Algorithms for Average Regret Minimization

Sabine Storandt,[1] **Stefan Funke**[2]
[1]University of Konstanz, Germany, [2]University of Stuttgart, Germany
sabine.storandt@uni-konstanz.de, funke@fmi.uni-stuttgart.de

Abstract

In this paper, we study a problem from the realm of multi-criteria decision making in which the goal is to select from a given set S of d-dimensional objects a minimum sized subset S' with bounded regret. Thereby, regret measures the unhappiness of users which would like to select their favorite object from set S but now can only select their favorite object from the subset S'. Previous work focused on bounding the maximum regret which is determined by the most unhappy user. We propose to consider the average regret instead which is determined by the sum of (un)happiness of all possible users. We show that this regret measure comes with desirable properties as supermodularity which allows to construct approximation algorithms. Furthermore, we introduce the regret minimizing permutation problem and discuss extensions of our algorithms to the recently proposed k-regret measure. Our theoretical results are accompanied with experiments on a variety of inputs with d up to 7.

Introduction

There are numerous (web) services in which it is impossible to present all available options to the user at once; for example, the list of matching pages in a web search, or the list of products of a certain type in an online warehouse. Hence there needs to be a preselection of a concise subset of the options to be shown to the user first (e.g., on page 1). A typical approach to make this selection is to rank all options using a multivariate function (e.g., for products weighting price, quality, date of appearance, and sales volume) and then presenting the top q options according to this function to the user. The hope is that the function captures the preference of a typical user or the majority of users. But in case user preferences are diverse, considering only a single function might be insufficient. As a remedy, an alternative selection approach based on *regret* takes the preferences of all (possible) users into account. The regret induced by a user preference expressed as a multivariate function f (to be maximized) with respect to the full set of options S and a subset S' thereof is

$$1 - \frac{\max_{s \in S'} f(s)}{\max_{s \in S} f(s)}.$$

So in case the best option in S according to the user preference is also contained in S' the regret is 0. In general, the regret value can assume any value in $[0, 1]$.

In previous work, the goal usually was to identify a concise subset S' of S which exhibits a small *maximum regret*, that is, a subset that makes the unhappiest user sufficiently happy. In this paper, we propose the usage of the *average regret* as a viable alternative. We focus on the scenario where user preference functions are (all possible) convex linear combinations of up to d criteria, and provide novel theoretical and practical results for computing subsets with small average regret in this setting.

Related Work

The notion of maximum regret as a measure for subset quality was introduced in (Nanongkai et al. 2010) in the context of databases. The idea was generalized to the k–maximum regret measure in (Chester et al. 2014), which does not measure the quality of a subset in relation to the users' top-1 choice in the whole set S but to the top-k choice.

It was proven in (Chester et al. 2014) that the problem of computing a subset of given size that minimizes the maximum regret is NP-hard for sufficiently large d. For $d = 2$, an exact algorithm based on dynamic programming was proposed in (Chester et al. 2014). For larger d, a simple greedy algorithm was discussed in (Nanongkai et al. 2010) and a randomized greedy linear programming algorithm for k-regret in (Chester et al. 2014). Note that both of these greedy algorithms do not come with an approximation guarantee. Known algorithms with an approximation guarantee are either based on the notion of *coresets* (Agarwal et al. 2017; Cao et al. 2017; Kumar and Sintos 2018) or by reformulating the problem as a set cover or hitting set problem (Agarwal et al. 2017; Asudeh et al. 2017; Kumar and Sintos 2018). In (Soma and Yoshida 2017), the relation between regret minimization and multi-objective submodular function maximization was investigated and two algorithms with provable guarantees were proposed.

Although the findings above are all concerned with maximum regret, we are not the first to consider a measure related to average regret. In the 2-page paper by (Zeighami and Wong 2016), it was suggested to minimize the *average regret ratio*, that is, the expected maximum regret ratio of a user. Their model is based on the explicit availability of a

finite set of possible user utility functions and a probability distribution over these functions. In contrast, we allow an infinite set of utility functions, namely all convex combinations of d criteria.

Contribution

We motivate and formally define average regret as a measure for the quality of a subset of d-dimensional objects.

We show that the average regret function is supermodular while the maximum regret function is not. Based on supermodularity, we are able to design a greedy algorithm that computes a subset S' of given size that comes with a quality guarantee. And in case we consider the average happiness of users, which is defined as 1 minus the average regret, we have a submodular function for which we can get a constant-factor approximation algorithm. We show that this even applies in case we consider the more complicated k-regret measure. As a crucial ingredient for the super- and submodular greedy algorithm, we show that the computation of the average k-regret is possible in time polynomial in $|S|$ and k for fixed dimension d.

In addition, we introduce the average regret minimizing permutation problem, in which the goal is to sort the objects in S such that for every prefix $s_1, \ldots, s_i$ the regret is as small as possible. This allows to smoothly decide for a prefix set S' with a desired trade-off between size and quality. Moreover, that decision can even be made for each user individually, which allows for more dynamic and customizable services.

Besides theoretical results, we also discuss efficient heuristics for practical use and show their applicability with suitable experiments on a variety of benchmark instances.

Preliminaries

We now provide formal definitions for the average regret measure and the respective optimization problems that will be studied in the paper.

Basic Setting and Properties

We are given a finite set of d-dimensional objects or points $S \subset \mathbb{R}^d_+$ which have non-negative entries in each dimension. Throughout the paper, we assume the user utility functions to be convex linear functions, that is, for some point $s \in S$ and some preference $\alpha \in \mathbb{R}^d_+$ with $\sum_{i=1}^d \alpha_i = 1$, we have $f(s, \alpha) = \alpha^T s$. This is a standard model for user utility functions also used, e.g., in (Soma and Yoshida 2017). Provided with the whole set S, a user chooses the point in S which maximizes his utility function, hence it yields $f(S, \alpha) = \max_{s \in S} f(s, \alpha) = \max_{s \in S} \alpha^T s$. As we assume to not know which users will use a service priori, we always consider all possible preferences, that is, all possible α. In a geometric interpretation, the α values form a d-dimensional simplex. However note that due to the constraint $\sum_{i=1}^d \alpha_i = 1$ the value α_d is uniquely defined after fixing $\alpha_1, \ldots, \alpha_{d-1}$. Hence all d-dimensional preferences are sufficiently described by using a $(d-1)$-dimensional simplex to which we will refer to as Ω. We will use this observation later to design efficient algorithms.

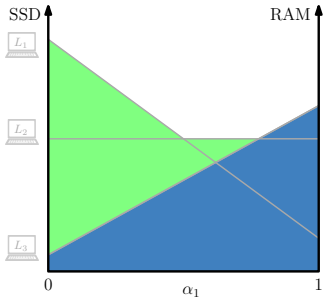

Figure 1: Example for $d = 2$: Set S consists of three laptops $\{L_1, L_2, L_3\}$ with different amounts of SSD memory and RAM (1TB/8GB), (512GB/24GB), (128GB/32GB). The bottom segment in the illustration is the 1-dimensional simplex containing all possible values for α_1 (and implicitly also $\alpha_2 = 1 - \alpha_1$). The happiness volume induced by S is the area marked green and blue. For the subset $S' = \{L_3\}$ the happiness volume is the blue area. The ratio of those two is 0.566, leading to an average regret of $1 - 0.566 = 0.434$.

We are now ready to formally define the average regret for a subset S' of S. Intuitively, we first compute the sum over all possible utility functions $f(S', \alpha)$ which results in a 'happiness volume' induced by subset S'. Then we compute the 'happiness volume' of the whole set S and divide the former by the latter to get an average happiness or a 'happiness ratio' in $[0, 1]$ which is then subtracted from 1.

Definition 1 (Average Regret) *Given a set S and a subset S' thereof, the average regret of S' with respect to S is defined as $r_{avg}(S', S) = 1 - \int_\Omega f(S', \alpha) d\alpha / \int_\Omega f(S, \alpha) d\alpha$.*

The concept of average regret is illustrated in Figure 1.

So why should one use r_{avg} instead of the commonly applied maximum regret measure $r_{max}(S', S) = 1 - \min_\alpha \{f(S', \alpha)/f(S, \alpha)\}$?

Assuming the best achievable maximum regret for a subset of given size is M, the regret of all other users is completely irrelevant when using the maximum regret measure as long as it is below M. So while there could be sets where the regret for most users is indeed 0, algorithms that minimize the maximum regret would not prefer such a solution over one where the regret is exactly M for all users. The average regret on the other hand would clearly favour a subset in which the regret is small for many users.

In addition, the average regret can be used to lower bound the maximum regret as shown in the following lemma.

Lemma 2 $r_{avg}(S', S) \leq r_{max}(S', S)$.

Proof. Let B be the maximum regret subtracted from 1, $B = 1 - r_{max}(S', S) = \min_\alpha \{f(S', \alpha)/f(S, \alpha)\}$. Then we know that for all α it yields $f(S', \alpha)/f(S, \alpha) \geq B$ and hence $f(S', \alpha) \geq B \cdot f(S, \alpha)$. If we plug this into our formula for the average regret subtracted from 1 we get

$$\frac{\int_\Omega f(S', \alpha) d\alpha}{\int_\Omega f(S, \alpha) d\alpha} \geq B \frac{\int_\Omega f(S, \alpha) d\alpha}{\int_\Omega f(S, \alpha) d\alpha} = B$$

and hence $r_{avg}(S', S) \leq 1 - B = r_{max}(S', S)$. ∎

Optimization Problems

Based on our notion of average regret, we study two related optimization problems in this paper. The first one is the classical problem of computing a subset of fixed size that minimizes the regret:

Definition 3 (Regret Minimization Problem) *Given a set S, find a subset S' with $|S'| = q$ such that*

$$r_{avg}(S', S) \leq r_{avg}(S'', S) \quad \forall S'' \subset S, |S''| = q,$$

that is, a subset S' with minimum average regret.

So if on page 1 of a product website there is space for $q = 10$ objects, a solution to the regret minimization problem constitutes a reasonable selection.

However, not in every scenario q can easily be determined a priori. For example, a certain website might be viewed on different devices with different screen sizes. On a smart phone, the number of objects that can be viewed at once might be smaller than the number of objects on a tablet, demanding different values of q. Moreover, if the user does not like any of the objects shown on page 1, the question is with which objects to proceed on page 2 and so on. This leads to an extended problem formulation:

Definition 4 (Regret Minimizing Permutation Problem) *Given a set S of size n, sort the elements in S such that $\sum_{i=1}^{n} r_{avg}(S_i, S)$ is minimized where S_i refers to the first i elements in the sorted list.*

We will show that both optimization problems can be tackled with similar techniques.

Modularity and Approximation

In this section, we will exploit the fact that the average regret function $r_{avg}(S', S)$ exhibits certain characteristics which allow to design efficient approximation algorithms. More precisely, we will investigate monotonicity, submodularity and supermodularity.

Minimizing Regret and Maximizing Happiness

Remember that the average regret function is defined as $r_{avg}(S', S) = 1 - \int_\Omega f(S', \alpha)d\alpha / \int_\Omega f(S, \alpha)d\alpha$. From now on we call the subtrahend the average happiness function $h_{avg}(S', S) = 1 - r_{avg}(S', S)$. Obviously, a subset S' that minimizes the average regret automatically maximizes the average happiness. However, that unfortunately doesn't imply that a good approximation algorithm for one of those functions automatically also constitutes a good approximation algorithm for the other function, as discussed below.

Submodularity of Average Happiness

For a monotone and submodular set function to be maximized subject to a cardinality constraint on the subset, there exists a simple greedy algorithm that admits a $1-1/e$ approximation guarantee (Fisher, Nemhauser, and Wolsey 1978). So if we can prove that the average happiness function is both, monotone and submodular, we can identify in polynomial time a subset S' of given size q which exhibits an average happiness which is at least $1 - 1/e$ times the average happiness of the optimal q-sized subset.

Definition 5 (monotone) *A set function $f : 2^S \to \mathbb{R}$ is monotone if $A \subset B \subset S$ implies $f(A) \leq f(B)$.*

Lemma 6 *$h_{avg}(S', S)$ is monotone.*

Proof. $h_{avg}(S', S) = \int_\Omega f(S', \alpha)d\alpha / \int_\Omega f(S, \alpha)d\alpha$. The denominator is fixed for a given set S. Let $S'' \supset S'$ be a superset of S'. As $f(S', \alpha) = \max_{s \in S'} \alpha^T s$ it clearly yields $f(S', \alpha) \leq f(S'', \alpha)$ as the additional points in $S'' \setminus S'$ can only increase the function value. ∎

Intuitively, a set function is submodular if the gain of adding an element to a set A is always at least as large as the gain of adding the same element to a superset of A.

Definition 7 (submodular) *A set function $f : 2^S \to \mathbb{R}$ is submodular if $\forall A \subset B \subset S$ and $s \in S \setminus B$ we have $f(B + s) - f(B) \leq f(A + s) - f(A)$.*

Lemma 8 *$h_{avg}(S', S)$ is submodular.*

Proof. We prove that for each α we have $f(A + s, \alpha) - f(A, \alpha) \geq f(B + s, \alpha) - f(B, \alpha)$ for all $A \subset B$ using a case distinction. In the first case, s does not uniquely determine the function value for $B + s$. Then we have $f(B + s, \alpha) = f(B, \alpha)$ and hence the right side of the submodular inequality becomes 0, making it always true. In the second case, s uniquely determines the function value for $B + s$ but not for $A + s$. This is impossible as then we would have $f(A + s, \alpha) < f(B + s, \alpha) = \alpha^T s$ while $\alpha^T s$ would also be a valid function value for $A + s$. In the third case, s uniquely determines the function value for $B + s$ and $A + s$. Then we have $f(A + s, \alpha) = f(B + s, \alpha)$ which allows to rearrange the submodular inequality to $-f(A, \alpha) \geq -f(B, \alpha) \Leftrightarrow f(A, \alpha) \leq f(B, \alpha)$ which is true as f is monotone. ∎

According to those two lemmas, we can use the greedy framework described in (Fisher, Nemhauser, and Wolsey 1978) to find a subset with a constant factor guarantee on the average happiness. The greedy algorithm starts with $S' = \emptyset$ and proceeds by iteratively adding the element to S' which increases the function value the most. After q rounds, the desired subset is found.

Corollary 9 *The greedy algorithm computes a fixed size subset of S which exhibits at least $(1 - 1/e)$ of the average happiness of the optimal such subset.*

But how does this approximation guarantee for average happiness translate to the average regret function?

Theorem 10 *A $1-1/e$ approximation algorithm for optimal average happiness h provides a $1 + \frac{h}{e(1-h)}$ approximation for the optimal average regret.*

Proof. Let the average regret be denoted by $r = 1 - h$, we want to upper bound the ratio $\frac{1-p \cdot h}{1-h}$ for $p = 1 - 1/e$ to get the approximation guarantee for r while having an approximation guarantee of p for h. We get:

$$\frac{1 - p \cdot h}{1 - h} = 1 + \frac{h(1 - p)}{1 - h} = 1 + \frac{h \cdot 1/e}{1 - h}$$

∎

So while we have a constant-factor approximation guarantee for h, the approximation quality for r also depends on h.

Corollary 11 *For $h \leq 1/2$, the approximation guarantee for the average regret is bounded by $1 + 1/e$.*

According to the corollary, for h small enough, we get a sufficiently strong approximation guarantee for r. But with the average happiness converging to 1 the approximation factor for r becomes arbitrarily large.

Supermodularity of Average Regret

A function f is supermodular if and only if $-f$ is submodular. Equivalently, the following definition applies.

Definition 12 (supermodular) *A set function $f : 2^S \to \mathbb{R}$ is supermodular if $\forall A \subset B \subset S$ and $s \in S \setminus B$ we have $f(B + s) - f(B) \geq f(A + s) - f(A)$.*

Lemma 13 *$r_{avg}(S', S)$ is supermodular.*

Proof. We have shown in Lemma 8 that h_{avg} is submodular. Therefore, $-h_{avg}$ is supermodular. Adding any constant term does not invalidate the supermodularity. In conclusion, $1 - h_{avg} = r_{avg}$ is supermodular. ∎

To demonstrate that this characteristic crucially depends on our regret definition, we now provide a counter-example for supermodularity of the maximum regret r_{max} by demonstrating that $1 - r_{max}$ is not always submodular.

Example 14 *We consider the function*

$$g(S, S) = 1 - r_{max}(S', S) = \min_{\alpha}\{f(S', \alpha)/f(S, \alpha)\}.$$

To disprove submodularity, we have to find sets A and B with $A \subset B$, such that adding an element $s \in S \setminus B$ to B increases the function value more than it does for A. We use the following example for $d = 2$: $S = \{(6, 1), (2, 2), (1, 6)\}, B = \{(6, 1), (2, 2)\}$, $A = \{(2, 2)\}$ and $s = (1, 6)$. This leads to $g(B) = 1/3$ (for $\alpha = (0, 1)$), $g(A) = 1/3$ (for $\alpha = (1, 0)$ and $\alpha = (0, 1)$) and $g(B+s) = g(S) = 1$ ($B + s$ now equals S), $g(A + s) = 1/3$ (for $\alpha = (0, 1)$). Hence we get $g(B + s) - g(B) = 2/3 \geq g(A + s) - g(A) = 0$ which contradicts submodularity.

So for $1 - r_{max}$ we can not easily design a constant factor approximation greedy algorithm as we did for the average happiness function. And we can also not apply existing results for supermodular functions (Zeighami and Wong 2016) to r_{max} while we can indeed use them for r_{avg}. In particular, we get the following guarantee when using a *reverse greedy algorithm*, which starts with $S' = S$ and then in each of $n - q$ many rounds deletes the element whose removal increases the average regret the least:

Lemma 15 *The reverse greedy algorithm provides an approximation guarantee of $(e^t - 1)/t$ for the average regret minimization problem where $t = x/(1-x)$ with x being the steepness of $r_{avg}(S', S)$.*

Here steepness refers to the maximum possible decrease of $r_{avg}(S', S)$ when removing an element from S'. The decrease is most profound when the element is the only element in S'. Hence we get $x = \max_{s \in S} \int_{\Omega} f(\{s\}, \alpha) d\alpha$.

Regret Minimizing Permutations

One drawback of using the regret minimization problem for subset selection is that the solution is only valid for the particular choice of q (the subset size). In contrast, the classical ranking method (using only a single mutivariate function) produces a total order of the elements. Hence once the order is obtained, a solution for any q is available by simply outputting the top-q elements in that order.

We now aim for the same level of flexibility when using the regret measure. For that purpose, we want to solve the *average regret minimizing permutation problem*. Here, the goal is to compute an ordering of the elements in S, such that the average regret is as small as possible for every prefix S_i. More precisely, we want to minimize the accumulated average regret over all S_i for $i = 1, \ldots, n = |S|$. We refer to the individual regrets as r_i and to the accumulated regret as $R = \sum_{i=1}^{n} r_i$ with $R \in [0, n]$. Obviously, it always holds $r_n = 0$ as $S_n = S$, and $r_i \geq r_{i+1}$ for all $i = 1, \ldots, n - 1$ due to monotonicity. Similarly, we define $h_i = 1 - r_i$ as the average happiness of set S_i, and $H = \sum_{i=1}^{n} h_i$ as the accumulated happiness with $h_n = 1$ and $h_{i+1} \geq h_i$ for all $i = 1, \ldots, n - 1$. Maximizing the accumulated average happiness yields a solution that minimizes the accumulated average regret and vice versa.

We observe that applying the same greedy algorithm we used for the regret minimization problem, also yields provably good solutions for the permutation problem:

Theorem 16 *The greedy algorithm yields a $1 - 1/e$ approximation guarantee for the accumulated average happiness.*

Proof. We set $q = n$ and use the order in which the greedy algorithm selects the elements as the desired permutation of S. Now we know that for any fixed q, the elements selected so far exhibit an average happiness h_q of at least $(1 - 1/e)h_q^*$ where h_q^* is the optimal average happiness achievable for a subset of size q, see Corollary 9. Hence we can lower bound H as follows: $H = \sum_{i=1}^{n} h_i \geq \sum_{i=1}^{n}(1 - 1/e)h_q^* = (1 - 1/e)\sum_{i=1}^{n} h_q^* = (1 - 1/e)H^*$ where H^* is the optimal accumulated average happiness. ∎

In the same way, the approximation guarantee for the average regret when using the reverse greedy algorithm (Lemma 15) transfers to the accumulated average regret R.

Average Regret Computation

Although we have proven in the last section that the greedy algorithm is a useful tool for average happiness and average regret computation, there is still one crucial ingredient missing to make the algorithm work: We need to be able to determine the next-best element efficiently, that is, the element in S whose addition to S' increases the average happiness the most in the standard greedy algorithm, or the element in S' whose removal increases the average regret the least in the reverse greedy algorithm. This demands suitable algorithms to compute $\int_{\Omega} f(S', \alpha) d\alpha$. We will now show that for fixed dimension d, this is always possible in polynomial time.

Computing the Happiness Volume

In the following we explain how to compute the happiness volume in polynomial time, both theoretically as well as in

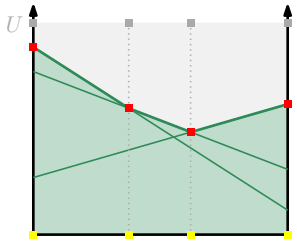

Figure 2: Volume computation for $d = 2$. We are interested in the green area defined by the upper envelope of the green lines. Via dualization we determine the vertices of the upper envelope (red boxes). We then compute the volume in two steps: We first determine the the volume of the convex hull of the yellow and gray (shadow) vertices, and then subtract from it the volume of the convex hull of the red and gray vertices (gray area) to finally obtain the green area.

practice. To compute the happiness volume for a set $S \subset \mathbb{R}^d_+$ of points, $|S| = n$, we first construct respective hyperplanes $\mathcal{H}$ in the space $\mathbb{R}_+ \times \Omega$. More precisely, for each point $p = (p_1, p_2, \ldots, p_d) \in S$, we create the hyperplane

$$h_p : y = (p_1 - p_d)x_1 + \cdots + (p_{d-1} - p_d)x_{d-1} + p_d.$$

Additionally, for some very large value M auxiliary hyperplanes $y = Mx_i, \forall i = 1, \ldots (d-1)$ and

$$y = \left(\sum_{i=1}^{d-1} Mx_i\right) - M$$

are created, which essentially restrict our focus of interest to the positive orthant with $\sum_{i=1}^{d-1} x_i \leq 1$. We are interested in the volume between the hyperplane $y = 0$ and the upper envelope of the hyperplanes.

Since implementations of direct upper envelope constructions in dimensions higher than 3 are quite rare, we make use of a well-known duality between the upper envelope of an arrangement of hyperplanes and the upper convex hull of respective dual points[1]. We dualize each hyperplane $h : y = \alpha_1 x_1 + \alpha_2 x_2 + \ldots \alpha_{d-1} x_{d-1} + \alpha_d$ to a point $\mathcal{D}(h) := (\alpha_1, \alpha_2, \ldots, \alpha_{d-1}, -\alpha_d) \in \mathbb{R}^d$. There is a one-to-one correspondance between the upper envelope of $\mathcal{H}$ and the boundary of the upper convex hull of its dual point set $\mathcal{D}(\mathcal{H}) = \{\mathcal{D}(h) : h \in \mathcal{H}\}$. So we compute the convex hull of $\mathcal{D}(\mathcal{H})$, which can be done in expected $O(n \log n + n^{\lfloor (d)/2 \rfloor})$, e.g., via (Clarkson and Shor 1989). Then every $(d-1)$-dimensional facet of the upper convex hull of $\mathcal{D}(\mathcal{H})$ corresponds to a vertex of the upper envelope (there are at most $O(n^{\lfloor d/2 \rfloor})$ of them according to the upper bound theorem (McMullen 1970)). Then for each constructed vertex $(\alpha_1, \alpha_2, \ldots, \alpha_{d-1}, y)$ of the upper envelope, we construct its 'shadow' vertex with coordinates $(\alpha_1, \alpha_2, \ldots, \alpha_{d-1}, U)$ for some large but fixed U. The resulting vertex set of size $O(n^{\lfloor d/2 \rfloor})$ is in convex position, computing a decomposition of its convex hull into simplices can be done in $O(n^{d^2/4})$. By summing up the volumes of the simplices we compute the volume 'above' the upper envelope. By subtracting it from the volume between the shadow vertices and their counterpart in the hyperplane $y = 0$ we obtain the volume below the upper envelope. See Figure 2 for an illustration.

The total running time is $O(n^{d^2/4})$. In practice, we expect the complexities of the occurring convex hulls to be considerably smaller than in the worst-case, though.

[1]The d transformation does not work for vertical planes, hence the formulation of the auxiliary hyperplanes via some large M.

Sampling-based Heuristic

Nevertheless, for higher dimensions or very large point sets the running times for computing the exact volume as above become prohibitive. In that case, a sampling-based approach can be employed which essentially discretizes the parameter space Ω. So for a given $\epsilon > 0$, we consider the set of points $\mathcal{P} := (k_1\epsilon, k_2\epsilon, \ldots, k_d\epsilon) : k_i \in \mathbb{N}, \sum k_i = 1/\epsilon$. Clearly, $|\mathcal{P}| = O(\epsilon^{-(d-1)})$. To approximate the volume below the upper envelope, we simply multiply for each $\alpha \in \mathcal{P}$ the volume of a $(d-1)$-dimensional cube of side length ϵ with $\max_{s \in S} \alpha^T s$. The running time for such an approximate volume approximation is then $O(n\epsilon^{-(d-1)})$. Clearly, the smaller ϵ, the better the approximation.

Extension to k-Regret

The k-regret measure was introduced in (Chester et al. 2014). The idea is to not measure the quality of a subset S' for a given preference α with respect to the best element in S but to the k^{th} best element in there. In case the function value for S' exceeds the one for S under this measure (which is impossible for $k = 1$ but could happen for $k \geq 2$), it is capped at the value for S. Then again, the k-regret value can only assume values within $[0, 1]$.

We now discuss how average k-regret works in distinction to the maximum k-regret measure discussed in (Chester et al. 2014). Again, we need the reference happiness volume for S and then the happiness volume for S' to measure the quality of S'. For $k = 1$, the computations of those two were independent. But for $k \geq 2$, we have to take care of capping the volume of S' appropriately in case the happiness of a user is larger than the happiness induced by the k^{th} best element from S. We now refer to the happiness volume of the k^{th} best elements in S as V_k and to the standard happiness volume of S' as V'. Then the average k-regret is defined as

$$r^{(k)}_{avg} = 1 - (V' \cap V_k)/V_k.$$

Figure 3 illustrates V_k and $V' \cap V_k$ for an example instance.

Exact Volume Computation Even in case of the generalization to k-regret, the respective volume computation can actually be done in polynomial time: As in the $k = 1$ case, we first derive the set $\mathcal{H}$ of hyperplanes, but then compute the full arrangements of hyperplanes, which can be done in time $O(n^d)$. We then traverse the arrangement, marking all cells below the $k-$level and below the hyperplanes in S'. This can be done in time linear in the complexity of the arrangement, hence $O(n^d)$. The volume of each of the marked cells (which are convex and of polynomial size) is then determined again via a decomposition into simplices whose

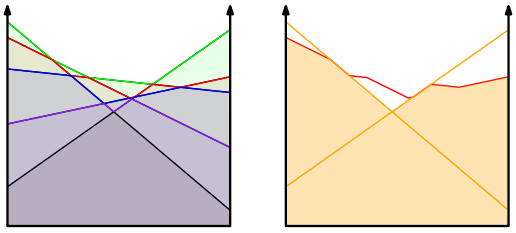

Figure 3: k-regret for $d = 2$. The left image shows the k^{th} best element in each direction among a set of 5 elements and the induced happiness volume for $k = 1$ (green), $k = 2$ (red), $k = 3$ (blue), $k = 4$ (purple) and $k = 5$ (black). The right image illustrates the hapiness volume (orange area) for a subset S' consisting of the two orange lines for $k = 2$. The area can never include points above the red line.

volumes are easily computed. Clearly, the overall running time is again polynomial, yet we do not consider this approach useful in practice.

Heuristic The above described sampling heuristic translates also to the general k-regret case by multiplying the volume of the $(d-1)$-dimensional ϵ-cube with the k^{th} highest function value in S, and the best function value in the set S' (possibly capped at the k^{th} highest function value in S).

Experiments

We implemented the proposed algorithms for average regret in C++, using CGAL 4.11 for computing convex hulls and Eigen 3.3.4 for determining volumes. Experiments were conducted on an AMD Ryzen 2400G with 3.6GHz and 64GB RAM. For benchmarking, we use a variety of different inputs with the following characteristics:

RC Random points in the unit hypercube (synthetic). We generated up to $n = 10^6$ points for d from 3 to 6.

ElNino Oceanographic data (real-world). It has $n = 178080$ points for $d = 5$ (zonal and meridional wind speed, water and surface temperature, relative humidity).

AirData Flight statistics (real-world). It has $n = 458311$ points with $d = 7$ (distance, air-time, arr-dely, dep-delay, taxi-out, taxi-in, actual-elapsed-time).

Weather Mean temperatures for every January and July ($d = 2$) for $n = 566262$ locations around the world.

The real-world data sets were all used before in related publications, e.g., (Soma and Yoshida 2017; Kumar and Sintos 2018). All data sets were normalized to only have values in the interval $[1, 100]$ by first subtracting the minimum value in each dimension from the other values in that dimension, and subsequent scaling of the values in each dimension.

Greedy Algorithm: Exact vs Heuristic Selection

We first evaluate the standard greedy algorithm (always adding the next-best element) which has a constant-factor approximation guarantee for the average happiness. We report running times and regret values for the exact variant as well as for the heuristic version, where we estimate the volume of $S' + s$ for all $s \in S \setminus S'$ in each round based on sampling. To judge the quality of the heuristic greedy truthfully, we compute the exact happiness volume induced by the final subset S'.

			exact		heuristic	
d	n	q	r_{avg}	time (s)	r_{avg}	time(s)
3	10^6	2	0.0014	1462	0.0014	0.54
		16	$2.2 \cdot 10^{-6}$	1690	0.0006	0.54
3	10^5	2	0.0019	144	0.0019	0.06
		16	$5.5 \cdot 10^{-6}$	168	0.0017	0.06
4	10^5	2	0.0052	603	0.0052	0.20
		16	$2.6 \cdot 10^{-5}$	1204	0.0007	0.20
4	10^4	2	0.0060	61	0.0060	0.02
		16	0.0002	182	0.0059	0.02
5	10^4	2	0.0158	274	0.0158	0.05
		16	0.0005	3452	0.0116	0.05
5	10^3	2	0.0101	31	0.0101	0.01
		16	0.0002	525	0.0088	0.01
6	10^3	2	0.0104	113	0.0164	0.01
		16	0.0006	15925	0.0141	0.00
6	10^2	2	0.0504	21	0.0504	0.00
		16	0.0008	4190	0.0299	0.00

Table 1: Results for RC with varying values of n, d and q: exact greedy, and heuristic greedy with $\epsilon = 0.1$.

	exact		heuristic	
q	r_{avg}	time (s)	r_{avg}	time(s)
1	0.016	63	0.016	0.04
2	0.005	133	0.005	0.08
4	0.001	145	0.001	0.09
8	0.000	175	0.001	0.09
16	0.000	196	0.001	0.10

Table 2: Results for the Weather benchmark ($d = 2$): exact greedy, and heuristic greedy with $\epsilon = 0.1$.

Synthetic Data For the RC benchmark, the results are summarized in Table 1. For small q, the average regret values for the exact and heuristic version are the same in almost all settings. For $q = 16$, we see that the exact algorithm clearly outperforms the heuristic variant in terms of regret, but at the same time the running times are significantly higher.

Real-world Data For the real-world data sets, the greedy results are collected in Table 2 (Weather), Figure 4 (ElNino) and Table 3 (AirData).

For the Weather benchmark, we observe that the heuristic version results in very similar regret values as the exact algorithm, while being faster by up to three orders of magnitude. For $\epsilon \leq 0.02$ we even get the very same results from the heuristic and the exact version for all tested values of q. So in conclusion, the heuristic approach works even better on this real-world data than on our synthetic data.

For the ElNino benchmark, we performed a sensitivity analysis for the heuristic greedy algorithm with respect to

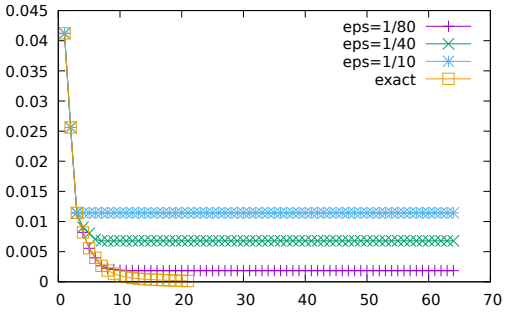

Figure 4: Average regret in dependency of q for the ElNino data set using heuristic greedy with different choices of ϵ.

	heuristic (ϵ_1)		heuristic (ϵ_2)	
q	r_{avg}	time (s)	r_{avg}	time(s)
1	0.2936	361	0.2936	15027
2	0.1920	361	0.1919	17018
4	0.1043	361	0.1020	18000
8	0.1043	361	0.0716	18062
16	0.1043	361	0.0715	18065

Table 3: Results for the AirData benchmark ($d = 7$): heuristic greedy with $\epsilon_1 = 1/20$ (left) and $\epsilon_2 = 1/40$ (right).

the choice of ϵ, see Figure 4. Interestingly, the first elements chosen are always the same regardless of the ϵ value. But from some value of q onwards the regret does not decrease anymore. This is due to the sampling based approximate volume computation is not precise enough to reliably detect volume increase by different elements anymore. Only a high sampling resolution of $\epsilon = 1/80$ leads to regret values close to 0. But of course, there is a price to pay: While the algorithm only takes about 2 seconds for $\epsilon = 1/10$, it takes almost 3000 seconds for $\epsilon = 1/80$.

For the AirData benchmark the exact greedy algorithm was too slow to produce solutions, as the exact volume computation becomes expensive with growing dimension d. But the heuristic version was still applicable. In Table 3, the results for two different choices of ϵ can be found. We observe a similar trade-off between quality and running time as for the ElNino data. The time to select the next point decreases rapidly in both cases.

Standard vs Reverse Greedy

In our theoretical analysis, we exploited two different greedy algorithms: The standard one (always add the next-best element) and the reverse greedy (always remove the element which contributes the least). Both are able to solve the average regret minimization problem for given q as well as the average regret minimizing permutation problem.

In Figure 5, we compare the two algorithms on an RC benchmark with $d = 4$ and $n = 1000$. Evidently, the forward greedy algorithm outperforms the reverse greedy algorithm in all aspects; not only is the accumulated regret R smaller but also for individual choices of q up to 30, the r_q value is always better. For larger choices of q the average regret is almost zero for both algorithms. Concerning the running time, the reverse greedy algorithm is slightly slower than the forward algorithm when computing the whole permutation (10.7 minutes versus 13.5 minutes). For (small) choices of q, however, the standard greedy algorithm is way faster as it only requires q rounds while the reverse greedy algorithm requires $n - q$ rounds.

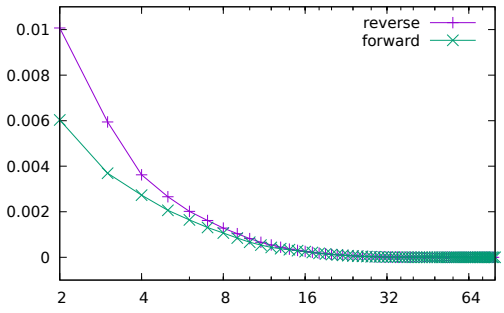

Figure 5: Average regret in dependency of the subset size q for the standard (forward) greedy algorithm as well as for the reverse greedy algorithm (x-axis in logscale). For the sake of visualization, the plot only starts at $q = 2$. For $q = 1$, the average regret is 0.0107 for the standard greedy and 0.2456 for the reverse greedy algorithm. The accumulated regret over all q is $R = 0.0342$ and $R = 0.2790$, respectively.

	k=1	k=3	k=5	k=7	k=9
q	r_{avg}	r_{avg}	r_{avg}	r_{avg}	r_{avg}
1	0.3280	0.3161	0.3121	0.3095	0.3016
2	0.2028	0.1949	0.1907	0.1880	0.1790
4	0.0943	0.0901	0.0880	0.0864	0.0772
8	0.0064	0.0018	0.0012	0.0002	0.0001
16	0.0004	0.0002	0.0000	0.0000	0.000

Table 4: Results for the AirData benchmark ($d = 7$): heuristic greedy with $\epsilon = 0.1$ for different choices of k, average regret values are based on the approximated volumes.

Heuristic Computation of Average k-Regret

Finally, we investigate what happens if we use average k-regret as a quality measure.

Table 4 shows our results on the AirData benchmark for $k = 1, 3, 5, 7, 9$. For $q = 16$ running times were between 2 and 3 minutes for all settings. We observe that the average regret decreases with growing k as to be expected. But note that the regret values here are only estimations based on the approximated volume (computed by sampling), as the exact volume computation was too expensive even a posteriori. Nevertheless it appears that increasing k has the desired effect on this real-world data set.

Conclusions and Future Work

We motivated the average regret (or the average happiness volume) as an alternative measure for subset quality and showed that a standard greedy algorithm achieves good results in theory (based on submodularity) and practice. At

the heart of our implementation lies the computation of the multi-dimensional happiness volume. For larger d, this is at the moment only possible in a heuristic fashion. In future work, tools from computational geometry should be applied to make the exact greedy algorithm scalable.

References

Agarwal, P. K.; Kumar, N.; Sintos, S.; and Suri, S. 2017. Efficient algorithms for k-regret minimizing sets. *arXiv preprint arXiv:1702.01446*.

Asudeh, A.; Nazi, A.; Zhang, N.; and Das, G. 2017. Efficient computation of regret-ratio minimizing set: A compact maxima representative. In *Proceedings of the 2017 ACM International Conference on Management of Data*, 821–834. ACM.

Cao, W.; Li, J.; Wang, H.; Wang, K.; Wang, R.; Chi-Wing Wong, R.; and Zhan, W. 2017. k-regret minimizing set: Efficient algorithms and hardness. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 68. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Chester, S.; Thomo, A.; Venkatesh, S.; and Whitesides, S. 2014. Computing k-regret minimizing sets. *Proceedings of the VLDB Endowment* 7(5):389–400.

Clarkson, K. L., and Shor, P. W. 1989. Application of random sampling in computational geometry, II. *Discrete & Computational Geometry* 4:387–421.

Fisher, M. L.; Nemhauser, G. L.; and Wolsey, L. A. 1978. An analysis of approximations for maximizing submodular set functions—ii. In *Polyhedral combinatorics*. Springer. 73–87.

Kumar, N., and Sintos, S. 2018. Faster approximation algorithm for the k-regret minimizing set and related problems. In *2018 Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX)*, 62–74. SIAM.

McMullen, P. 1970. The maximum numbers of faces of a convex polytope. *Mathematika* 17(2):179–184.

Nanongkai, D.; Sarma, A. D.; Lall, A.; Lipton, R. J.; and Xu, J. 2010. Regret-minimizing representative databases. *Proceedings of the VLDB Endowment* 3(1-2):1114–1124.

Soma, T., and Yoshida, Y. 2017. Regret ratio minimization in multi-objective submodular function maximization. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 905–911.

Zeighami, S., and Wong, R. C.-W. 2016. Minimizing average regret ratio in database. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, 2265–2266. New York, NY, USA: ACM.