Efficiently Archiving Photos under Storage Constraints

Susan B. Davidson
University of Pennsylvania
susan@cis.upenn.edu

Shay Gershtein
Meta
shayg@meta.com

Tova Milo
Tel Aviv University
milo@post.tau.ac.il

Slava Novgorodov
Meta
slavanov@meta.com

May Shoshan
Tel Aviv University
mayshoshan@mail.tau.ac.il

ABSTRACT

Our ability to collect data is rapidly outstripping our ability to effectively store and use it. Organizations are therefore facing tough decisions of what data to archive (or dispose of) to effectively meet their business goals. We address this general problem in the context of image data (photos) by proposing which photos to archive to meet an online storage budget. The decision is based on factors such as usage patterns and their relative importance, the quality and size of a photo, the relevance of a photo for a usage pattern, the similarity between different photos, as well as policy requirements of what photos must be retained. We formalize the photo archival problem, analyze its complexity, and give two approximation algorithms. One algorithm comes with an optimal approximation guarantee and another, more scalable, algorithm that comes with both worst-case and data-dependent guarantees. Based on these algorithms we implemented an end-to-end system, PHOcus, and discuss how to automatically derive the inputs for this system in many settings. An extensive experimental study based on public as well as private datasets demonstrates the effectiveness and efficiency of PHOcus. Furthermore, a user study using business analysts in a real e-commerce application shows that it can save a tremendous amount of human effort and yield unexpected insights.

1 INTRODUCTION

Although the Big Data revolution has enabled incredible advances in areas such as medicine, commerce, transportation, and science, we are facing an inflection point [34]: The ability to collect data outstrips our ability to effectively use it and will eventually outstrip our ability to store it [9]. Organizations must therefore determine which data to move to larger, cheaper, and typically slower storage (or dispose of) to meet an online storage budget based on factors such as *usage* patterns (workflows) and relative importance of the workflows, the *quality* and *size* of data as well as the *similarity* between different data, requirements on the quality of input data to the workflows, as well as *policy* requirements, e.g. GDPR [15] regulations or document retention regulations for banks. Due to the volume of data, these disposal decisions must be automated.

Image data is an important special case of this problem due to the large file sizes as well as its abundance: it is estimated that up to three billion images are shared on the internet daily[20].

As a concrete example of the need to reduce the amount of image data, consider an e-commerce application such as XYZ[1]. XYZ has a huge archive of images of products that are displayed throughout a hierarchy of landing pages of product categories. For each page, there is a pre-defined subset of images that are relevant for the product category, out of which a small set are displayed. Each image may be relevant for a large number of different pages, and its value may differ between pages. For example, an image of an iPhone which shows the model number may be very valuable for a page which compares different models of iPhones, but not as valuable for a page which displays smartphones in general. Some of the images may also be required to appear on certain pages due to legal contracts (*policies*). For example, a company may require only approved images to be used on pages that are specific to their products. Finally, the landing pages themselves may vary in importance, reflecting the relative popularity of product categories. To speed up the page display, images that are used on pages are stored in a fast-access cache, which is much smaller than the size of the archive. The problem is to find a set of images that can fit in the cache, meet the content and policy requirements of each of the landing pages, and maximize the value over all pages.

The e-commerce example is just one among many instances of what we call the *Photo Archive Reduction problem* (PAR). At a personal level, you may encounter it as the need to delete photos locally on your smartphone to meet some storage budget, relying on cloud storage for your full set of photos. You may have explicitly organized subsets of the photos in albums, or implicitly organized them by labeling photos with the same tag. Image tagging software may also automatically organize photos by features such as date, location and facial recognition. You may require that some of your photos remain in local storage for fast access, for example, photos of your passport, vaccination record and recent favorite photos of your family. Again, the problem is to automate the (local) deletion (and uploading to the cloud) of photos to maximize the value across these pre-defined subsets of photos and satisfy retention requirements to meet storage constraints.

Note that, although we present data disposal in the context of photos (PAR), the general problem shares many of the same characteristics: queries (rather than landing pages) generate subsets of data (rather than photos), queries may vary in terms of importance, data may have varying degrees of importance for queries, and certain data may be required to be retained for legal reasons (see [10]). We choose to focus on the special case of photos in this paper since it is an important problem in e-commerce, for which we can show how to automatically generate the problem inputs and provide an effective user study.

We start by presenting a formal model of PAR, its inputs, and the optimization goal. We then show that, although an exact solution for PAR is not computationally feasible, we can combine methods from submodular optimization and sparsification to give two approximation algorithms. One algorithm comes with

[1]Company name omitted due to privacy considerations.

optimal worst-case approximation guarantees that match our hardness bound. Another algorithm, based on the scheme of [30], has somewhat lower guarantees in the worst case, however, it is much more scalable and also comes with data-dependent bounds that enable providing performance guarantees that far exceed the a priori worst-case guarantee. To further optimize the running time of the latter algorithm we also generalize sparsification techniques presented initially in the more restricted setting of [32], to accelerate nearest-neighbour computations with a small error.

To show the effectiveness of our solution, we evaluate PHOcus using both private e-commerce datasets from XYZ, and public datasets derived from the Open Images dataset [28]. For each dataset, we show how the inputs to PAR can be automatically obtained. Some of the inputs are straightforward – the input set of photos, the size of each photo, retention requirements, and the overall storage constraint. Other inputs – the pre-defined subsets of photos, the relative importance of pre-defined subsets of photos, and the value of a photo for a pre-defined subset – seem difficult to obtain. However, we show that there are, in fact, common practical solutions for obtaining these inputs automatically, not only for the datasets we used but more generally for other applications. In addition to extensive experiments on datasets of different sizes and budgets that show the quality and efficiency of PHOcus, we also discuss concrete examples of datasets and budgets in the e-commerce domain to emphasize the practical importance of our approach.

Finally, since we were fortunate enough to have access to business analysts at XYZ, we were able to perform a user study for a real application. In particular, we evaluated PHOcus within XYZ on several different product categories (suggested by our collaborators based on its business value). Each product category was tested separately using expert XYZ analysts, with its own space constraint. The results show that PHOcus significantly reduces manual work and resulted in better quality solutions. Specifically, analysts reported that it took them less than 10 minutes on medium size datasets using PHOcus compared to hours of manual work invested without PHOcus, and that quality of the solution was 15-25% higher. As a result, they gained unexpected insights in terms of which photos to retain. We also validated our quality metrics using a gold standard based on the domain experts.

Contributions. The contributions of this paper can be summarized as follows:

(1) A formulation of the Photo Archive Reduction problem (PAR), which accounts for pre-defined subsets of photos, the relative importance of a pre-defined subset of photos, the value of a photo for a pre-defined subset, the size (number of bytes) of photos, retention requirements, and an overall storage constraint in terms of size.
(2) A study of the theoretical computational complexity of PAR, and proof of its NP-hardness and approximation hardness. We also provide an optimal approximation algorithm based on the scheme of [45] that also proves the tightness of the hardness bound.
(3) A much more efficient approximation algorithm to solve PAR, based on the submodular optimization scheme of [30] which despite its lower worst-case guarantee also comes with data-dependent bounds that enable providing much better guarantees in practice.
(4) An optimization based on sparsification methods that accelerate nearest neighbor computations, originally proposed in the context of the facility location problem [32].
(5) An implementation of the optimized algorithm in PHOcus, and discussion of how inputs to PHOcus can be automatically derived in many settings.
(6) An extensive experimental study based on eight datasets from two different sources, one publicly available and the other provided by a large e-commerce site, which demonstrate the effectiveness and efficiency of our algorithms.
(7) A real user study using XYZ data and analysts focusing on qualitative metrics and the scope of manual intervention required for implementing our human-in-the-loop approach.

Novelty. As we discuss in the related work (Section 2), similar work has been done in various settings (e.g., [35, 43]). The key novelty of our approach stems from two critical properties that are addressed simultaneously: 1) the budget is more general than a specified number of photos; and 2) the input to the problem can include pre-defined subsets of photos rated by importance. To our knowledge, existing solutions take at most one of the properties into account, but not both at the same time, and cannot be easily adapted to our setting.

We note that the prototype of PHOcus was first demonstrated in [11]. This demo paper provided only a high-level overview of its capabilities and user interface whereas the present paper details the model and algorithms underlying our solution as well as their experimental evaluation.

Paper Organization. We start in Section 2 by reviewing related work. Section 3 formally defines the Photo Archive Reduction problem, PAR, and gives the hardness result. Section 4 presents the optimized algorithm for PAR and its implementation in PHOcus. Section 5 presents experimental results. We conclude in Section 6.

2 RELATED WORK

We start by comparing with related work on image and data summarization, and then describe somewhat related work on materialized view selection. Finally, we discuss complementary work on caching and compression schemes that could offer relevant architectures for efficiently storing and retrieving the selected photos.

Image summarization. The problem of selecting a representative subset of photos has been considered in many different settings [35, 42–44, 46]. The models and solutions proposed select a subset of photos with an objective of *summarization* as opposed to our goal of reducing space usage. This leads to several key differences: First, work on summarization aims to maximize not only coverage, as in our objective, but also the diversity of the summary. As a result, the objective function is not monotone [44, 46] since solutions may be penalized for partial redundancy. In contrast, while diversity is a by-product of the objective in our setting, if the most diverse set includes similar photos then no penalty is incurred as long as the space constraint is satisfied. Therefore, the objective function is monotone, which allows improved approximation guarantees (see Section 3). Second, in summarization the size constraint is on the *number* of photos [42–44, 46], whereas in our setting the bound is on the *sum of sizes* of the photos. Moreover, our input includes a specification of the relevant photo subsets (along with importance weights), instead of attempting to derive this implicitly based on embedding

Table 1: Comparison between various image summarization systems and PHOcus.

Paper	Space Constraint	Coverage Focus	Approximation Guarantee
Canonview [42]	×	×	×
Personal photologs [44]	×	×	×
Submodular mixture [46]	×	✓	✓
Fantom [35]	×	✓	✓
Image corpus [43]	×	×	×
PHOcus	✓	✓	✓

similarity or clustering [42, 44]. Note that the complementary work of [46] focuses solely on learning the importance weights in settings where only the photo subsets are given or automatically derived, and that some summarization works also assume that the input consists of disjoint photo categories [35]. Lastly, while we embed the images into a metric space to derive similarity scores (as done, e.g., in [44]), an important novelty is that the embedding is contextualized by the predefined subset, i.e. there is a different embedding of the same photo for different predefined subsets based on common contextual embedding methods (e.g., [26, 47]).

Table 1 compares our proposed system, PHOcus, to a representative selection of image summarization solutions (Canonview [42], Personal photologs [44], Submodular mixture [46], Fantom [35] and Image corpus [43]) along the dimensions discussed above: whether the space constraint is the number of photos (×) or the sum of sizes (✓), whether or not the coverage focus of the algorithm can be specified (this is done in PHOcus via the predefined subsets and weight parameters), and whether or not it provides a worst-case approximation guarantee.

Other summarization settings. In *text summarization* [3], the inherent semantic constraints require using fundamentally different models from image summarization, and solutions are based on state-of-the-art NLP methods [1]. In summarization of *network traffic*, submodular optimization techniques are commonly used [2] as in our solution. However, the scale and summary objectives are much different and the focus is on streaming algorithms [5].

View materialization. In *Materialized View Selection* (*MVS*) [17, 27, 36] the goal is to materialize a set of views that optimize the execution time of an expected query load while adhering to space constraints. At a high level, *MVS* is similar to *PAR*, with views having roughly the same role as predefined subsets, and tuples corresponding to photos. Moreover, as in our solution, submodular optimization techniques are often used [22]. However, there are several important differences. Most importantly, in *MVS* there is no notion of similarity so tuples cannot be omitted from the view. In *PAR*, however, space usage is reduced by identifying partial redundancies, as the utility of retaining an image is partially assessed by how similar it is to other photos in given contexts. Additionally, in *MVS* the view result (or a superset of the result) is cached [18]), therefore the same tuple may appear in multiple views. In contrast, in *PAR* there is no analogous redundancy as we are not precomputing query results, but rather removing some of the data over which the query results are computed.

Caching. For optimizing retrieval time there is much research on caching schemes (e.g., multi-level caching [48]) both on the server/cloud, as in the *MVS* setting above, and on the client-side, as in *semantic caching* [40, 41], where the cache includes both retrieved query results and semantic descriptions that may help avoid unnecessary future queries. We note that these caching solutions are not relevant for *PAR*, since similarities are not leveraged to save space, i.e., the decision of which items to retain is not based on any redundancy in the data, but on frequency/recency of the use.

Secondary storage solutions. Our research is focused on *identifying* the optimal set of photos to retain. What is done subsequently with the removed photos is outside the scope of our model. One may either delete these photos or archive them in secondary storage. In the latter case, there are many relevant complementary works on architectures that optimize specific trade-offs between securing privacy, efficient space consumption, and optimizing retrieval time. For example, there are recent works on approximate image storage [49] and compression schemes for cold storage [12].

Getting rid of data. Our work is related to a line of work that address the general problem of getting rid of unnecessary/unused data. This problem has interest in the DB community for the past several years (e.g., [10, 23, 24, 34]). Specifically, [24] introduces a notion of databases with amnesia - DBMSs that selectively forget tuples (by marking them inactive) for the sake of storage management and responsiveness. Different strategies to retain information while forgetting tuples are examined, and a number of metrics for quantifying information retention are defined. [34] is a vision paper that discusses the logical, algorithmic, and methodological foundations required for the systematic disposal of large-scale data, for constraint enforcement and for the development of applications over the retained information. In particular, it highlights new research challenges and potential reuse of existing techniques. Our paper can be viewed as a first step in implementing this vision.

Data reduction in E-Commerce. Our work complements a highly related line of work that focuses on data reduction in e-commerce. Large e-commerce companies deal with a lot of data of different types and from various sources that needs to be efficiently and effectively managed and (at times) disposed [10]. Working with such data frequently requires solutions that are specific for the particular e-commerce application. Some recent work in this domain includes efficient inventory management and reduction to reduce the maintenance costs [16]; summarization of large amounts of texts, such as reviews [31]; and generation of short product descriptions that can fit on mobile devices screens [39].

3 MODEL AND HARDNESS RESULT

As described in the introduction, our model must capture factors such as *usage* patterns (which we capture as *pre-defined subsets of photos*) and the relative importance of the pre-defined subsets, the *quality* and *size* of photos as well as the *similarity* between different photos, the *relevance* of input photos to each pre-defined subset, as well as *policy* requirements. These requirements were derived from e-commerce problems, but have broad applicability within the context of photos. In this section, we formalize the model and state the optimization goal of PAR. We then show that an optimal solution to PAR is not computationally feasible. We also show through an example that many of the inputs to our model are straightforward to obtain, and will discuss in Section 5.1 how other inputs can be automatically obtained.

3.1 Model

The inputs to our problem are as follows: The set of photos is denoted by P, and the set of photos that must be retained due to policy requirements by S_0. The number of input photos is denoted by $n = |P|$. The cost of each photo is its size (in terms of the disk space required to store it) and is given by the cost function $C : P \mapsto \mathbb{R}_+$. For a subset of photos $S \subseteq P$, we denote its cost, which is defined as the sum of the individual costs, by $C(S) = \sum_{p \in S} C(p)$. The set of pre-defined subsets of photos is denoted by $Q \subseteq 2^P$.

The function $\mathcal{W} : Q \mapsto \mathbb{R}_+$ assigns a positive weight to each pre-defined subset, to reflect its importance (i.e. how valuable it is to retain the photos in this subset). Given any specific $q \in Q$, the relevance function $\mathcal{R} : Q \times P \mapsto \mathbb{R}_+$ assigns a score to each photo p in q that reflects how relevant it is for this pre-defined subset (the score for photos in $P \setminus q$ is 0). The relevance scores of all photos in a pre-defined subset are normalized (in advance) to sum up to 1: $\forall q \in Q : \sum_{p \in q} \mathcal{R}(q, p) = 1$. A (contextualized) similarity function $\mathcal{SIM} : Q \times P \times P \mapsto [0, 1]$ produces a normalized measure of the similarity of any given pair of photos, w.r.t. a given pre-defined subset. Note that the similarity of any pair of photos differs based on the pre-defined subset, which is referred to as the *context* of the similarity. In particular, if at least one of the two photos is not in the context subset, then the similarity score is defined to be 0. Moreover, the similarity of any photo to itself is 1. However, a similarity score of 1 does not necessarily imply that the pair of photos are identical. Lastly, the storage budget B is an upper bound on the cost of the solution.

Putting everything together, the input for a PAR instance consists of the tuple $\langle P, S_0, Q, C, \mathcal{W}, \mathcal{R}, \mathcal{SIM}, B \rangle$. Although the parameter set seems large, each one is needed to capture the the real world setting of efficient photo archival. As we later show in Section 5, solutions that ignore or simplify some of the parameters (e.g., simpler SIM function) achieve lower quality.

Example 3.1. Consider the e-commerce application introduced earlier. P is the image archive. S_0 is the set of images that are required to appear on certain pages due to legal contracts. Q represents the set of landing pages (sets of images). C gives the size of each image. $\mathcal{W}$ represents the relative importance of a landing page. $\mathcal{R}$ represents the relevance of each image to the landing page. Finally, $\mathcal{SIM}$ is the similarity between images. Note that P, S_0, C and B are straightforward to obtain; we will show in Section 5.1 how the inputs Q, $\mathcal{W}$, $\mathcal{R}$ and $\mathcal{SIM}$ can be efficiently obtained. □

Objective. Given an input instance, the goal is to find the "best" subset of photos to retain subject to the online budget B, i.e a solution $S \subseteq P$ such that $C(S) \leq B$, $S_0 \subseteq S$, and the score of S is maximized. Informally, the score of a subset S of photos for a pre-defined subset q is how relevant the best match in S is for each photo in q, as measured by $\mathcal{SIM}$. For example, if $q \subset S$ then S has a perfect score for q. The overall score of S is taken over all pre-defined subsets in Q.

More formally: given a solution $S \subseteq P$, a pre-defined subset $q \in Q$, and a photo $p \in q$, the *nearest neighbour* of p in S w.r.t. q is the most similar photo in $S \cap q$ to p, denoted by $NN(q, p, S) = \arg\max_{p' \in S \cap q} \mathcal{SIM}(q, p, p')$ (if there are several such photos, we choose one arbitrarily). Observe that if $p \in S$, then $NN(q, p, S) = p$.

We define the *score* of a solution S w.r.t. a given pre-defined subset $q \in Q$ as

$$\mathcal{G}(q, S) = \sum_{p \in q} \mathcal{R}(q, p) \cdot \mathcal{SIM}(q, p, NN(q, p, S)).$$

Recall that the relevance scores of all photos in a pre-defined subset q are normalized to sum up to 1 ($\sum_{p \in q} \mathcal{R}(q, p) = 1$) so that the scores do not depend on the size of q.

Abusing notation, the overall score of a solution S is defined as the weighted sum of the scores w.r.t. pre-defined subsets:

$$\mathcal{G}(S) = \sum_{q \in Q} W(q) \cdot \mathcal{G}(q, S).$$

Thus, the optimization problem of *PAR* is to produce a solution that maximizes the objective function. That is, the goal is to produce

$$\arg\max_{S_0 \subseteq S \subseteq P, C(S) \leq B} \mathcal{G}(S).$$

Unfortunately, we can show that providing a solution which optimizes this score is computationally infeasible.

3.2 Hardness Result

To show the approximation hardness of the optimization problem of PAR, we use the classical Maximum Coverage problem which is provably hard to approximate [14].

Definition 3.2. Given a set of sets over a universe of elements and an integer k, the goal of the **Maximum Coverage problem (*MC*)** is to select k sets such the cardinality of their union is maximized.

Theorem 3.3. *[14] The MC problem is NP-hard to approximate beyond a* $(1 - 1/e) \approx 0.63$ *factor.*

Theorem 3.4. *The PAR problem is NP-hard to approximate beyond a* $(1 - 1/e)$ *factor.*

Proof. We next present the proof sketch, based on a reduction from the Maximum Coverage (*MC*) problem (Definition 3.2). Concretely, in the derived *PAR* instance, every set s becomes a photo p_s, and for each element e in the *MC* instance we add to the *PAR* instance a pre-defined subset q_e, that consists of all the photos that correspond to the sets that contain e. The cost of each photo is 1 and the budget is $B = k$. The weight of each pre-defined subset is 1. For each pre-defined subset q_e, the relevance score of every photo in it is $1/|q_e|$. The similarity of any two photos appearing in the same pre-defined subset is 1 (and 0 otherwise). This implies that selecting at least one photo of a given pre-defined subset ensures a maximum score of 1 for that pre-defined subset (same as selecting at least one set that covers a given element is sufficient to "fully cover" the element in the *MC* setting). It is straightforward to show that the two problem formulations are equivalent, and any α-approximation of the *PAR* instance implies the same performance ratio for the *MC* instance (by selecting all the sets that correspond to the photos in the *PAR* solution). □

In light of this hardness result, we give in the next section an algorithm which achieves a $(1 - 1/e)$-approximation of *PAR*. This is therefore optimal for a PTIME algorithm.

Algorithm 1: *MainAlgorithm*

```
res1 = LazyGreedy(UC)
res2 = LazyGreedy(CB)
return argmax(res1, res2)
```

4 ALGORITHM

We now present our algorithm for *PAR*, which combines methods from submodular optimization and data sparsification. We first prove that the *PAR* objective satisfies properties that allow employing the submodular optimization scheme of [45] to derive a $(1-1/e)$-approximation, which is optimal according to Theorem 3.3. The algorithm in [45], however, is not scalable, and we therefore leverage the much more efficient algorithm of [30] which was originally proposed in the context of outbreak detection. Importantly, while the worst-case guarantee of the latter algorithm is $(1-1/e)/2$, [30] also proposes a method to provide online bounds on the output of *any* algorithm, which we leverage to empirically show that the performance ratio in practice is far better than this bound. To further optimize the efficiency of this algorithm, we extend it by adapting sparsification methods that accelerate nearest neighbor computations, originally proposed in the context of the facility location problem [32]. We show empirically that sparsification significantly improves the running time, with only a negligible effect on the performance ratio.

The remainder of this section is organized as follows. We first prove the theoretical properties of *PAR* necessary to apply the submodular optimization algorithms stated above. We then provide the implementation of the algorithm of [30], adapted to our context. Afterwards, we present the complete *PAR* algorithm, that extends algorithm of [30] with sparsification methods. Finally, we demonstrate the operation of the algorithm on a small, yet real world, example.

4.1 Submodular optimization algorithms

To apply the approximation schemes of [45] and [30], we first need the following definitions and results relating to set functions ($f: 2^U \rightarrow \mathbb{R}$, given a universe U).

Definition 4.1. f is **nonnegative** if $\forall S \subseteq U: f(S) \geq 0$.

Definition 4.2. f is **monotone** if $\forall S \subseteq U, \forall v \in U$: $f(S \cup \{v\}) \geq f(S)$.

Definition 4.3. f is **submodular** if $\forall S \subseteq T \subseteq U, \forall v \in U$: $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$.

Theorem 4.4. *[25, 45] Let f denote a nonnegative, monotone, and submodular set function over a universe U, where each element in the universe has a positive cost. There exists a PTIME $(1-1/e)$-approximation algorithm for the problem of maximizing f, subject to the Knapsack constraint that the sum of the costs of the selected elements does not exceed a specified budget upper bound B.*

We next prove that the *PAR* objective satisfies all the properties defined above.

Lemma 4.5. *The objective function of PAR is nonnegative, monotone and submodular.*

Proof. The function is nonnegative by definition. The function is also monotone nondecreasing, as adding a photo to a solution (without exceeding the budget) may only increase the similarity score of the most similar photo of each non-selected photo. Finally, to show that the function is submodular, consider two sets $S \subset T \subseteq P$ and a photo $p \in P$. Clearly, the increase in the overall score resulting from adding p to T is bounded by the analogous increase of adding p to S, as the set of photos in $P \setminus \{S \cup \{p\}\}$ for which p is the nearest neighbor is a superset of the analogous photos in $P \setminus \{T \cup \{p\}\}$. □

Algorithm 2: *LazyGreedy*(*type*)

```
S ← S_0
B ← B − C(S_0)
foreach p ∈ P do
    δ_p ← ∞
while ∃p ∈ P \ S : C(S ∪ {p}) ≤ B do
    foreach p ∈ P \ S do
        cur_p ← False
    while True do
        if type = UC then
            p* ← argmax_{p∈P\S, C(S∪{p})≤B} (δ_p)
        if type = CB then
            p* ← argmax_{p∈P\S, C(S∪{p})≤B} (δ_p / C(p))
        if cur_{p*} then
            S ← S ∪ p*
            break
        else
            δ_p ← G(S ∪ {p}) − G(S)
            cur_p ← True
return S
```

From Lemma 4.5 and Theorem 4.4, it follows that we can use the algorithm in [45] to derive a $(1-1/e)$-approximation of *PAR*, which, following Theorem 3.4, is optimal for a PTIME algorithm.

Theorem 4.6. *There exists a tight PTIME $(1-1/e)$-approximation algorithm for PAR.*

4.2 A more efficient algorithm

The time complexity of the algorithm in [45] is $\Omega(B \cdot n^4)$, where B is the budget and n is the number of photos, as this is the number of times it evaluates the gain of adding a specific photo to the solution. We therefore leverage a more efficient algorithm for maximizing a submodular function, subject to a knapsack constraint, proposed originally in [30] in the context of outbreak detection in networks. This algorithm is significantly more scalable, since the number of times it evaluates the gain from adding a photo is $O(B \cdot n)$, and (more importantly) it uses lazy evaluation; these were shown to improve the running time by a factor of 700. While using this algorithm reduces the worst-case approximation guarantee to $(1-1/e)/2$, we also leverage the method provided in [30] to bound the performance ratio a posteriori, and show in Section 5 that the performance in practice far exceeds the worst-case bound.

A high-level description of the approximation scheme of [30] adapted to our context is shown in Algorithm 2, and we explain its operation next. The complete algorithm, where we also use sparsification methods, is presented in the following subsection.

Algorithm 1 runs two greedy procedures separately and outputs the best solution. The procedures are shown in Algorithm 2, and their implementations only differ based on the argument of the *type* parameter passed from Algorithm 1. Each of the two procedures iteratively selects the photo that produces the highest marginal gain based on a specified objective (there is a different objective for each procedure) and adds it to the current solution,

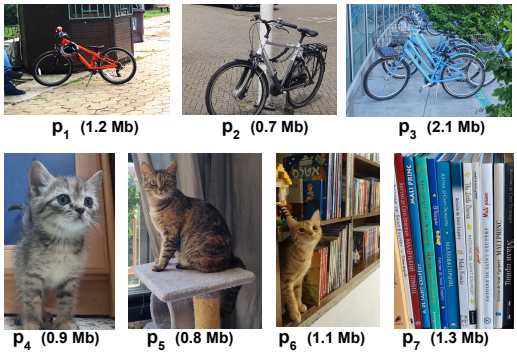


Figure 1: Sample input to PAR.

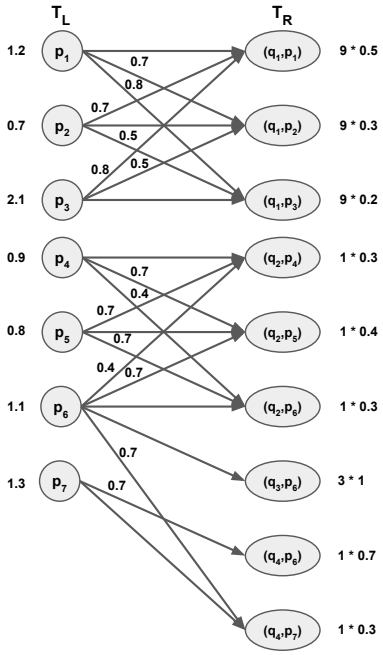


Figure 2: GFL formulation of PAR input.

until no photos can be added without exceeding the budget. For $type = UC$, the algorithm treats all photos as if they have *unit cost* (i.e., it ignores the differences in cost) and selects in each iteration the photo whose addition to the solution selected so far maximally increases the value of the objective function $\mathcal{G}$. In contrast, for $type = CB$ the algorithm uses a *cost-based* approach, and selects in each iteration the photo that maximizes the ratio between the increase in the objective function $\mathcal{G}$ and its cost.

4.3 Input sparsification

We now explain how to optimize the efficiency of Algorithm 1 using a sparsification preprocessing step that reduces the time necessary to compute each evaluation of the marginal gain of selecting a given photo. Specifically, we modify the function SIM by rounding down to zero all photo similarities that are below a specified threshold τ, so that fewer neighbors are considered for each photo in nearest neighbor computations. Importantly, we prove a data-dependent bound on the error incurred by this sparsification, which generalizes results proven in [32] for the Facility Location problem, which is equivalent to a special case of our problem. This bound to some extent allows one to control the trade-off between efficiency and accuracy. To further facilitate efficiency, we use Locality Sensitive Hashing (LSH) [21] to perform the sparsification without first computing all the pairwise similarities. This randomized method allows us, in roughly linear time, to find with high probability almost all pairs whose similarity exceed a specified threshold. Specifically, to perform this optimization, the calculation of $\mathcal{G}$ in line 17 of Algorithm 2 is modified in order to reduce the number of required comparisons. A step-by-step demonstration of this algorithm can be found in Section 4.4.

We first prove worst-case data-dependent bounds on the ratio of the optimal solution to the original instance and the optimal solution of the τ-sparsified instance, where all similarities below τ are rounded-down to 0.

To prove the error bounds on the sparsification, we use an equivalent formulation of our problem which generalizes the Facility Location problem [7]. To select the desired trade-off between the degree of the sparsifcation and the worst-case accuracy loss, different values of the threshold τ can be tested. Moreover, we show in the experiments that the actual accuracy loss in practice is much smaller then the above worst-cases bounds indicate.

Generalized Facility Location (GFL) problem. Consider a weighted bipartite graph over the node set $T_L \cup T_R$, where $T_L = P$ and $T_R = \{(q,p)|p \in q\}$. The edges are a subset of $T_L \times T_R$, constructed as follows: For every q and every $p_1, p_2 \in q$, there exist two edges that connect p_1 (in T_L) to (q,p_2) (in T_R) and p_2 to (q,p_1), each of weight $\mathcal{SIM}(q,p_1,p_2)$; if $p_1 = p_2$ then only one edge (of weight 1) is added. The weight of each node $p \in T_L$ is $w_L(p) = C(p)$ and the weight of each node $(q,p) \in T_R$ is $w_R(q,p) = \mathcal{W}(q) \cdot \mathcal{R}(q,p)$. For simplicity of presentation, we define $w_E(S,(q,p))$ as the maximum weight of an edge incident to (q,p) in the subset S. The optimization goal over this input is to find a subset $S \subseteq T_L$ that maximizes $F(S) = \sum_{(q,p)\in T_R} w_E(S,(q,p))$ and whose total weight is at most B.

Example 4.7. An example of input to PAR is depicted in Figure 1. It consists of 4 predefined subsets (q_1 to q_4) that correspond to 4 natural language queries (shown next to the subsets) that were executed over a small set of images (shown at the top with their sizes). The relative importance of each subset is given as w. The lower part of the figure contains the pairwise photos similarities and the relevance of each photo to the predefined subset. Note that the relevance scores are normalized, i.e., their sum over each query equals 1. The GFL formulation of the sample input is depicted in Figure 2. Note that all edges that connect p_i to (q,p_i) are of weight 1, and are omitted from the figure.. It is easy to verify that this GFL formulation is equivalent to the original formulation provided in Section 3.

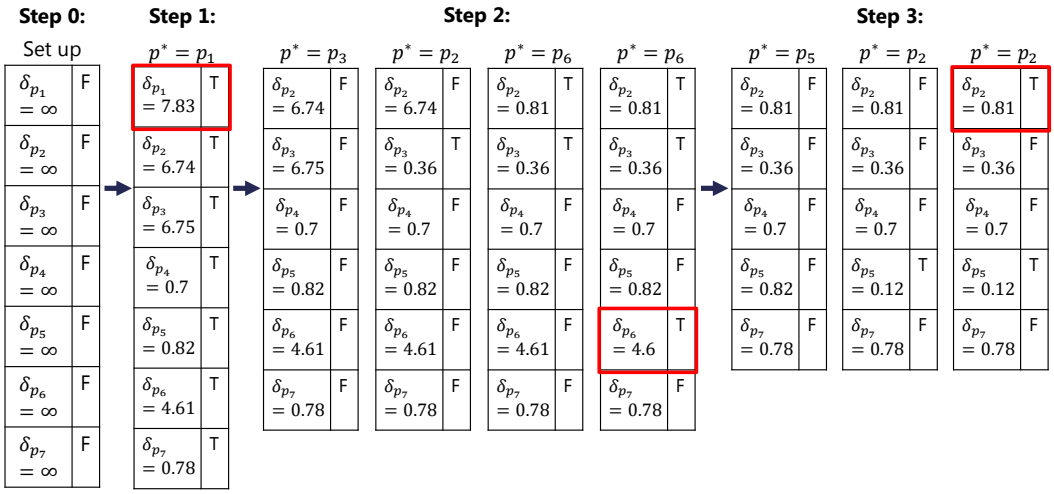


Figure 3: Step-by-step operation of Algorithm 2.

We note that if all weights are equal to 1, then we get the Facility Location problem as formulated in [32]. The following theorem generalizes the error bounds provided in [32] for τ-sparsification.

Theorem 4.8. *Let O_τ be the optimal solution to a τ-sparsified GFL instance. Let $W_R = \sum_{(q,p)\in T_R} w_R(q,p)$. If there exists $S \subseteq T_L$ such that $\sum_{p\in S} w_L(p) \leq B$ and $\sum_{v\in N_\tau(S)} w_R(v) = \alpha * W_R$ (where $N_\tau(S)$ denotes the set of neighbors of S in the τ-sparsified instance) then*

$$F(O_\tau) \geq \frac{1}{(1+1/\alpha)} \cdot OPT$$

Proof. Let O be the optimal solution to the original problem. Let F_τ and $\bar{F}_\tau$ denote the functions derived by restricting F to consider only the edges of weight at least τ and all remaining edges, respectively. By definition,

$$F(O) = F_\tau(O) + \bar{F}_\tau(O)$$

and

$$\bar{F}_\tau(O) \leq \tau \cdot W_R.$$

From the assumption on the set S we have

$$F_\tau(S) \geq \alpha \cdot \tau \cdot W_R.$$

Overall, we get

$$\begin{aligned} F(O) &= F_\tau(O) + \bar{F}_\tau(O) \\ &\leq F_\tau(O) + \tau \cdot W_R \\ &\leq F_\tau(O) + \frac{1}{\alpha} \cdot F_\tau(S) \\ &\leq F_\tau(O_\tau) + 1/\alpha \cdot F_\tau(O_\tau) \\ &\leq (1 + 1/\alpha) \cdot F_\tau(O_\tau) \end{aligned}$$

□

Note that the second to last inequality follows from the fact that O_τ is by definition the optimal solution with respect to $F_\tau(\cdot)$.

Finding a set S that maximizes the sum of weights of the neighbors of S, $N(S)$, is exactly the Budgeted Maximum Coverage problem [25]. Note that the solution to this problem can be reasonably approximated via schematically the same algorithms as the PARproblem, however, the running time is much faster, as in addition to the sparsification, to evaluate each solution one only computes the total weight of covered items, without taking into account similarities or computing nearest neighbors at each iteration (also recall that this faster sub-algorithm is executed offline for the sake of evaluation if one seeks a posteriori data-dependent bounds for the sparsification).

In our solution, we measure similarity between image embeddings using a cosine similarity function (see Section 5.1), a common similarity metric for vector embeddings and images in particular [38]. As mentioned earlier, one way to sparsify the input is to first compute all the pairwise cosine similarities between any two photos for each predefined subset, and then round down to 0 the similarities below τ (equivalent to removing these edges from the GFL formulation). However, to optimize the running time for larger instances, we employ a randomized method based on LSH that hashes each embedding vector a constant number of times, and only considers similarities between vector pairs corresponding to hash collisions. Given the correct tuning of parameters, this yields with probability arbitrarily close to 1 all vectors pairs of of similarity at least τ, except for an arbitrarily small fraction of pairs. Since cosine similarity is our measure of similarity, we use the LSH method SimHash [6] which is based on random projections, since it is easy to compute and is widely used for this purpose (e.g., [4, 13, 29]). This LSH approach allows us to identify (almost) all sufficiently similar pairs in roughly linear time, which is preferable when there are many large predefined subsets. Empirical results pertaining to the speed-up gains and incurred errors of this method are provided in Section 5.

4.4 Algorithm Demonstration

We conclude this section by demonstrating the operation of Algorithm 2. For simplicity, we demonstrate it only for the case where $type = UC$; for $type = CB$, the steps are identical, and the only difference is the exact formula for computing δ_p.

Figure 3 depicts the step by step operation of the algorithm on the input presented in Figure 1. The main data structure used is a priority-queue (PQ) that allows the photo with the largest marginal gain to be efficiently found. The algorithm starts

(Step 0) by initializing the marginal gain of each photo, δ_p, to ∞, and setting an indicator of whether or not δ_p was recalculated since the last solution update, $curr_p$, to False. Next, the algorithm computes the values of δ_p and updates all $curr_p$ to True. Finally, it picks the p with highest value of δ_p, shown as p^*, and adds it to the solution. In Step 1, p^* is p_1. Each time a new element is added to the solution set, all values of $curr_p$ are set to False (shown at the beginning of Steps 2 and 3). This indicates that the values of δ_p may be outdated and therefore require recalculation, since the marginal gain of an additional photo depends on what is already in the solution. Note that due to submodularity of the objective function, the value of δ_p for each of the photos can only decrease. Hence if the newly calculated value δ_p is at the top of the priority queue (i.e., it is the largest among all other photos) it can be selected without recalculating this value for the rest of the photos. To do this, the algorithm takes the p with the highest value of δ_p, removes it from the PQ, recalculates δ_p, updates $curr_p$ to True, and adds p back to the PQ. If the updated value is still the highest (i.e., it is p^*) the algorithm adds it to the solution and continues to the next step. Otherwise, it repeats for the next highest δ_p. In Step 2, the algorithm tests p_3 and then p_2, but neither are selected since they do not have the highest the highest δ_p after recalculation; intuitively, both are pretty well covered by p_1 which has already been selected. Therefore p_6 is selected as p^*, since after recalculation it still has the maximum value of δ_p, and Step 2 ends. In Step 3, p_5 is initially selected, but after recalculation it turns out that p_2 is again the highest. After recalculation, p_2 still has the highest value, and Step 3 ends with p_2 being selected as p^*. We showcase here only the first few steps, but the algorithm would continue adding to the solution until there are no more photos or the budget is exceeded.

5 EXPERIMENTAL EVALUATION

We begin this section by giving an overview of the architecture of our system, PHOcus. We continue by explaining how to derive and preprocess the input to best exploit our algorithms. We then describe the datasets (both private and public) used for the experiments, the algorithms we compared, and the quantitative evaluation methods, followed by the evaluation results. Lastly, we describe the setup, methodology and findings of a user study, focusing on qualitative metrics and the scope of manual intervention required for implementing our human-in-the-loop approach.

5.1 System Architecture

The high-level architecture of PHOcus is shown in Figure 4, and consists of a user interface and two modules: the *Data Representation Module*, which prepares the input to PAR; and the *Solver*, which runs the optimization algorithm for PAR. *Solver* is implemented using Python and Flask.

The user interacts with the system via the UI to specify the input, as described below. Note that P, S_0, C and B are straightforward to obtain; we show below that the inputs $\mathcal{Q}$, $\mathcal{W}$, $\mathcal{R}$ and $\mathcal{SIM}$ can also be efficiently obtained.

The pre-defined subsets Q and relevance scores $\mathcal{R}$ may be specified in one of three ways:

(1) **Directly**: each photo is tagged with all the subsets that include it. The relevance scores, which are assumed to be uniform by default, may be adjusted via the UI;

Table 2: Datasets.

Dataset name	# Photos	# Predefined subsets
P-1K	1000	193
P-5K	5000	1409
P-10K	10000	3955
P-50K	50000	14326
P-100K	100000	33721
EC-Fashion	18745	250
EC-Electronics	22783	250
EC-Home & Garden	19235	250

(2) **Queries**: users provide queries such as ("Paris vacation"), and the subsets are computed via the PHOcus search engine. The confidence scores of the engine are then converted into the relevance scores; or
(3) **Automatic tagging**: Subsets are derived using automatic tagging methods. The weights for subsets derived by all methods may be adjusted using a dedicated UI.

The Data Representation module receives this input and, if it contains user-provided queries, uses an internal search engine to compute the pre-defined subsets and relevance scores. It then normalizes the relevance scores (as described in Section 3), and derives the contextualized similarities ($\mathcal{SIM}$). This is done using the approach in [44], which computes the distance between two photos based on both quantitative and categorical attributes that are derived via standard methods, including, e.g., reading the EXIF metadata and generating visual words via the SIFT algorithm [33].

Another adaptation one could do to our setting is normalizing the distances differently for each context subset. This is done by dividing all distances by the maximum distance between any two photos in the context. Intuitively, this emphasizes smaller variations across photos for more granular predefined subsets. For example, when searching for photos of all "trips", having many photos of the same trip to Paris in 2016 may seem redundant. However, when searching for this specific trip, photos are only redundant if they are very similar in more specific features (e.g., many photos of Eiffel Tower taken on the same day).

The complete input is then passed to *Solver*, which runs the optimization algorithm described in Section 3.

Example 5.1. Returning to the e-commerce application, which we have implemented for use within XYZ: Q is calculated directly from the set of landing pages, each of which is defined by a set of relevant images for the page as well as a title (e.g., "Nike red shirts", "Samsung smartphones" or "shoes"). The relative importance of a landing page, $\mathcal{W}$, is calculated based on the landing page popularity, i.e. the number of visits in the last 90 days, normalized by sum of all visits to all landing pages. The relevance function, $\mathcal{R}$, which reflects how relevant a photo p is for a pre-defined subset q, is computed based both on the quality of the image (using ML model for image embedding, e.g., [8]) and the relevance score of the product represented in the image (using the product title and the retrieval score). Finally, the similarity between images, $\mathcal{SIM}$, is calculated using cosine similarity between image embeddings. The embeddings are based on the ResNet-50 network [19] and is trained on over 50M product images from the XYZ site.

5.2 Experimental Setup

We evaluated *Solver* extensively using a server with 128GB RAM and 32 cores. The experiments used eight datasets from two

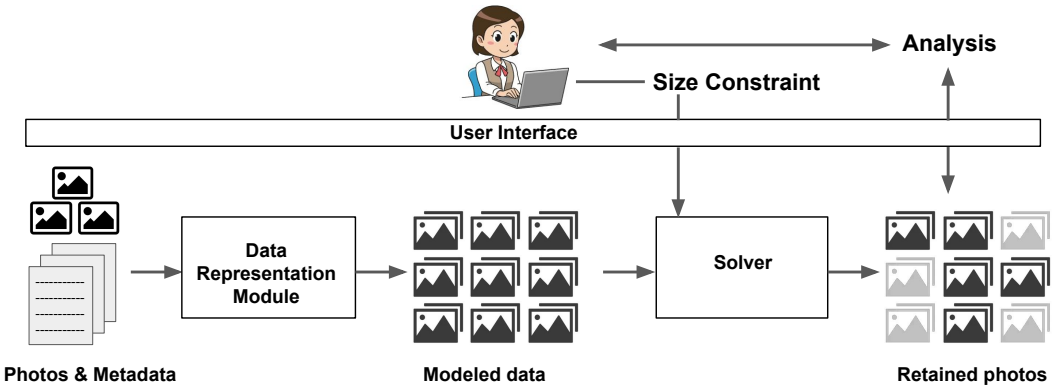


Figure 4: PHOcus - System Architecture

sources, one publicly available and the other provided by XYZ. We used 6 different algorithms to form a comparison under a wide range of different parameters such as dataset size and budget.

Datasets. The datasets used for our evaluation are:

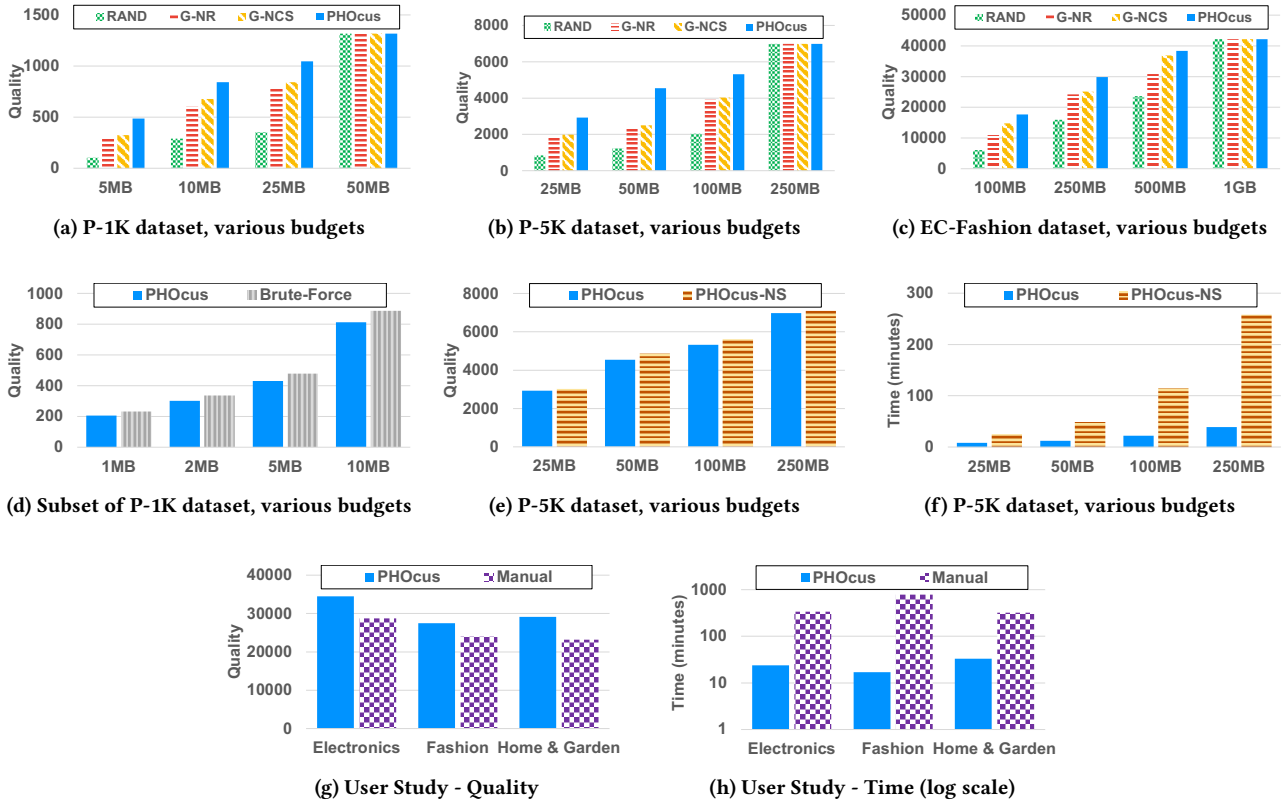
- **Public Photos Datasets (P)** - The first source is a publicly available dataset of photos [28]. It consists of millions of labeled images; the label contains information about an object that appears on the image and the confidence level. To test the applicability of our approach in various domains and dataset sizes, we extracted 5 datasets of various sizes from this source in the following way. First, we randomly selected 5 sets of photos of sizes 1K, 5K, 10K, 50K and 100K. For each set, we then extracted the labels that appeared on the photos, and used these labels to define the predefined subsets. The confidence of a label represents the relevance score of the photo to the predefined subset. The importance of the predefined subset is derived from its frequency in the full dataset. Finally, to compute the similarities between images we used cosine similarity between image embeddings. For the embeddings we used a commonly used pretrained ResNet-50 network [19].
  It is important to note that, while one might think that using uniform sampling to construct the predefined subsets and the resulting sets of labels would lead to little variance between them, the large number of photos (in the millions) together with the large number of labels (over 6000) in practice yields sufficient variance to measure the quality of the solution as well as for testing scalability.
- **E-Commerce Dataset (EC)** - The second source is private, and comes from a large e-commerce company (XYZ). It consists of photos of various product types from different categories, e.g., Smartphones, Running Shoes, and Office Chairs. These products come from 3 major domains: Electronics, Fashion, and Home & Garden. We therefore extracted 3 separate datasets from this data source, one for each domain. These domains were proposed by our collaborators due to their business value.
  The datasets were constructed as follows: First, in-house XYZ business analysts extracted all search queries in each of the three aforementioned domains from the query log of the fourth quarter of 2021. Using these queries, the analysts then derived 250 predefined subsets that correspond to the top-250 most frequent queries. The importance of the predefined subsets was derived from the query frequencies. The photos were then taken from the products that appeared in the result sets of the queries, and the relevance score derived from the retrieval score assigned by the search engine. The similarity between photos was calculated using cosine similarity between image embeddings (computed by XYZ analysts using an internal ML model). The number of photos in the datasets is 18745 (Electronics), 22783 (Fashion) and 19235 (Home & Garden), that are revtrieved using the 750 queries (250 per each domain) that were mentioned above.

Details about the eight datasets from the two sources presented above are summarized in Table 2.

Baselines. We compare our algorithm against four other algorithms, including two variants of a simple random algorithm, two greedy algorithms, and a variant of our algorithm without the sparsification optimization step in Section 4.3.

- **RAND-A** - Starts with an empty set, and during each iteration, randomly selects a photo, adds it to the set and stops when the budget limit is met.
- **RAND-D** - Starts with a set of all photos, and during each iteration, randomly selects a photo, deletes it from the set and stops when the budget limit is met.
- **Greedy-NR** - An Iterative Greedy algorithm that in each iteration finds the photo that maximizes the gain *without* taking into account the effect of covering the other photos, i.e. using the score function in Section 3.1 with $\mathcal{SIM}(q, p, p')$ set to 1.
- **Greedy-NCS** - Another Iterative Greedy algorithm that in each iteration finds the photo that maximizes the gain, taking into account the effect of covering the other photos using a *non-contextual similarity* function for all predefined subsets. This is, using the score function in Section 3.1 with $\mathcal{SIM}(q, p, p')$ set to the similarity of (p, p') for all q.
- **PHOcus-NS** - Our proposed algorithm (Algorithm 1, see Section 4), *without* the sparsification optimization.
- **PHOcus** - Our proposed algorithm *with* the proposed sparsification optimization.

Note that for the case where all costs are uniform, the well-known greedy algorithm of [37] is known to provide an optimal



(a) P-1K dataset, various budgets

(b) P-5K dataset, various budgets

(c) EC-Fashion dataset, various budgets

(d) Subset of P-1K dataset, various budgets

(e) P-5K dataset, various budgets

(f) P-5K dataset, various budgets

(g) User Study - Quality

(h) User Study - Time (log scale)

Figure 5: Evaluation results

$(1 - 1/e)$ worst-case approximation. Since Algorithm 1 takes the best out of two outputs produced by two sub-algorithms, one of which is the greedy algorithm, when costs are uniform Algorithm 1 is provably optimal. For this reason, we do not adapt summarization algorithms devised for settings where there are no explicit costs to include in our experimental evaluation. As explained in Section 2, these works either use the greedy algorithm or an inferior algorithm that is instead focused on a more general theoretical setting. For example, [35] makes several different assumptions (e.g., the photo subsets are assumed to be disjoint) and studies a more general problem with arbitrarily many knapsack constraints with different types of budgets and costs.

5.3 Evaluation Results

The experiments measured the *quality* of the solution ($\mathcal{G}(S)$ in Section 3.1) as well as the *feasibility* of PHOcus compared to stronger yet less practical baselines such as PHOcus-NS and the Brute-Force algorithms.

Quality. Figures 5a, 5b, and 5c show the quality achieved by each baseline over the public datasets of size 1K and 5K (P-1K and P-5K, respectively) and over the private e-commerce dataset from the Fashion domain (EC-Fashion), for a range of different budget values. Note that PHOcus-NS has similar quality to PHOcus and is therefore not included. Furthermore, both *RAND-A* and *RAND-D* achieved almost identical quality scores, hence we omit *RAND-D* and show only results for *RAND-A* (named *RAND* in all graphs). Results using the other 5 datasets in Table 2 are similar to those shown, hence are omitted.

Figure 5a depicts the quality achieved over the P-1K dataset for 4 different budget values. The ranking of the algorithms for all budgets is the same: our PHOcus algorithm has the best quality, followed by the two greedy variants, *G-NR* and *G-NCS*, and finally by the random algorithm *RAND*. Note that the rightmost column represents a budget that is large enough to retain all photos, hence no photos are removed and all baselines achieve the maximum possible score. The poor quality of the random algorithm using lower budget settings is predictable, since the photos are selected without using a "smart" strategy. Ignoring the similarity between photos leads to worse performance, as can be seen by *Greedy-NR*, e.g., if one good photo of an Eiffel Tower is already selected, adding a second similar photo adds very little to the overall quality of the solution. Using a contextual similarity function improves performance slightly, as seen by the performance of *Greedy-NCS* relative to *Greedy-NR*.

Figures 5b and 5c show similar trends to those in Figure 5a, i.e., PHOcus outperforms its competitors and *RAND* show the worst performance. However, in several cases (e.g., budget of 250MB in Figure 5c and budgets of 50MB and 100MB in Figure 5b) the difference in performance between *Greedy-NCS* and *Greedy-NR* is almost negligible, while in other cases *Greedy-NCS* strictly outperforms *Greedy-NR*.

Recall that Algorithm 1 consists of taking the best out of two outputs produced by two sub-algorithms. One of these algorithms is optimal for the case of uniform costs, and takes costs into account only for the stopping condition. However in our experiments, the other subalgorithm (which explicitly accounts for the weights) was superior in roughly 90% of the cases. This validates

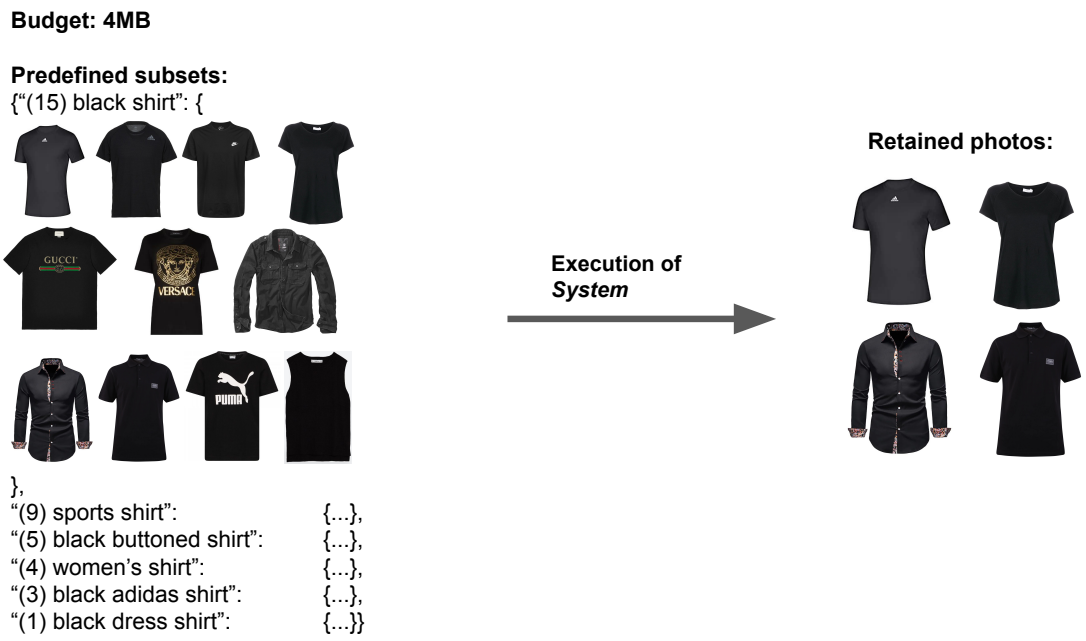


Figure 6: Example of execution of PHOcus during the user study on subset of products in the Fashion domain.

our claim that algorithms without explicit costs are not suited for our problem.

Comparison to the optimal algorithm. Figure 5d compares PHOcus and the brute-force algorithm over a small subset of the P-1K dataset that contains only 100 photos, as the brute-force algorithm could not run over larger inputs in a reasonable amount of time. Naturally, there is some loss in quality compared to the (non-practical) exhaustive search. However, the loss is always less than 15% (and in some cases even less then 10%). The brute-force results over small subsets of other EC datasets showed roughly the same trends, and are omitted.

Effects of sparcification. Finally, we studied the effects of the proposed sparcification approach, which reduces the amount of pairwise similarity computations needed. The relevant experiments involved testing a wide range of budget values (including a budget sufficient to cover all the predifined subsets). Concretely, we compare PHOcus and PHOcus-NS (a modification of PHOcus that does not apply the sparcification) algorithms both in terms of the solution quality and the required running time.

Figure 5e shows the solution quality of PHOcus and PHOcus-NS over the P-5K dataset with varying budgets. The results show that, in practice, sparcification has a very small effect in terms of quality (decrease of at most 5%), which is even far better the the non-tight worst-case theoretical bounds provided in Section 4.

Figure 5f shows the running time of the PHOcus and PHOcus-NS over the same dataset and the same budget values as in Figure 5e. The results show that, while the decrease in solution was almost negligible, sparcification significantly reduces the running time (from hours to tens of minutes).

Budget scenarios in practice. As can be seen in Figures 5a-5c, the difference in quality decreases as the budget gets closer to the cost of the full dataset. However, in the real-life e-commerce scenarios that we worked with, the ratio of the budget to the cost of the full dataset is much smaller, and corresponds to the range where our algorithm is shown to have the largest quality advantage in these figures. In particular, the budget for images on landing pages in the Electronics domain was 2MB, due to a hard limit of 100ms for loading all media on the web-page. These 2MB (roughly 25 images) had to be selected out of 640 photos (roughly 50MB), i.e the budget was about 4% of the total dataset. With these inputs, PHOcus achieved the best results (35% of the total quality), while the closest competitors (*Greedy-NCS* and *Greedy-NR*) reached only 18% and 16% of the total quality, respectively. This showcases the practical importance of an efficient solution for small budgets.

5.4 User Study

We were fortunate to have access to business analysts who work for XYZ and face a problem that our proposed solution addresses, hence we were able to perform a user study for a real application using real data.

The user study consists of two parts. In the first (where we consider only the PHOcus algorithm as it shows the best performance compared to other baselines), we compare our solution to the manual work of domain experts, both in terms of solution quality and time invested. In the second, we evaluate our quality function by using domain experts to generate the gold standard, which is then used to compare different baselines. Since the experts cannot manually examine large datasets (otherwise our solution would not be required), we asked them to repeat the experiment 50 times on small subsets of roughly 100 photos.

Before delving into technical details of the user study, it is important to mention that our industry collaborators helped us to design the study. Specifically, they chose domains that had the highest business value in a particular period of time and decided how many experts would work with us and the size of the provided datasets.

Setup. The user study involved 3 in-house business analysts from XYZ, each of whom is an expert in one of 3 domains - Fashion, Electronics and Home & Garden. The use case was the task

that the analysts perform in their daily job - generating landing pages – which requires a lot of manual effort and takes a lot of time. Based on the popular queries that are trending in a given period of time, the goal is to generate a landing page that contains a list of links to offers (products) that are relevant to the query. In order to make the page more attractive to potential buyers, the analyst carefully designs the page and adds photo(s) of relevant product(s). These photos are typically stored in a high-speed memory (cache) so they can be loaded quickly, hence improving the browsing experience. The analyst manually decides on a set of photos that will represent the landing pages given a list of search queries and a budget constraint (which is either the available memory or the number of allowed photos). In many cases the predefined subsets (products of the landing pages) intersect, hence some of the photos can be stored once and used multiple times for different landing pages. Finding such photos for as many landing pages as possible is desirable, and finding other similar photos that could be removed is a challenging task. In this study we compare a manual execution of this task as performed by an analyst to an execution of PHOcus with final touches and approval by the analyst (semi-automatic).

Example 5.2. Consider the small example shown in Figure 6. It contains 6 queries that yield 6 predefined subsets, each with 10 product photos (due to space constraints we show the actual photos only for the first predefined subset). Next to each query we show its importance[2]. Assume that all photos have roughly the same size of 1MB and that the budget is 4MB, hence only 4 images can be retained. Next to each query we indicate its importance (derived from the frequency). Even in this tiny example, the analyst needs to browse through up-to 60 different photos to find the best photos to retain, which is time consuming. However, using PHOcus, the result on the right is quickly returned. Note that the 4 retained photos (an Adidas shirt, a women's t-shirt, a button-up dress shirt, and a polo shirt) are all included in the first (and most important) predefined subset. The retained photos also provide good coverage of the rest of the predefined subsets: The Adidas shirt is also included in the sports shirt and Adidas shirt predefined subsets; the women's t-shirt covers the black women's shirt predefined subset; the button-up dress shirt cover both the buttoned shirt and dress shirt predefined subsets. Finally, polo shirt was the last that was chosen by PHOcus to retain since it has the maximum marginal gain. This set of 4 photos is the optimal solution that ensures that each of the landing pages corresponding to the predefined subsets has at least one relevant photo.

Results. In the first part of the study we measured the performance of our system compared to the manual solution produced by the analysts both in terms of the quality of the solution and the time that it took.

Figure 5g depicts the quality achieved by both methods over the three domains. We observe that PHOcus achieves a quality that is 15-25% higher than the quality of the manual solution, and that as a result the analysts gained unexpected insights in terms of which photos to retain.

Figure 5h depicts the time it took for both methods over the three domains, using a log scale. The time that it took for the manual effort was anywhere from 6 hours (in the best cases) to 14 hours (in the worst case), compared to 10 minutes of running time by PHOcus. These results show the applicability and benefit of our proposed solution in various real-life scenarios.

In the second part of the study, we asked the experts to perform 50 iterations, and in each iteration to compare the solutions generated by the two best performing methods - *Greedy-NCS* and PHOcus, on a set of roughly 100 photos (larger sets are harder for manual inspection). The experts then chose the best solution or clicked the "cannot decide" button if the solutions were similar. We then calculated how many times each of the solutions was selected as the best. For the Fashion domain, the solution generated by PHOcus was selected 35 times, *Greedy-NCS* was selected 3 times, and "cannot decide" was selected 12 times (meaning that the results were similar). For the Electronics and Home & Garden domains the results were simiar (37, 4, 9 and 34, 5, 11 respectively). These results prove that the utility function that we use for our evaluation is good.

6 CONCLUSIONS

To address the "data deluge" [34] that is facing many organizations, in particular those in e-commerce, we focus on the important special case of image data. We formalize the problem of deciding what images to archive to meet an online storage budget as the *Photo Archive Reduction* (PAR) problem, and show that it is NP-hard to approximate above a $(1-1/e)$ factor. Nevertheless, we show that there is a highly scalable algorithm with a $(1-1/e)/2$ approximation guarantee along with data-dependent guarantees that in practice far exceed this a priori bound. To further optimize the running time of our algorithm we also develop sparsification techniques which significantly improve the running time while incurring a negligible error in solution quality. Extensive experiments based on eight different (public and private) datasets show that our algorithm is not only efficient but yields high quality results.

We also show the practical effectiveness of our approach, as implemented in PHOcus. In particular, we show how the inputs to PAR can be automatically obtained for a variety of different applications, thus removing a barrier to its use. We also performed a user study of PHOcus within the XYZ e-commerce company, using expert XYZ analysts and mimicking what they were doing on a regular basis. The results show that PHOcus dramatically reduced the time it takes to arrive at a solution, and resulted in better quality solutions. The experts also gained unexpected insights in terms of which photos to retain. Finally, we give concrete examples of datasets and budgets in the e-commerce domain to emphasize the practical importance of our approach, and discuss how our quality metrics were validated using a gold standard based on the domain experts.

In future work, we plan to consider which photos to compress (i.e., to sacrifice quality to gain space) rather than to remove. While we believe that our model can already capture this problem, it would be interesting to see how it performs practically.

Finally, although in this paper we present data disposal in the context of photos, the general problem shares many of the same characteristics. Hence, we also plan to expand the model to include other forms of structured and unstructured data, as well as processes, and consider techniques that not only archive data but provide effective abstractions of the data given their intended use.

[2]While the provided list of queries is a subset of real-world set of queries, we do not provide their real frequency values due to its sensitivity. Hence, the provided importance values are for the demonstration proposes only.

REFERENCES

[1] Surabhi Adhikari et al. Nlp based machine learning approaches for text summarization. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 535–538. IEEE, 2020.

[2] Mohiuddin Ahmed. Intelligent big data summarization for rare anomaly detection. *IEEE Access*, 7:68669–68677, 2019.

[3] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*, 2017.

[4] Daisuke Aritomo, Chiemi Watanabe, Masaki Matsubara, and Atsuyuki Morishima. A privacy-preserving similarity search scheme over encrypted word embeddings. In *Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services*, pages 403–412, 2019.

[5] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680, 2014.

[6] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[7] Gérard Cornuéjols, George Nemhauser, and Laurence Wolsey. The uncapicitated facility location problem. Technical report, Cornell University Operations Research and Industrial Engineering, 1983.

[8] Arnon Dagan, Ido Guy, and Slava Novgorodov. An image is worth a thousand terms? analysis of visual e-commerce search. In *SIGIR*, 2021.

[9] Dataage 2025 - the digitization of the world, seagate us. http://www.seagate.com/our-story/data-age-2025.

[10] Susan B Davidson, Shay Gershtein, Tova Milo, and Slava Novgorodov. Disposal by design. *IEEE Data Eng. Bull.*, 45(1), 2022.

[11] Susan B. Davidson, Shay Gershtein, Tova Milo, Slava Novgorodov, and May Shoshan. Phocus: Efficiently archiving photos. *Proc. VLDB Endow.*, 15(12):3630–3633, 2022.

[12] Qianqian Fan, David J Lilja, and Sachin S Sapatnekar. Adaptive-length coding of image data for low-cost approximate storage. *IEEE Transactions on Computers*, 69(2):239–252, 2019.

[13] Luyue Fang, Xiaoqiang Di, Xu Liu, Yiping Qin, Weiwu Ren, and Qiang Ding. Quicklogs: A quick log parsing algorithm based on template similarity. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1085–1092, 2021.

[14] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[15] General Data Protection Regulation (GDPR). https://en.wikipedia.org/wiki/General_Data _Protection_Regulation.

[16] Shay Gershtein, Tova Milo, and Slava Novgorodov. Inventory reduction via maximal coverage in e-commerce. In *EDBT*, pages 522–533, 2020.

[17] Himanshu Gupta and Inderpal Singh Mumick. Selection of views to materialize in a data warehouse. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):24–43, 2005.

[18] Venky Harinarayan, Anand Rajaraman, and Jeffrey D Ullman. Implementing data cubes efficiently. *Acm Sigmod Record*, 25(2):205–216, 1996.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] How Many Images Are On The Internet 2021. https://www.16best.net/how-many-images-are-on-the-internet/.

[21] Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh Vempala. Locality-preserving hashing in multidimensional spaces. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 618–625, 1997.

[22] Howard Karloff and Milena Mihail. On the complexity of the view-selection problem. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 167–173, 1999.

[23] Martin L Kersten. Big data space fungus. In *CIDR*, 2015.

[24] Martin L Kersten and Lefteris Sidirourgos. A database system with amnesia. In *CIDR*, 2017.

[25] Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information processing letters*, 70(1):39–45, 1999.

[26] Kun Ho Kim, Oisin Mac Aodha, and Pietro Perona. Context embedding networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8679–8687, 2018.

[27] Yannis Kotidis and Nick Roussopoulos. A case for dynamic view management. *ACM Transactions on Database Systems (TODS)*, 26(4):388–423, 2001.

[28] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.

[29] Young-Man Kwon, Jae-Ju An, Myung-Jae Lim, Seongsoo Cho, and Won-Mo Gal. Malware classification using simhash encoding and pca (mcsp). *Symmetry*, 12(5):830, 2020.

[30] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429, 2007.

[31] Haoran Li, Peng Yuan, Song Xu, Youzheng Wu, Xiaodong He, and Bowen Zhou. Aspect-aware multimodal summarization for chinese e-commerce products. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8188–8195, 2020.

[32] Erik Lindgren, Shanshan Wu, and Alexandros G Dimakis. Leveraging sparsity for efficient submodular data summarization. *Advances in Neural Information Processing Systems*, 29, 2016.

[33] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[34] Tova Milo. Getting rid of data. *ACM J. Data Inf. Qual.*, 12(1):1:1–1:7, 2020.

[35] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning*, pages 1358–1367. PMLR, 2016.

[36] Hoshi Mistry, Prasan Roy, S Sudarshan, and Krithi Ramamritham. Materialized view selection and maintenance using multi-query optimization. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 307–318, 2001.

[37] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[38] Hieu V Nguyen and Li Bai. Cosine similarity metric learning for face verification. In *Asian conference on computer vision*, pages 709–720. Springer, 2010.

[39] Slava Novgorodov, Ido Guy, Guy Elad, and Kira Radinsky. Generating product descriptions from user reviews. In *The World Wide Web Conference*, pages 1354–1364, 2019.

[40] Qun Ren and Margaret H Dunham. Using semantic caching to manage location dependent data in mobile computing. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 210–221, 2000.

[41] Qun Ren, Margaret H Dunham, and Vijay Kumar. Semantic caching and query processing. *IEEE transactions on knowledge and data engineering*, 15(1):192–210, 2003.

[42] Ian Simon, Noah Snavely, and Steven M Seitz. Scene summarization for online image collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[43] Anurag Singh, Lakshay Virmani, and AV Subramanyam. Image corpus representative summarization. In *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, pages 21–29. IEEE, 2019.

[44] Pinaki Sinha, Sharad Mehrotra, and Ramesh Jain. Summarization of personal photologs using multidimensional content and context. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, pages 1–8, 2011.

[45] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[46] Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in neural information processing systems*, pages 1413–1421, 2014.

[47] Canwen Xu, Zhenzhong Chen, and Chenliang Li. Obj-glove: Scene-based contextual object embedding. *arXiv preprint arXiv:1907.01478*, 2019.

[48] Shao YiChuan and Xingjia Yao. Research of real-time data warehouse storage strategy based on multi-level caches. *Physics Procedia*, 25:2315–2321, 2012.

[49] Hengyu Zhao, Linuo Xue, Ping Chi, and Jishen Zhao. Approximate image storage with multi-level cell stt-mram main memory. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 268–275. IEEE, 2017.