP, FRQ) with varying efficiency. We also derive a non-trivial lower bound on the number of required pairwise comparison questions. (Sec.4)
- We carried out experiments using both simulations and real crowdsourcing marketplace to compare the amount of comparisons required by different instantiations of the framework under varying problem sizes. The results demonstrate the effectiveness of selecting only candidate questions, macro-ordering, and micro-ordering. When these ideas are stacked together, they use orders of magnitude less comparisons than a brute-force approach. The results also reveal that FRQ is nearly optimal and the lower bound is practically tight, since FRQ gets very close to the lower bound. (Sec.5)

2. RELATED WORK

This is the first work on crowdsourcing Pareto-optimal object finding. There are several recent studies on using crowdsourcing to rank objects and answer group-by, top-*k* and skyline queries. Crowd-BT [12] ranks objects by crowdsourcing pairwise object comparisons. Polychronopoulos et al. [24] find top-*k* items in an itemset by asking human workers to rank small subsets of items. Davidson et al. [14] evaluate top-*k* and group-by queries by asking the crowd to answer *type* questions (whether two objects belong to the same group) and *value* questions (ordering two objects). Lofi et al. [22] answer skyline queries over incomplete data by asking the crowd to provide missing attribute values. Table 1 summarizes the similarities and differences between these studies and our work. The studies on full and top-*k* ranking [12, 24, 14] do not consider multiple attributes in modeling objects. On the contrary, the concepts of skyline [22] and Pareto-optimal objects (this paper) are defined in a space of multiple attributes. [22] assumes explicit attribute representation. Therefore, they resort to the crowd for completing missing values, while other studies including our work request the crowd to compare objects. Our work considers strict partial orders among objects on individual attributes. Differently, other studies assume a bucket/total order [12, 24, 14] or multiple bucket/total orders on individual attributes [22].

Besides [12], there were multiple studies on ranking objects by pairwise comparisons, which date back to decades ago as aggregating the preferences of multiple agents has always been a fundamental problem in social choice and welfare [4]. The more recent studies can be categorized into three types: **1)** Approaches such as [20, 25, 30] predict users' object ranking by completing a user-object scoring matrix. Their predications take into account users' similarities in pairwise comparisons, resembling *collaborative filtering* [16]. They thus do not consider explicit attribute representation for objects. **2)** Approaches such as [7, 9, 8] infer query-specific (instead of user-specific) ranked results to web search queries. Following the paradigm of *learning-to-rank* [21], they rank a query's result documents according to pairwise result comparisons of other queries. The documents are modeled by explicit ranking features. **3)** Approaches such as [6, 18, 2, 3, 23] are similar to [12] as they use pairwise comparisons to infer a single ranked list that is neither user-specific nor query-specific. Among them, [18] is special in that it also applies learning-to-rank and requires explicit feature representation. Different from our work, none of these studies is about Pareto-optimal objects, since they all assume a bucket/total order among objects; those using learning-to-rank require explicit feature representation, while the rest do not consider multiple attributes. Moreover, except [18, 2, 3], they all assume comparison results are already obtained before their algorithms kick in. In contrast, we aim at minimizing the pairwise comparison questions to ask in finding Pareto-optimal objects.

3. GENERAL FRAMEWORK

By the definition of Pareto-optimal objects, the key to finding such objects is to obtain the preference relations, i.e., the strict

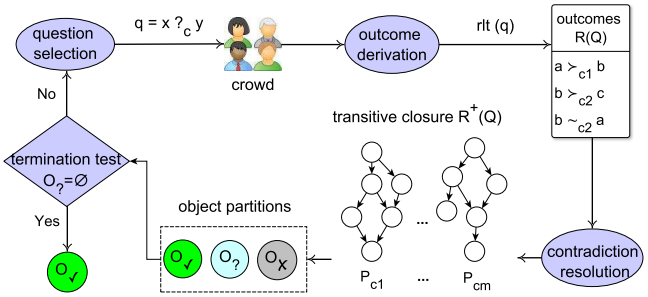


Figure 3: The general framework.

partial orders on individual criteria. Toward this end, the most basic operation is to perform *pairwise comparison*—given a pair of objects x and y and a criterion c, determine whether one is better than the other (i.e., $(\mathsf{x},\mathsf{y}) \in P_c$ or $(\mathsf{y},\mathsf{x}) \in P_c$) or they are indifferent (i.e., $(\mathsf{x},\mathsf{y}) \notin P_c \wedge (\mathsf{y},\mathsf{x}) \notin P_c$).

The problem of crowdsourcing Pareto-optimal object finding is thus essentially crowdsourcing pairwise comparisons. Each comparison task between x and y by criterion c is presented to the crowd as a question q (denoted $\mathsf{x}?_c\mathsf{y}$). The outcome to the question (denoted $rlt(q)$) is aggregated from the crowd's answers. Given a set of questions, the outcomes thus contain an (incomplete) knowledge of the crowd's preference relations for various criteria. Fig.2 illustrates the screenshot of one such question (comparing two movies by *story*) used in our empirical evaluation. We note that there are other viable designs of question, e.g., only allowing the first two choices ($\mathsf{x} \succ_c \mathsf{y}$ and $\mathsf{y} \succ_c \mathsf{x}$). Our work is agnostic to the specific question design.

Given n objects and r criteria, a brute-force approach will perform pairwise comparisons on all object pairs by every criterion, which leads to $r{\cdot}n{\cdot}(n{-}1)/2$ comparisons. The corresponding question outcomes amount to the complete underlying preference relations. The quadratic nature of the brute-force approach renders it wasteful. The bad news is that, in the worst case, we cannot do better than it. To understand this, consider the scenario where all objects are indifferent by every criterion. If any comparison $\mathsf{x}?_c\mathsf{y}$ is skipped, we cannot determine if x and y are indifferent or if one dominates another.

In practice, though, the outlook is much brighter. Since we look for only Pareto-optimal objects, it is an overkill to obtain complete preference relations. Specifically, for a Pareto-optimal object, knowing it is not dominated by any object is sufficient, and we do not need to find all the objects dominated by it; for a non-Pareto optimal object, knowing it is dominated by at least one object is sufficient, and we do not need to find all its dominating objects. Hence, without exhausting all possible comparisons, incomplete knowledge on preference relations collected from a set of questions may suffice in fully discerning all Pareto-optimal objects.

Our objective is to find all Pareto-optimal objects with as few questions as possible. By pursuing this goal, we are applying a very simple cost model—the cost of a solution only depends on its number of questions. Although the cost of a task in a crowdsourcing environment may depend on monetary cost, latency and other factors, the number of questions is a generic, platform-independent cost measure and arguably proportionally correlates with the real cost. Therefore, we assume a sequential execution model which asks the crowd an ordered sequence of questions $Q = \langle q_1, ..., q_n \rangle$—it only asks q_{i+1} after $rlt(q_i)$ is obtained. Thereby, we do not consider asking multiple questions concurrently. Furthermore, in discussion of our approach, the focus shall be on how to find a short question sequence instead of the algorithms' complexity.

Algorithm 1: The general framework

Input: O : the set of objects
Output: $O_{\surd}$: Pareto-optimal objects of O

```
1 R(Q) ← ∅;                                          /* question outcomes */
2 repeat
3     x?cy ← question selection;
4     rlt(x?cy) ← outcome derivation;                /* resolve conflict, if any */
5     R(Q) ← R(Q) ∪ {rlt(x?cy)};
6     (O√, O×, O?) ← partitioning objects based on R+(Q);
      /* R+(Q) is the transitive closure of R(Q) */
7 until O? = {};
8 return O√;
```

To find a short sequence, we design a general algorithm framework, as displayed in Fig.3. Alg.1 shows the framework's pseudocode. Its execution is iterative. Each iteration goes through four steps—*question selection*, *outcome derivation*, *contradiction resolution*, and *termination test*. In the i-th iteration, a question $q_i{=}\mathsf{x}?_c\mathsf{y}$ is selected and presented to the crowd. The question outcome $rlt(q_i)$ is derived from the crowd's aggregated answers. On unusual occasions, if the outcome presents a contradiction to the obtained outcomes of other questions so far, it is changed to the closest outcome to resolve contradiction. By computing $R^+(Q_i)$, the *transitive closure* of $R(Q_i)$—the obtained outcomes to questions so far $\langle q_1, \ldots, q_i \rangle$, the outcomes to certain questions are derived and such questions will never be asked. Based on $R^+(Q_i)$, if every object is determined to be either Pareto-optimal or non-Pareto optimal without uncertainty, the algorithm terminates.

Below, we discuss outcome derivation and termination test. Sec.3.1 examines the framework's key step—question selection, and Sec.3.2 discusses contradiction resolution.

Outcome derivation Given a question $\mathsf{x}?_c\mathsf{y}$, its outcome $rlt(\mathsf{x}?_c\mathsf{y})$ must be aggregated from multiple crowdsourcers, in order to reach a reliable result with confidence. Particularly, one of three mutually-exclusive outcomes is determined based on k crowdsourcers' answers to the question:

$$rlt(\mathsf{x}?_c\mathsf{y}) = \begin{cases} \mathsf{x} \succ_c \mathsf{y} & \text{if } \frac{\#\mathsf{x}}{k} \geq \theta \\ \mathsf{y} \succ_c \mathsf{x} & \text{if } \frac{\#\mathsf{y}}{k} \geq \theta \\ \mathsf{x} \sim_c \mathsf{y}\ (\mathsf{x} \not\succ_c \mathsf{y} \wedge \mathsf{y} \not\succ_c \mathsf{x}) & \text{otherwise} \end{cases} \quad (1)$$

where θ is such a predefined threshold that $\theta{>}50\%$, $\#\mathsf{x}$ is the number of crowdsourcers (out of k) preferring x over y on criterion c, and $\#\mathsf{y}$ is the number of crowdsourcers preferring y over x on c. Fig.1b shows the outcomes of all 15 questions according to Equation (1) for comparing movies by *story* using $k{=}5$ and $\theta{=}60\%$. Other conceivable definitions may be used in determining the outcome of $\mathsf{x}?_c\mathsf{y}$. For example, the outcome may be defined as the choice (out of the three possible choices) that receives the most votes from the crowd. The ensuing discussion is agnostic to the specific definition.

The current framework does not consider different levels of confidence on question outcomes. The confidence on the outcome of a question may be represented as a probability value based on the distribution of crowdsourcers' responses. An interesting direction for future work is to find Pareto-optimal objects in probabilistic sense. The confidence may also reflect the crowdsourcers' quality and credibility [17].

Termination test In each iteration, Alg.1 partitions the objects into three sets by their Pareto-optimality based on the transitive closure of question outcomes so far. If every object's Pareto-optimality has been determined without uncertainty, the algorithm terminates. Details are as follows.

Definition 1 (Transitive Closure of Outcomes)**.** Given a set of questions $Q=\langle q_1, ..., q_n\rangle$, the transitive closure of their outcomes $R(Q)=\{rlt(q_1), ..., rlt(q_n)\}$ is $R^+(Q)=\{\mathsf{x}\succsim_c\mathsf{y} \mid \mathsf{x}\sim_c\mathsf{y}\in R(Q)\}\bigcup\{\mathsf{x}\succ_c\mathsf{y} \mid (\mathsf{x}\succ_c\mathsf{y}\in R(Q))\vee(\exists \mathsf{w}_1,\mathsf{w}_2,...,\mathsf{w}_m : \mathsf{w}_1=\mathsf{x}, \mathsf{w}_m=\mathsf{y}\wedge(\forall 0<i<m : \mathsf{w}_i\succ_c\mathsf{w}_{i+1}\in R(Q)))\}$. □

In essence, the transitive closure dictates $\mathsf{x}\succ_c\mathsf{z}$ without asking the question $\mathsf{x}?_c\mathsf{z}$, if the existing outcomes $R(Q)$ (and recursively the transitive closure $R^+(Q)$) contains both $\mathsf{x}\succ_c\mathsf{y}$ and $\mathsf{y}\succ_c\mathsf{z}$. Based on $R^+(Q)$, the objects O can be partitioned into three sets:

$O_{\surd}=\{\mathsf{x}\in O \mid \forall \mathsf{y}\in O : (\exists c\in C : \mathsf{x}\succ_c\mathsf{y}\in R^+(Q))\vee(\forall c\in C : \mathsf{x}\sim_c\mathsf{y}\in R^+(Q))\}$;

$O_{\times}=\{\mathsf{x}\in O \mid \exists \mathsf{y}\in O : (\forall c\in C : \mathsf{y}\succ_c\mathsf{x}\in R^+(Q)\vee \mathsf{x}\sim_c\mathsf{y}\in R^+(Q))\wedge(\exists c\in C : \mathsf{y}\succ_c\mathsf{x}\in R^+(Q))\}$;

$O_?=O\backslash(O_{\surd}\cup O_{\times})$.

$O_{\surd}$ contains objects that must be Pareto-optimal, $O_{\times}$ contains objects that cannot possibly be Pareto-optimal, and $O_?$ contains objects for which the incomplete knowledge $R^+(Q)$ is insufficient for discerning their Pareto-optimality. The objects in $O_?$ may turn out to be Pareto-optimal after more comparison questions. If the set $O_?$ for a question sequence Q is empty, $O_{\surd}$ contains all Pareto-optimal objects and the algorithm terminates. We call such a Q a *terminal sequence*, defined below.

Definition 2 (Terminal Sequence)**.** A question sequence Q is a terminal sequence if and only if, based on $R^+(Q)$, $O_?=\varnothing$.

3.1 Question Selection

Given objects O and criteria C, there can be a huge number of terminal sequences. Our goal is to find a sequence as short as possible. As Fig.3 and Alg.1 show, the framework is an iterative procedure of object partitioning based on question outcomes. It can also be viewed as the process of moving objects from $O_?$ to $O_{\surd}$ and $O_{\times}$. Once an object is moved to $O_{\surd}$ or $O_{\times}$, it cannot be moved again. With regard to this process, we make two important observations, as follows.

- *In order to declare an object* x *not Pareto-optimal, it is sufficient to just know* x *is dominated by another object*. It immediately follows that we do not really care about the dominance relationship between objects in $O_{\times}$ and thus can skip the comparisons between such objects. Once we know $\mathsf{x}\in O_?$ is dominated by another object, it cannot be Pareto-optimal and is immediately moved to $O_{\times}$. Quickly moving objects into $O_{\times}$ can allow us skipping many comparisons between objects in $O_{\times}$.
- *In order to declare an object* x *Pareto-optimal, it is necessary to know that no object can dominate* x. This means we may need to compare x with all other objects including non Pareto-optimal objects. As an extreme example, it is possible for x to be dominated by only a non-Pareto optimal object y but not by any other object (not even the objects dominating y). This is because object dominance based on preference relations is intransitive, which is formally stated in Property 1.

  Property 1 (Intransitivity of Object Dominance)**.** Object dominance based on the preference relations over a set of criteria is not transitive. Specifically, if $\mathsf{x}\succ\mathsf{y}$ and $\mathsf{y}\succ\mathsf{z}$, it is not necessarily true that $\mathsf{x}\succ\mathsf{z}$. In other words, it is possible that $\mathsf{x}\sim\mathsf{z}$ or even $\mathsf{z}\succ\mathsf{x}$. □

  We show the intransitivity of object dominance by an example. Consider objects $O=\{\mathsf{x},\mathsf{y},\mathsf{z}\}$, criteria $C=\{c_1, c_2, c_3\}$, and the preference relations in Fig.4. Three dominance relationships violate transitivity: (i) $\mathsf{x}\succ\mathsf{y}$ (based on $\mathsf{x}\succ_{c_1}\mathsf{y}$, $\mathsf{x}\sim_{c_2}\mathsf{y}$, $\mathsf{x}\sim_{c_3}\mathsf{y}$), (ii) $\mathsf{y}\succ\mathsf{z}$ (based on $\mathsf{y}\sim_{c_1}\mathsf{z}$, $\mathsf{y}\succ_{c_2}\mathsf{z}$, $\mathsf{y}\sim_{c_3}\mathsf{z}$), and (iii) $\mathsf{z}\succ\mathsf{x}$ (based on $\mathsf{z}\sim_{c_1}\mathsf{x}$, $\mathsf{z}\sim_{c_2}\mathsf{x}$, $\mathsf{z}\succ_{c_3}\mathsf{x}$). As another example, in Fig.1a, $\mathsf{b}\succ\mathsf{c}$ (since $\mathsf{b}\sim_s\mathsf{c}$, $\mathsf{b}\sim_m\mathsf{c}$, $\mathsf{b}\succ_a\mathsf{c}$, where s=*story*, m=*music*, a=*acting*) and $\mathsf{c}\succ\mathsf{a}$ (since $\mathsf{c}\succ_s\mathsf{a}$, $\mathsf{c}\sim_m\mathsf{a}$, $\mathsf{c}\succ_a\mathsf{a}$), but $\mathsf{a}\sim\mathsf{b}$ (since $\mathsf{b}\succ_s\mathsf{a}$, $\mathsf{a}\succ_m\mathsf{b}$, $\mathsf{b}\succ_a\mathsf{a}$) where transitivity does not hold.

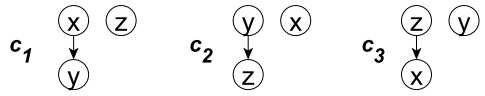


Figure 4: Intransitivity of object dominance: $\mathsf{x}\succ\mathsf{y}$, $\mathsf{y}\succ\mathsf{z}$, $\mathsf{z}\succ\mathsf{x}$.

Algorithm 2: Question selection

Input: $R^+(Q_i), O_{\surd}(Q_i), O_?(Q_i), O_{\times}(Q_i)$
Output: Q^1_{can} or Q^2_{can}

$Q_{can}\leftarrow\{\mathsf{x}?_c\mathsf{y} \mid rlt(\mathsf{x}?_c\mathsf{y})\notin R^+(Q_i)\wedge\mathsf{x}\in O_?(Q_i)\wedge(\nexists c'\in C : \mathsf{x}\succ_{c'}\mathsf{y}\in R^+(Q_i))\}$;
$Q^1_{can}\leftarrow\{\mathsf{x}?_c\mathsf{y} \mid \mathsf{x}?_c\mathsf{y}\in Q_{can}, \mathsf{y}\notin O_{\times}(Q_i)\}$;
$Q^2_{can}\leftarrow\{\mathsf{x}?_c\mathsf{y} \mid \mathsf{x}?_c\mathsf{y}\in Q_{can}, \mathsf{y}\in O_{\times}(Q_i)\}$;
/* *Macro-ordering: consider* Q^1_{can} *before* Q^2_{can}. */
if $Q^1_{can}\neq\varnothing$ **then**
  return *micro-ordering*(Q^1_{can});
else if $Q^2_{can}\neq\varnothing$ **then**
  return *micro-ordering*(Q^2_{can});

Differently, transitivity of object dominance holds in skyline analysis [5]. The contradiction is due to the lack of explicit attribute representation—in our case two objects may be considered equally good on a criterion if they are indifferent, while in skyline analysis they are equally good regarding an attribute if they bear identical values. Skyline query algorithms exploit the transitivity of object dominance to reduce execution cost, because an object can be immediately excluded from further comparison once it is found dominated by any other object. However, due to Property 1, we cannot leverage such pruning anymore.

Based on these observations, the overriding principle of our question selection strategy (Alg.2) is to identify non-Pareto optimal objects as early as possible. At every iteration of the framework (Alg.1), we choose to compare x and y by criterion c (i.e., ask question $\mathsf{x}?_c\mathsf{y}$) where $\mathsf{x}?_c\mathsf{y}$ belongs to *candidate questions*. Such candidate questions must satisfy three conditions (Definition 3). There can be many candidate questions. In choosing the next question, by the aforementioned principle, we select such $\mathsf{x}?_c\mathsf{y}$ that x is more likely to be dominated by y. More specifically, we design two ordering heuristics—*macro-ordering* and *micro-ordering*. Given the three object partitions $O_{\surd}$, $O_{\times}$ and $O_?$, the macro-ordering idea is simply that we choose x from $O_?$ (required by one of the conditions on candidate questions) and y from $O_{\surd}\cup O_?$ (if possible) or $O_{\times}$ (otherwise). The reason is that it is less likely for an object in $O_{\times}$ to dominate x. Micro-ordering further orders all candidate questions satisfying the macro-ordering heuristic. In Sec.4, we instantiate the framework into a variety of solutions with varying power in pruning questions, by using different micro-ordering heuristics.

Definition 3 (Candidate Question)**.** Given Q, the set of asked questions so far, $\mathsf{x}?_c\mathsf{y}$ is a *candidate question* if and only if it satisfies the following conditions:

(i) The outcome of $\mathsf{x}?_c\mathsf{y}$ is unknown yet, i.e., $rlt(\mathsf{x}?_c\mathsf{y})\notin R^+(Q)$;
(ii) x must belong to $O_?$;
(iii) Based on $R^+(Q)$, the possibility of $\mathsf{y}\succ\mathsf{x}$ must not be ruled out yet, i.e., $\nexists c'\in C : \mathsf{x}\succ_{c'}\mathsf{y}\in R^+(Q)$.

We denote the set of candidate questions by Q_{can}. Thus, $Q_{can}=\{\mathsf{x}?_c\mathsf{y} \mid rlt(\mathsf{x}?_c\mathsf{y})\notin R^+(Q)\wedge\mathsf{x}\in O_?\wedge(\nexists c'\in C : \mathsf{x}\succ_{c'}\mathsf{y}\in R^+(Q))\}$. □

If no candidate question exists, the question sequence Q is a terminal sequence. The reverse statement is also true, i.e., upon a terminal sequence, there is no candidate question left. This is formalized in the following property.

Property 2. $Q_{can} = \varnothing$ if and only if $O_? = \varnothing$.

Proof. The proof is omitted due to space limitations and can be found in the technical report [1]. □

Questions violating the three conditions may also lead to terminal sequences. However, choosing only candidate questions matches our objective of quickly identifying non-Pareto optimal objects. Below we justify the conditions.

Condition (i): This is straightforward. If $R(Q)$ or its transitive closure already contains the outcome of $\mathsf{x}?_c\mathsf{y}$, we do not ask the same question again.

Condition (ii): This condition essentially dictates that at least one of the two objects in comparison is from $O_?$. (If only one of them belongs to $O_?$, we make it x.) Given a pair x and y, if neither is from $O_?$, there are three scenarios—(1) $\mathsf{x}\in O_\surd, \mathsf{y}\in O_\surd$, (2) $\mathsf{x}\in O_\surd, \mathsf{y}\in O_\times$ or $\mathsf{x}\in O_\times, \mathsf{y}\in O_\surd$, (3) $\mathsf{x}\in O_\times, \mathsf{y}\in O_\times$. Once we know an object is in $O_\surd$ or $O_\times$, its membership in such a set will never change. Hence, we are not interested in knowing the dominance relationship between objects from $O_\surd$ and $O_\times$ only. In all these three scenarios, comparing x and y is only useful for indirectly determining (by transitive closure) the outcome of comparing other objects. Intuitively speaking, such indirect pruning is not as efficient as direct pruning.

Condition (iii): This condition requires that, when $\mathsf{x}?_c\mathsf{y}$ is chosen, we cannot rule out the possibility of y dominating x. Otherwise, if y cannot possibly dominate x, the outcome of $\mathsf{x}?_c\mathsf{y}$ cannot help prune x. Note that, in such a case, comparing x and y by c may help prune y, if y still belongs to $O_?$ and x may dominate y. Such possibility is not neglected and is covered by a different representation of the same question—$\mathsf{y}?_c\mathsf{x}$, i.e., swapping the positions of x and y in checking the three conditions. If it is determined x and y cannot dominate each other, then their further comparison is only useful for indirectly determining the outcome of comparing other objects. Due to the same reason explained for condition (ii), such indirect pruning is less efficient.

The following simple Property 3 helps to determine whether $\mathsf{y}\succ\mathsf{x}$ is possible: If x is better than y by any criterion, then we can already rule out the possibility of $\mathsf{y}\succ\mathsf{x}$, without knowing the outcome of their comparison by every criterion. This allows us to skip further comparisons between them. Its correctness is straightforward based on the definition of object dominance.

Property 3 (Non-Dominance Property)**.** At any given moment, suppose the set of asked questions is Q. Consider two objects x and y for which the comparison outcome is not known for every criterion, i.e., $\exists c$ such that $rlt(\mathsf{x}?_c\mathsf{y}) \notin R^+(Q)$. It can be determined that $\mathsf{y}\not\succ\mathsf{x}$ if $\exists c \in C$ such that $\mathsf{x}\succ_c\mathsf{y} \in R^+(Q)$. □

In justifying the three conditions in defining candidate questions, we intuitively explained that indirect pruning is less efficient—if it is known that x does not belong to $O_?$ or y cannot possibly dominate x, we will not ask question $\mathsf{x}?_c\mathsf{y}$. We now justify this strategy theoretically and precisely. Consider a question sequence $Q=\langle q_1, \ldots, q_n\rangle$. We use $O_\surd(Q)$, $O_?(Q)$, $O_\times(Q)$ to denote object partitions according to $R^+(Q)$. For any question q_i, the subsequence comprised of its preceding questions is denoted $Q_{i-1}=\langle q_1, \ldots, q_{i-1}\rangle$. If q_i was not a candidate question when it was chosen (i.e., after $R(Q_{i-1})$ was obtained), we say it is a *non-candidate*. The following Theorem 1 states that, if a question sequence contains non-candidate questions, we can replace it by a shorter or equally long sequence without non-candidate questions that produces the same set of dominated objects $O_\times$. Recall that the key to our framework is to recognize dominated objects and move them into $O_\times$ as early as possible. Hence, the new sequence will likely lead to less cost when the algorithm terminates. Hence, it is a good idea to only select among candidate questions.

Theorem 1. If Q contains non-candidate questions, there exists a question sequence Q' without non-candidate questions such that $|Q'| \le |Q|$ and $O_\times(Q') = O_\times(Q)$.

Proof. We prove by demonstrating how to transform Q into such a Q'. Given any non-candidate question $q_i=\mathsf{x}?_c\mathsf{y}$ in Q, we remove it and, when necessary, replace several questions. The decisions and choices are partitioned into three mutually exclusive scenarios, which correspond to violations of the three conditions in Definition 3. The detailed proof is omitted due to space limitations and can be found in the technical report [1]. □

3.2 Resolving Unusual Contradictions in Question Outcomes

A preference relation can be more accurately derived, if more input is collected from the crowd. However, under practical constraints on budget and time, the limited responses from the crowd (k answers per question) may present two types of contradicting preferences.

(i) Suppose $rlt(\mathsf{x}?_c\mathsf{y})=\mathsf{x}\succ_c\mathsf{y}$ and $rlt(\mathsf{y}?_c\mathsf{z})=\mathsf{y}\succ_c\mathsf{z}$ have been derived, i.e., they belong to $R(Q)$. They together imply $\mathsf{x}\succ_c\mathsf{z}$, since a preference relation must be transitive. Therefore the question $\mathsf{x}?_c\mathsf{z}$ will not be asked. If the crowd is nevertheless asked to further compare x and z, the result $rlt(\mathsf{x}?_c\mathsf{z})$ might be possibly $\mathsf{z}\succ_c\mathsf{x}$, which presents a contradiction.

(ii) Suppose $rlt(\mathsf{x}?_c\mathsf{y})=\mathsf{x}\sim_c\mathsf{y}$ and $rlt(\mathsf{y}?_c\mathsf{z})=\mathsf{y}\succ_c\mathsf{z}$ have been derived from the crowd. If the crowd is asked to further compare x and z, the result $rlt(\mathsf{x}?_c\mathsf{z})$ might be possibly $\mathsf{z}\succ_c\mathsf{x}$. The outcomes $\mathsf{y}\succ_c\mathsf{z}$ and $\mathsf{z}\succ_c\mathsf{x}$ together imply $\mathsf{y}\succ_c\mathsf{x}$, which contradicts with $\mathsf{x}\sim_c\mathsf{y}$. (A symmetric case is $rlt(\mathsf{x}?_c\mathsf{y})=\mathsf{x}\sim_c\mathsf{y}$, $rlt(\mathsf{y}?_c\mathsf{z})=\mathsf{z}\succ_c\mathsf{y}$, and the crowd might respond with $rlt(\mathsf{x}?_c\mathsf{z})=\mathsf{x}\succ_c\mathsf{z}$, which also leads to contradiction with $\mathsf{x}\sim_c\mathsf{y}$. The following discussion applies to this symmetric case, which is thus not mentioned again.)

In practice, such contradictions are uncommon. This is easy to understand intuitively—as long as the underlying preference relation is transitive, collective wisdom of the crowds will reflect it. We can find evidence of it in [26, 10], which confirmed that preference judgments of relevance in document retrieval are transitive.

Nevertheless, contradictions still occur. Type (i) contradictions can be prevented by enforcing the following simple Rule 1 to assume transitivity and thus skip certain questions. They will never get into the derived preference relations. In fact, in calculating transitive closure (Definition 1) and defining candidate questions (Sec.3.1), we already apply this rule.

Rule 1 (Contradiction Prevention by Skipping Questions)**.** Given objects x, y, z and a criterion c, if $rlt(\mathsf{x}?_c\mathsf{y})=\mathsf{x}\succ_c\mathsf{y}$ and $rlt(\mathsf{y}?_c\mathsf{z})=\mathsf{y}\succ_c\mathsf{z}$, we assume $rlt(\mathsf{x}?_c\mathsf{z})=\mathsf{x}\succ_c\mathsf{z}$ and thus will not ask the crowd to further compare x and z by criterion c. □

To resolve type (ii) contradictions, we enforce the following simple Rule 2.

Rule 2 (Contradiction Resolution by Choosing Outcomes)**.** Consider objects x, y, z and a criterion c. Suppose $rlt(\mathsf{x}?_c\mathsf{y})=\mathsf{x}\sim_c\mathsf{y}$ and $rlt(\mathsf{y}?_c\mathsf{z})=\mathsf{y}\succ_c\mathsf{z}$ are obtained from the crowd. If $rlt(\mathsf{x}?_c\mathsf{z})=\mathsf{z}\succ_c\mathsf{x}$ is obtained from the crowd afterwards, we replace the outcome of this question by $\mathsf{x}\sim_c\mathsf{z}$. (Note that we do not replace it by $\mathsf{x}\succ_c\mathsf{z}$, since $\mathsf{z}\succ_c\mathsf{x}$ is closer to $\mathsf{x}\sim_c\mathsf{z}$.) □

i	$rlt(q_i)$	Derived Results	$O_\surd$	$O_?$	$O_\times$
1-9	$b \succ_m \underline{e}, \underline{c} \sim_a d, \underline{a} \sim_m c$ $\underline{c} \succ_s e, \underline{b} \sim_s d, b \succ_a \underline{a}$ $\underline{d} \succ_m e, \underline{b} \sim_m d, b \succ_s f$		$\varnothing$	{a,b,c,d,e,f}	$\varnothing$
10	$\underline{a} \succ_m d$	$a \succ_m e$			
11	$\underline{c} \succ_a \underline{a}$				
12	$\underline{b} \sim_s c$				
13	$c \succ_m \underline{d}$	$c \succ_m e$			
14-19	$d \succ_s \underline{e}, \underline{e} \sim_a c, \underline{d} \sim_a f$ $\underline{a} \sim_a d, \underline{f} \succ_a \underline{a}, \underline{b} \succ_a e$				
20	$b \succ_a \underline{d}$	$b \succ d$	$\varnothing$	{a,b,c,e,f}	{d}
21-23	$c \succ_s \underline{f}, a \succ_s \underline{e}, f \sim_m b$				
24	$a \succ_s \underline{f}$	$a \sim f$			
25	$\underline{e} \succ_a f$	$b \succ_a f, b \succ_a a$ $e \succ_a a, b \succ f$	$\varnothing$	{a,b,c,e}	{d,f}
26	$\underline{b} \succ_a c$				
27	$\underline{a} \succ_m \underline{b}$				
28	$b \succ_s \underline{e}$	$b \succ e$	$\varnothing$	{a,b,c}	{d,e,f}
29	$c \succ_s \underline{a}$	$c \succ_s e, c \succ a$	$\varnothing$	{b,c}	{a,d,e,f}
30	$\underline{b} \sim_m c$	$b \succ c$	{b}	$\varnothing$	{a,c,d,e,f}

Table 2: RandomQ on Example 1.

i	$rlt(q_i)$	Derived Results	$O_\surd$	$O_?$	$O_\times$
1	$c \succ_s \underline{f}$		$\varnothing$	{a,b,c,d,e,f}	$\varnothing$
2	$\underline{f} \succ_m c$	$f \sim c$			
3 − 4	$\underline{a} \succ_s e, \underline{a} \succ_m e$				
5	$\underline{e} \succ_a \underline{a}$	$a \sim e$			
6 − 7	$c \succ_s \underline{e}, c \succ_m \underline{e}$				
8	$\underline{e} \sim_a c$	$c \succ e$	$\varnothing$	{a,b,c,d,f}	{e}
9	$\underline{b} \succ_s a$	$b \succ_s e$			
10	$\underline{a} \succ_m b$	$a \sim b$			
11	$\underline{d} \succ_s f$				
12	$\underline{f} \succ_m d$	$f \sim d$			
13	$d \succ_s \underline{a}$	$d \succ_s e$			
14	$\underline{a} \succ_m d$	$a \sim d$			
15 − 16	$b \sim_s c, b \sim_m c$				
17	$\underline{b} \succ_a c$	$b \succ c$	$\varnothing$	{a,b,d,f}	{c,e}
18 − 19	$d \sim_s \underline{b}, d \sim_m b$				
20	$b \succ_a \underline{d}$	$b \succ d$	$\varnothing$	{a,b,f}	{c,d,e}
21	$\underline{a} \succ_s f$	$b \succ_s f$			
22	$\underline{a} \sim_m f$				
23	$\underline{f} \succ_a \underline{a}$	$a \sim f$			
24	$b \sim_m f$				
25	$\underline{b} \succ_a f$	$b \succ_a a, b \succ f$	{b}	{a}	{c,d,e,f}
26 − 27	$c \succ_s \underline{a}, \underline{a} \sim_m c$				
28	$c \succ_a \underline{a}$	$c \succ a$	{b}	$\varnothing$	{a,c,d,e,f}

Table 3: RandomP on Example 1.

At every iteration of Alg.1, we choose a question $x?_c y$ from the set of candidate questions. By macro-ordering, when available, the question selection strategy (Alg.2) chooses a candidate question in which $y \notin O_\times$, i.e., it chooses from Q^1_{can}. Otherwise, it chooses from Q^2_{can}. The size of Q^1_{can} and Q^2_{can} can be large. Micro-ordering is for choosing from the many candidates. As discussed in Sec.3, in order to find a short question sequence, the overriding principle of our question selection strategy is to identify non-Pareto optimal objects as early as possible. Guided by this principle, this section discusses several micro-ordering strategies. Since the strategies are the same for Q^1_{can} and Q^2_{can}, we will simply use the term "candidate questions" without distinction between Q^1_{can} and Q^2_{can}.

4. MICRO-ORDERING IN QUESTION SELECTION

4.1 Random Question (RandomQ)

RandomQ, as its name suggests, simply selects a random candidate question. Table 2 shows an execution of the general framework under RandomQ for Example 1. For each iteration i, the table shows the question outcome $rlt(q_i)$. Following the question form $x?_c y$ in Definition 3, the object "x" in a question is underlined when we present the question outcome. The column "derived results" displays derived question outcomes by transitive closure (e.g., $a \succ_m e$ based on $rlt(q_7)$=$d \succ_m e$ and $rlt(q_{10})$=$a \succ_m d$) and derived object dominance (e.g., $b \succ d$ after q_{20}). The table also shows the object partitions ($O_\surd$, $O_?$ and $O_\times$) when the execution starts and when the partitions are changed after an iteration. Multiple iterations may be presented together if other columns are the same for them.

As Table 2 shows, this particular execution under RandomQ requires 30 questions. When the execution terminates, it finds the only Pareto-optimal object b. This simplest micro-ordering strategy already avoids many questions in the brute-force approach. The example clearly demonstrates the benefits of choosing candidate questions only and applying macro-strategy.

4.2 Random Pair (RandomP)

RandomP randomly selects a pair of objects x and y and keeps asking questions to compare them ($x?_c y$ or $y?_c x$) until no such candidate question remains, upon which it randomly picks another pair of objects. This strategy echoes our principle of eagerly identifying non-Pareto optimal objects. To declare an object x non-Pareto optimal, we must identify another object y such that y dominates x. If we directly compare x and y, it requires comparing them by every criterion in C in order to make sure $y \succ x$. By skipping questions according to transitive closure, we do not need to directly compare them by every criterion. However, Property 4 below states that we still need at least $|C|$ questions involving x—some are direct comparisons with y, others are comparisons with other objects which indirectly lead to outcomes of comparisons with y. When there is a candidate question $x?_c y$, it means y may dominate x. The fewer criteria remain for comparing them, the more likely y will dominate x. Hence, by keeping comparing the same object pair, RandomP aims at finding more non-Pareto objects by less questions.

Property 4. Given a set of criteria C and an object $x \in O$, at least $|C|$ pairwise comparison questions involving x are required in order to find another object y such that $y \succ x$.

Proof. The proof is omitted due to space limitations and can be found in the technical report [1]. □

Table 3 illustrates an execution of RandomP for Example 1. The initial two questions are between c and f. Afterwards, it is concluded that $c \sim f$ by Property 3. Therefore, RandomP moves on to ask 3 questions between a and e. In total, the execution requires 28 questions. Although it is shorter than Table 2 by only 2 questions due to the small size of the example, it clearly moves objects into $O_\times$ more quickly. (In Table 2, $O_\times$ is empty until the 20th question. In Table 3, $O_\times$ already has 3 objects after 20 questions.) The experiment results in Sec.5 exhibit significant performance gain of RandomP over RandomQ on larger data.

4.3 Pair with Fewest Remaining Questions (FRQ)

Similar to RandomP, once a pair of objects x and y are chosen, FRQ keeps asking questions between x and y until there is no such candidate questions. Different from RandomP, instead of randomly picking a pair of objects, FRQ always chooses a pair with the fewest remaining questions. There may be multiple such pairs. To break ties, FRQ chooses such a pair that x has dominated the fewest other objects and y has dominated the most other objects. Furthermore, in comparing x and y, FRQ orders their remaining questions (and thus criteria) by how likely x is worse than y on the criteria. Below we explain this strategy in more detail.

Selecting Object Pair Consider a question sequence Q_i so far and FRQ is to select the next question Q_{i+1}. We use $C_{x,y}$ to

denote the set of criteria c such that x?$_c$y is a candidate question, i.e., $C_{\mathsf{x,y}}$=$\{c \in C \mid \mathsf{x}?_c\mathsf{y} \in Q^1_{can}\}$. (We assume Q^1_{can} is not empty. Otherwise, FRQ chooses from Q^2_{can} in the same way; cf. Alg.2.) By Definition 3, the outcomes of these questions are unknown, i.e., $\forall c \in C_{\mathsf{x,y}} : rlt(\mathsf{x}?_c\mathsf{y}) \notin R^+(Q_i)$. Furthermore, if any remaining question (whose outcome is unknown) between x and y is a candidate question, then all remaining questions between them are candidate questions. FRQ chooses a pair with the fewest remaining candidate questions, i.e., a pair belonging to $S_1 = \arg\min_{(\mathsf{x,y})} |C_{\mathsf{x,y}}|$.

The reason to choose such a pair is intuitive. It requires at least $|C_{\mathsf{x,y}}|$ candidate questions to determine y$\succ$x. (The proof would be similar to that of Property 4.) Therefore, $\min_{(\mathsf{x,y})} |C_{\mathsf{x,y}}|$ is the minimum number of candidate questions to further ask, in order to determine that an object is dominated, i.e., non-Pareto optimal. Thus, a pair in S_1 may lead to a dominated object by the fewest questions, matching our goal of identifying non-Pareto optimal objects as soon as possible.

We further justify this strategy in a probabilistic sense. For y$\succ$x to be realized, it is necessary that none of the remaining questions has an outcome x$\succ_c$y, i.e., $\forall c \in C_{\mathsf{x,y}} : rlt(\mathsf{x}?_c\mathsf{y}) \neq \mathsf{x} \succ_c \mathsf{y}$. Make the simplistic assumption that every question x?$_c$y has an equal probability p of not having outcome x$\succ_c$y, i.e., $\forall \mathsf{x}?_c\mathsf{y} \in Q^1_{can}$, $P(rlt(\mathsf{x}?_c\mathsf{y}) \neq \mathsf{x} \succ_c \mathsf{y}) = p$. Further assuming independence of question outcomes, the probability of satisfying the aforementioned necessary condition is $p^{|C_{\mathsf{x,y}}|}$. By taking a pair belonging to S_1, we have the largest probability of finding a dominated object. We note that, for y$\succ$x to be realized, in addition to the above necessary condition, another condition must be satisfied—if $\nexists c$ such that $\mathsf{y} \succ_c \mathsf{x} \in R^+(Q_i)$, the outcome of at least one remaining question should be y$\succ_c$x, i.e., $\exists c \in C_{\mathsf{x,y}} : rlt(\mathsf{x}?_c\mathsf{y}) = \mathsf{y} \succ_c \mathsf{x}$. Our informal probability-based analysis does not consider this extra requirement.

Breaking Ties There can be multiple object pairs with the fewest remaining questions, i.e., $|S_1|>1$. To break ties, FRQ chooses such an x that has dominated the fewest other objects, since it is more likely to be dominated. If there are still ties, FRQ further chooses such a y that has dominated the most other objects, since it is more likely to dominate x. More formally, FRQ chooses a pair belonging to $S_2 = \{(\mathsf{x,y}) \in S_1 \mid \nexists (\mathsf{x'},\mathsf{y'}) \in S_1$ such that $d(\mathsf{x'}) > d(\mathsf{x}) \vee (d(\mathsf{x'}) = d(\mathsf{x}) \wedge d(\mathsf{y'}) > d(\mathsf{y}))\}$, where the function $d(\cdot)$ returns the number of objects so far dominated by an object, i.e., $\forall \mathsf{x}, d(\mathsf{x}) = |\{\mathsf{y} | \mathsf{x} \succ \mathsf{y}$ based on $R^+(Q_i)\}|$. This heuristic follows the principle of detecting non-Pareto optimal objects as early as possible. Note that S_2 may still contain multiple object pairs. In such a case, FRQ chooses an arbitrary pair.

Selecting Comparison Criterion Once a pair (x,y) is chosen, FRQ has to select a criterion for the next question. FRQ orders the remaining criteria $C_{\mathsf{x,y}}$ based on the heuristic that the sooner it understands y$\succ$x will not happen, the lower cost it pays. As discussed before, $|C_{\mathsf{x,y}}|$ questions are required in order to conclude that y$\succ$x; on the other hand, only one question (if asked first) can be enough for ruling it out. Consider the case that x is better than y by only one remaining criterion, i.e., $\exists c \in C_{\mathsf{x,y}} : rlt(\mathsf{x}?_c\mathsf{y}) = \mathsf{x} \succ_c \mathsf{y}$ and $\forall c' \in C_{\mathsf{x,y}}, c' \neq c : rlt(\mathsf{x}?_{c'}\mathsf{y}) = \mathsf{x} \not\succ_{c'} \mathsf{y}$. If FRQ asks x?$_c$y after all other remaining questions, it takes $|C_{\mathsf{x,y}}|$ questions to understand y does not dominate x; but if x?$_c$y is asked first, no more questions are necessary, because there will be no more candidate questions in the form of x?$_c$y.

Therefore, FRQ orders the criteria $C_{\mathsf{x,y}}$ by a scoring function that reflects the likelihood of x's superiority than y by the corresponding criteria. More specifically, for each $c \in C_{\mathsf{x,y}}$, its score is $r_c(\mathsf{x,y}) = r_c(\mathsf{y}) + r'_c(\mathsf{y}) - r''_c(\mathsf{y}) - (r_c(\mathsf{x}) + r'_c(\mathsf{x}) - r''_c(\mathsf{x}))$ where $r_c(\mathsf{y}) = |\{\mathsf{z} \mid \mathsf{z} \succ_c \mathsf{y} \in R^+(Q_i)\}|$, $r'_c(\mathsf{y}) = |\{\mathsf{z} \mid \mathsf{y} \sim_c \mathsf{z} \in R^+(Q_i)\}|$, and $r''_c(\mathsf{y}) = |\{\mathsf{z} \mid \mathsf{y} \succ_c \mathsf{z} \in R^+(Q_i)\}|$. In this scoring function, $r_c(\mathsf{y})$ is the number of objects preferred over y by criterion c, $r'_c(\mathsf{y})$ is the number of objects equally good (or bad) as y by c, and $r''_c(\mathsf{y})$ is the number of objects to which y is preferred with regard to c. FRQ asks the remaining questions in decreasing order of the corresponding criteria's scores. This way, it may find such a question that $rlt(\mathsf{x}?_c\mathsf{y}) = \mathsf{x} \succ_c \mathsf{y}$ earlier than later.

i	$rlt(q_i)$	Derived Results	(x,y), $C_{\mathsf{x,y}}$	$O_\surd$	$O_?$	$O_\times$
			(a,b), $\{s, m, a\}$	∅	{a,b,c,d,e,f}	∅
1	b$\succ_s$a		(a,b), $\{m, a\}$			
2	a$\succ_m$b	a∼b	(a,c), $\{s, a, m\}$			
3	c$\succ_s$a		(a,c), $\{a, m\}$			
4	c∼$_a$a		(a,c), $\{m\}$			
5	c$\succ_m$a	c$\succ$a	(b,c), $\{a, s, m\}$	∅	{b,c,d,e,f}	{a}
6	b∼$_a$c		(b,c), $\{s, m\}$			
7	b∼$_s$c		(b,c), $\{m\}$			
8	b$\succ_m$c	b$\succ_m$a, b$\succ$c	(d,b), $\{a, s, m\}$	∅	{b,d,e,f}	{a,c}
9	b∼$_a$d		(d,b), $\{s, m\}$			
10	b∼$_s$d		(d,b), $\{m\}$			
11	b$\succ_m$d	b$\succ$d	(e,b), $\{a, s, m\}$	∅	{b,e,f}	{a,c,d}
12	b$\succ_a$e		(e,b), $\{s, m\}$			
13	b$\succ_s$e		(e,b), $\{m\}$			
14	b$\succ_m$e	a$\succ_m$e, b$\succ$e	(f,b), $\{a, s, m\}$	∅	{b,f}	{a,c,d,e}
15	b$\succ_a$f		(f,b), $\{s, m\}$			
16	b$\succ_s$f		(f,b), $\{m\}$			
17	b$\succ_m$f	b$\succ$f		{b}	∅	{a,c,d,e,f}

Table 4: FRQ on Example 1.

Table 4 presents the framework's execution for Example 1, by applying the FRQ policy. In addition to the same columns in Tables 2 and 3, Table 4 also includes an extra column to show, at each iteration, the chosen object pair for the next question (x,y) and the set of remaining comparison criteria between them ($C_{\mathsf{x,y}}$). The criteria in $C_{\mathsf{x,y}}$ are ordered by the aforementioned ranking function $r(\cdot)$. At the beginning of the execution, the object pair is arbitrarily chosen and the criteria are arbitrarily ordered. In the example, we assume a?$_s$b is chosen as the first question. After q_2, FRQ can derive that a∼b. Hence, there is no more candidate question between them and FRQ chooses the next pair (a,c). Three questions are asked for comparing them. At the end of q_5, multiple object pairs have the fewest remaining questions. By breaking ties, (b,c) is chosen as the next pair, since only c has dominated any object so far. The remaining criteria $C_{\mathsf{b,c}}$ are ordered as $\{a, s, m\}$, because $r_a(\mathsf{b,c}) > r_s(\mathsf{b,c})$ and $r_a(\mathsf{b,c}) > r_m(\mathsf{b,c})$. The execution sequence terminates after 17 questions, much shorter than the 30 and 28 questions by RandomQ and RandomP, respectively.

To conclude the discussion on micro-ordering, we derive a lower bound on the number of questions required for finding all Pareto-optimal objects (Theorem 2). The experiment results in Sec.5 reveal that FRQ is nearly optimal and the lower bound is practically tight, since the number of questions used by FRQ is very close to the lower bound.

Theorem 2. Given objects O and criteria C, to find all Pareto-optimal objects in O, at least $(|O|-k)\times|C|+(k-1)\times 2$ pairwise comparison questions are necessary, where k is the number of Pareto-optimal objects in O.

Proof. The proof is omitted due to space limitations and can be found in the technical report [1]. □

5. EXPERIMENTS

We designed and conducted experiments to compare the efficiency of different instantiations of the general framework under varying problem sizes. Our experiments used both a real crowdsourcing marketplace and simulations based on a real dataset.

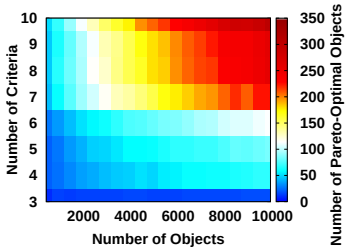
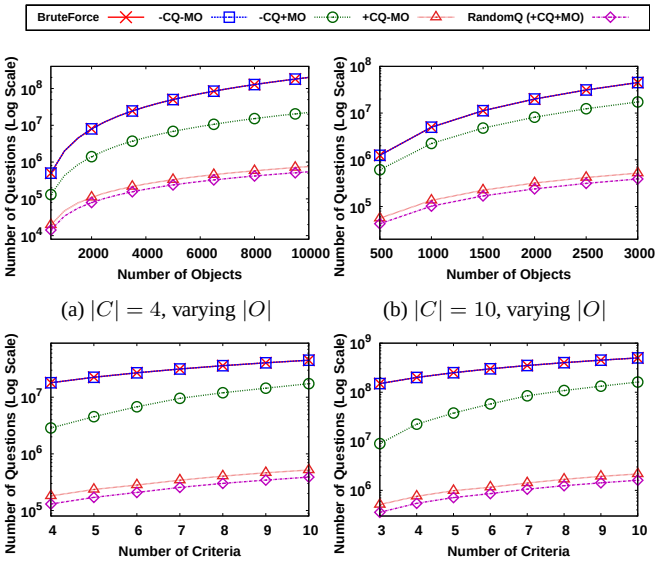


Figure 5: No. of Pareto-optimal objects. Varying $|O|$ and $|C|$.



(a) $|C| = 4$, varying $|O|$ (b) $|C| = 10$, varying $|O|$

(c) $|O| = 3,000$, varying $|C|$ (d) $|O| = 10,000$, varying $|C|$

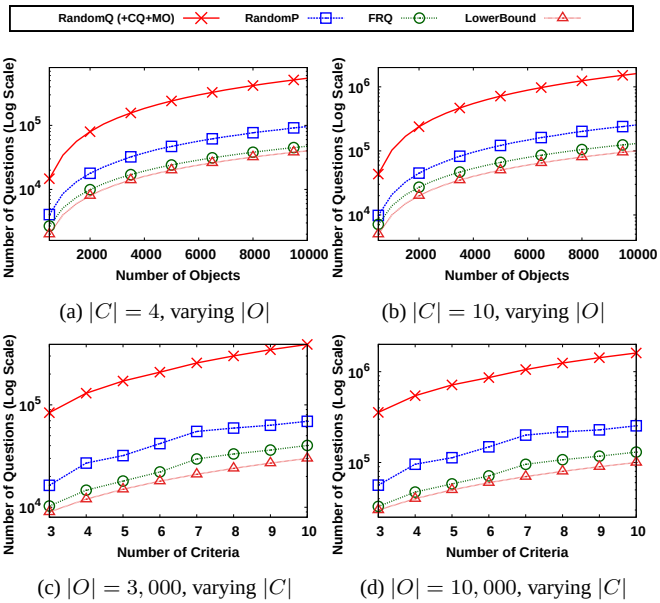
Figure 6: No. of questions by BruteForce and basic methods.

5.1 Efficiency and Scalability

We studied the efficiency and scalability of various instantiations of the general framework. Given the large number of questions required for such a study, we cannot afford using a real crowdsourcing marketplace. Hence, we performed the following simulation. Each object is an NBA player in a particular year. The objects are compared by 10 criteria, i.e., performance categories such as *points*, *rebounds*, *assists*, etc. We simulated the corresponding 10 preference relations based on the players' real performance in individual years, as follows. Consider a performance category c and two objects x=(player1, year1) and y=(player2, year2). x.c is player1's per-game performance on category c in year1 (similarly for y.c). Values in each category c are normalized into the range $[0, 1]$, where 0 and 1 correspond to the minimal and maximal values in c, respectively. Suppose x.c>y.c. We generated a uniform random number v in $[0, 1]$. If $v<1-e^{-(x.c-y.c)}$, we set x$\succ_c$y, otherwise we set x$\sim_c$y. This way, we introduced a perturbation into the preference relations in order to make sure they are partial orders, as opposed to directly using real performance statistics (which would imply bucket orders). Fig.5 shows that the number of Pareto-optimal objects increases by the sizes of both object set O (objects are randomly selected) and criteria set C (the first $|C|$ criteria of the aforementioned 10 criteria).

Effectiveness of candidate questions and macro-ordering

To verify the effectiveness of candidate questions and macro-ordering, we compared five methods—BruteForce, –CQ–MO, –CQ+MO, +CQ–MO, and +CQ+MO. The notation +/– before CQ and MO indicates whether a method only selects candidate questions (CQ) and whether it applies the macro-ordering strategy (MO), respectively. In all these five methods, qualifying questions are randomly selected, i.e., no particular micro-ordering heuristics are applied. For instance, +CQ+MO selects only candidate questions and applies macro-ordering. Hence, it is equivalent to RandomQ. Fig.6 shows the numbers of required pairwise comparisons (in logarithmic scale) for each method, varying by object set size ($|O|$ from 500 to $10,000$ for $|C|$=4 and $|C|$=10) and criterion set size ($|C|$ from 3 to 10 for $|O|$=$3,000$ and $|O|$=$10,000$). The figure clearly demonstrates the effectiveness of both CQ and MO, as taking out either feature leads to significantly worse performance than RandomQ. Particularly, the gap between +CQ–MO and –CQ+MO suggests that choosing only candidate questions has more fundamental impact than macro-ordering. If neither is applied (i.e., –CQ–MO), the performance is equally poor as that of BruteForce. (–CQ–MO uses slightly less questions than BruteForce, since it can terminate before exhausting all questions. However, the difference is negligible for practical purpose, as their curves overlap under logarithmic scale.)



(a) $|C| = 4$, varying $|O|$ (b) $|C| = 10$, varying $|O|$

(c) $|O| = 3,000$, varying $|C|$ (d) $|O| = 10,000$, varying $|C|$

Figure 7: No. of questions by different micro-ordering heuristics.

Effectiveness of micro-ordering

Fig.7 presents the numbers of pairwise comparisons required by different micro-ordering heuristics (RandomQ, i.e., +CQ+MO, RandomP, FRQ) and LowerBound (cf. Theorem 2) under varying sizes of the object set ($|O|$ from 500 to $10,000$ for $|C|$=4 and $|C|$=10) and the criteria set ($|C|$ from 3 to 10 for $|O|$=$3,000$ and $|O|$=$10,000$). In all these instantiations of the general framework, CQ and MO are applied. The results are averaged across 30 executions. All these methods outperformed BruteForce by orders of magnitude. (BruteForce is not shown in Fig.7 since it is off scale, but its number can be calculated by equation $|C|\times|O|\times(|O|-1)/2$.) For instance, for $5,000$ objects and 4 criteria, the ratio of pairwise comparisons required by even the naive RandomQ to that used by BruteForce is already as low as 0.0048. This clearly shows the effectiveness of CQ and MO, as discussed for Fig.6. The ratios for RandomP and FRQ are further several times smaller (0.00094 and 0.00048, respectively). The big gain by FRQ justifies the strategy of choosing object pairs with the fewest remaining questions. Especially, FRQ has nearly optimal performance, because it gets very close to LowerBound in Fig.7. The small

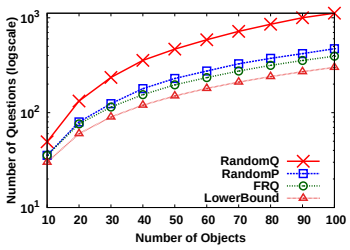


Figure 8: No. of questions by different micro-ordering heuristics. $|C| = 3$, varying $|O|$.

gap between FRQ and LowerBound also indicates that the lower bound is practically tight. The figure further suggests excellent scalability of FRQ as its number of questions grows almost linearly by both $|C|$ and $|O|$.

5.2 Experiments Using a Real Crowdsourcing Marketplace

We also studied the performance of the proposed algorithms using the popular crowdsourcing marketplace Amazon Mechanical Turk (AMT). The task is to compare 100 photos of our institution with regard to *color*, *sharpness* and *landscape*. To obtain the ground-truth data, all $14,850$ possible pairwise questions were partitioned into $1,650$ tasks, each containing 9 questions on a criterion. An AMT crowdsourcer is allowed to perform a task only if they have responded to at least 100 HITs (Human Intelligence Tasks) before with at least 90% approval rate. Furthermore, we implemented basic quality control by including 2 additional validation questions in each task that expect certain answers. For instance, one such question asks the crowd to compare a colorful photo and a dull photo by criterion *color*. A crowdsourcer's responses in a task are discarded if their response to a validation question deviates from our expectation. (236 crowdsourcers failed on this.) The parameters in Equation (1) were set to be k=5 and θ=0.6. Hence, in total $(1,650\times5+236)\times(9+2) = 93,346$ pairwise comparisons were performed by AMT crowdsourcers. We paid 1 cent for each comparison and therefore spent close to $1,000 in total.

The responses to all possible questions provide the ground-truth data. An algorithm execution only needs the responses to a subset of the questions. We randomly selected a subset of photos O ($|O|$ from 10 to 100) and applied various algorithms to find Pareto-optimal photos. Figure 8 shows, for varying $|O|$, the number of questions (in logarithmic scale) required by each micro-ordering strategy. To account for the randomness in RandomP and RandomQ, we repeated these two algorithms, respectively, 30 times, and we reported the average numbers of questions. Confirming the results in Figure 7, FRQ was close to the theoretical lower bound, performing better than the other two methods, and RandomP outperformed RandomQ.

6. CONCLUSIONS

This is the first study on using crowdsourcing to find Pareto-optimal objects when objects do not have explicit attributes and preference relations are strict partial orders. The partial orders are obtained by pairwise comparison questions to the crowd. It introduces an iterative question-selection framework that is instantiated into different methods by exploiting the ideas of candidate questions, macro-ordering and micro-ordering. Experiment were conducted by simulations on large object sets and by using a real crowdsourcing marketplace. The results exhibited not only orders of magnitude reductions in questions against a brute-force approach, but also close-to-optimal performance from the most efficient method.

7. REFERENCES

[1] Technical Report. Details omitted for double-blind reviewing. Anonymized version can be made available upon the request of the program committee.

[2] N. Ailon. Active learning ranking from pairwise preferences with almost optimal query complexity. In *NIPS*, pages 810–818, 2011.

[3] N. Ailon. An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *Journal of Machine Learning Research*, 13(1):137–164, Jan. 2012.

[4] K. J. Arrow. *Social choice and individual values*. Yale University Press, 1951.

[5] S. Borzsony, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[6] M. Braverman and E. Mossel. Noisy sorting without resampling. In *SODA*, pages 268–276, 2008.

[7] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[8] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to Rank with Nonsmooth Cost Functions. In *NIPS*, pages 193–200, 2006.

[9] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking SVM to document retrieval. In *SIGIR*, pages 186–193, 2006.

[10] B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. Here or there: Preference judgments for relevance. In *ECIR*, 2008.

[11] C.-Y. Chan, P.-K. Eng, and K.-L. Tan. Stratified computation of skylines with partially-ordered domains. In *SIGMOD*, 2005.

[12] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *WSDM*, pages 193–202, 2013.

[13] J. Chomicki. Preference formulas in relational queries. *TODS*, 2003.

[14] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In *ICDT*, pages 225–236, 2013.

[15] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing and aggregating rankings with ties. In *PODS*, 2004.

[16] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *CACM*, 35(12):61–70, 1992.

[17] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *HCOMP*, pages 64–67, 2010.

[18] K. G. Jamieson and R. D. Nowak. Active ranking using pairwise comparisons. In *NIPS*, pages 2240–2248, 2011.

[19] W. Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.

[20] N. N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In *CIKM*, pages 759–766, 2009.

[21] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, Mar. 2009.

[22] C. Lofi, K. El Maarry, and W.-T. Balke. Skyline queries in crowd-enabled databases. In *EDBT*, pages 465–476, 2013.

[23] S. Negahban, S. Oh, and D. Shah. Iterative ranking from pair-wise comparisons. In *NIPS*, pages 2483–2491, 2012.

[24] V. Polychronopoulos, L. de Alfaro, J. Davis, H. Garcia-Molina, and N. Polyzotis. Human-powered top-k lists. In *WebDB*, 2013.

[25] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[26] M. E. Rorvig. The simple scalability of documents. *Journal of the American Society for Information Science*, 41(8):590–598, 1990.

[27] D. Sacharidis, S. Papadopoulos, and D. Papadias. Topologically sorted skylines for partially ordered domains. In *ICDE*, 2009.

[28] N. Sarkas, G. Das, N. Koudas, and A. K. Tung. Categorical skylines for streaming data. In *SIGMOD*, pages 239–250, 2008.

[29] L. L. Thurstone. A law of comparative judgment. *Psychological Review*, 34:273–286, 1927.

[30] J. Yi, R. Jin, S. Jain, and A. K. Jain. Inferring users' preferences from crowdsourced pairwise comparisons: A matrix completion approach. In *HCOMP*, 2013.

[31] S. Zhang, N. Mamoulis, D. W. Cheung, and B. Kao. Efficient skyline evaluation over partially ordered domains. In *VLDB*, 2010.