Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

# FPT approximation schemes for maximizing submodular functions ☆

Piotr Skowron

*TU Berlin, Berlin, Germany*

ARTICLE INFO

*Article history:*
Received 7 December 2016
Received in revised form 10 August 2017

*Keywords:*
Parameterized complexity
Fixed parameter tractability
Approximation algorithms
Computational social choice
Matching

ABSTRACT

We investigate the existence of approximation algorithms for maximization of submodular functions, that run in a fixed parameter tractable (FPT) time. Given a non-decreasing submodular set function $v: 2^X \to \mathbb{R}$ the goal is to select a subset $S$ of $K$ elements from $X$ such that $v(S)$ is maximized. We identify three properties of set functions, referred to as $p$-separability properties, and we argue that many real-life problems can be expressed as maximization of submodular, $p$-separable functions, with low values of the parameter $p$. We present FPT approximation schemes for the minimization and maximization variants of the problem, for several parameters that depend on characteristics of the optimized set function, such as $p$ and $K$.

© 2017 Elsevier Inc. All rights reserved.

## 1. Introduction

We study (exponential-time) approximation algorithms for maximizing non-decreasing submodular set functions. A set function $v: 2^X \to \mathbb{R}$ is submodular if for each two subsets $A \subseteq B \subset X$ and each element $x \in X \setminus B$ it holds that $v(A \cup \{x\}) - v(A) \geq v(B \cup \{x\}) - v(B)$; $v$ is non-decreasing if for each two subsets $A \subseteq B \subset X$ it holds that $v(A) \leq v(B)$. We consider the problem where the goal is to select a subset $S$ of $K$ elements from $X$ such that the value $v(S)$ is maximal.

Maximization of non-decreasing submodular functions is a very general problem that is extensively used in various research areas, from recommendation systems [1,2], through voting theory [3,1], image engineering [4–6], information retrieval [7,8], network design [9,10], clustering [11], speech recognition [12], to sparse methods [13,14]. Algorithms for maximization of non-decreasing submodular functions are applicable to other general problems of fundamental significance, such as the MaxCover problem [15,16]. The universal relevance of the problem implies that the existence of good (approximation) algorithms for it is highly desired.

Indeed, the problem has already received a considerable amount of attention in the scientific community. For instance, it is known that the problem is NP-hard (on the other hand, Iwata et al. [17] have shown that minimization of submodular functions is solvable in polynomial time) and that the greedy algorithm, i.e., the algorithm that starts with the empty set and in each of $K$ consecutive steps adds to the partial solution such an element from $X$ that increases the value of the optimized function most, is an $(1 - 1/e)$-approximation algorithm for maximization of non-decreasing submodular functions [18]. The same approximation ratio can be achieved for the distributed [19] and online [20] variants of the problem. Algorithms for maximizing non-monotone submodular functions have been studied by Feige et al. [21], and the approximability of the problem with additional constraints has been investigated by Calinescu et al. [22], Sviridenko [23], Lee et al. [24], and

☆ The preliminary version of this paper was presented at the 12th Conference on Web and Internet Economics (WINE-2016).
*E-mail address:* p.k.skowron@gmail.com.

https://doi.org/10.1016/j.ic.2017.10.002
0890-5401/© 2017 Elsevier Inc. All rights reserved.

Vondrák et al. [25]. For the survey on maximization of submodular functions we refer the reader to the work of Krause and Golovin [26].

Unfortunately, the approximation guarantees of the greedy algorithm cannot be improved without compromising the efficiency of computation (there are some notable exceptions when the submodular function has a specific structure, for instance when it has low curvature [27,28]). Indeed, the MaxCover problem can be expressed as maximization of a non-decreasing submodular function, yet it is known that under standard complexity assumptions no polynomial-time algorithm can approximate it better than with ratio $(1 - 1/e)$ [29]. Motivated by this fact, and provoked by the desire to obtain better approximation guarantees, we turn our attention to algorithms that run in super-polynomial time. In our studies we follow the approach of parameterized complexity theory and look for algorithms that run in fixed parameter tractable time (in FPT time), for some natural parameters. To the best of our knowledge, FPT approximation of optimizing submodular functions has not been considered in the literature before.

Parameterized complexity theory aims at investigating how the complexity of a problem depends on the size of different parts of input instances, called parameters. An algorithm runs in FPT time for a parameter $P$ if it solves each instance $I$ of the problem in time $O(f(|P|) \cdot \mathrm{poly}(|I|))$, where $f$ is a computable function. This definition excludes a large class of algorithms, such as the ones with complexity $O(|I|^{|P|})$. From the point of view of parameterized complexity, FPT is seen as the class of easy problems. Intuitively, the complexity of an FPT algorithm consists of two parts: $f(|P|)$, which is relatively low for small values of the parameter, and $\mathrm{poly}(|I|)$ which is relatively low even for larger instances, because of polynomial relation between the computation time and the size of an instance. For details on parameterized complexity theory, we point the reader to appropriate overviews [30–33].

When referring to running times of the algorithms we assume that the size of the input is $|X|$, and we count each access to the submodular function as an elementary operation (i.e., we assume that the access to the submodular function is given by an oracle). For example, when we say that an algorithm for maximizing submodular functions runs in polynomial-time, then in particular this means that the number of accesses of this algorithm to the submodular functions is of the order of $O(|X|^c)$ for some constant $c$.

We identify several parameters that we believe are suitable for the analysis of the complexity of the maximization of non-decreasing submodular functions. Perhaps the most natural parameter to consider is the required size of solutions, $K$. Our other parameters depend on characteristics of the optimized set function. Specifically, we define a new property of set functions, called $p$-separability, and provide evidence that $p$ is a natural parameter to consider. We do that in Section 4, by presenting several examples of real-life computational problems that can be expressed as maximization of submodular $p$-separable set functions, where the value of $p$ is small.

Our main contribution is a presentation and an analysis of a few algorithms for the problem. We construct fixed parameter tractable approximation schemes, i.e., collections of algorithms that run in FPT time and that can achieve arbitrarily good approximation ratios. We provide algorithms for two variants of the problem: in the first variant, referred to as the *maximization variant*, the goal is to maximize the value $v(S)$. In the second one, referred to as the *minimization variant*, the goal is to minimize $\big(v(X) - v(S)\big)$. While these two variants of the problem have the same optimal solutions, they are not equivalent in terms of their approximability. Indeed, if there exists a solution $S$ with objectively high value, i.e., if $v(S)$ is close to $v(X)$, then an approximation algorithm for the minimization variant of the problem will be usually superior. For instance, if there exists a solution $S$ such that $v(S) = 0.95 \cdot v(X)$, then a 2-approximation algorithm for the minimization variant of the problem is guaranteed to return a solution with the value better than $0.9 \cdot v(X)$. On the other hand, a $1/2$-approximation algorithm for the maximization variant of the problem may return in such a case a solution with value $0.475 \cdot v(X)$. Conversely, if the value of an optimal solution is significantly lower than the value of the whole set $X$, then a good approximation algorithm for the maximization variant of the problem will produce solutions of a better quality.

Our algorithms run in FPT time for the parameter $(K, p)$, where $K$ is the size of the solution, and $p$ is the lowest value such that the set function is $p$-separable. To address the case of functions which are not $p$-separable for any reasonable values $p$, we define a weaker form of approximability, referred to as approximation of the *minimization-or-maximization variant*—here, the goal is to find a subset $S$ that is good in one of the previous two metrics. Such algorithms are also desired as they are guaranteed to find good approximation solutions, provided high quality solutions exist (i.e., if values of the optimal solutions are close to $v(X)$). We show that there exists a randomized FPT approximation scheme for minimization-or-maximization variant of the problem for the parameter $\left(K, \frac{\sum_{x \in X} v(\{x\})}{v(X)}\right)$.

We believe that the consequences of our general results are quite significant. In particular, in Section 4, we prove the existence of FPT approximation schemes for some natural problems in the computational social choice, in the matching theory, and in the theoretical computer science.

## 2. Notation and definitions

Let $X$ denote the universe set. We consider a set function $v : 2^X \to \mathbb{R}$ that is non-negative, i.e., such that for each $S \subseteq X$ we have $v(S) \geq 0$. We recall that a set function $v$ is submodular if for each two subsets $A \subseteq B \subset X$ and each element $x \in X \setminus B$ it holds that $v(A \cup \{x\}) - v(A) \geq v(B \cup \{x\}) - v(B)$. There are numerous equivalent conditions characterizing submodular functions—for a survey we refer the reader to the seminal article of Nemhauser et al. [18]. It is easy to see that if the set function $v$ is non-decreasing and submodular, then for each two subsets $A \subseteq B \subset X$ and each element $x \in X$ it holds that $v(A \cup \{x\}) - v(A) \geq v(B \cup \{x\}) - v(B)$ (here, we do not have to assume that $x \in X \setminus B$).

Below, we define a new class of properties of set functions that we call $p$-separability.

**Definition 1** *($p$-separable set function).* A submodular set function $v : 2^X \to \mathbb{R}$ is:

1. *$p$-superseparable*, if for each $S \subseteq X$ we have:
$$\sum_{x \in X} \big(v(S \cup \{x\}) - v(S)\big) \geq \sum_{x \in X} v(\{x\}) - p \cdot v(S). \quad (1)$$
2. *$p$-subseparable*, if for each $S \subseteq X$ we have:
$$\sum_{x \in X} \big(v(S \cup \{x\}) - v(S)\big) \leq p \cdot v(X) - p \cdot v(S). \quad (2)$$
3. *rev-$p$-subseparable*, if for each $S \subseteq X$ we have:
$$\sum_{x \in X} \big(v(S \cup \{x\}) - v(S)\big) \geq p \cdot v(X) - p \cdot v(S). \quad (3)$$

We will show that for $p$-superseparable and $p$-subseparable functions, a smaller value of the parameter $p$ makes the problem easier to approximate. In contrast, for rev-$p$-subseparability, a larger value of $p$ is better.[1]

Each of the three conditions in Definition 1 imposes some bound on the expression $\sum_{x \in X} \big(v(S \cup \{x\}) - v(S)\big)$. This expression can be seen as the sum of marginal gains of function $v$ at point $S$. If we write $|\nabla v(S)| = \sum_{x \in X} \big(v(S \cup \{x\}) - v(S)\big)$, and additionally use a common assumption that $v(\emptyset) = 0$ then the condition for the $p$-superseparability can be rewritten as $\frac{|\nabla v(\emptyset)| - |\nabla v(S)|}{v(S)} \leq p$. Thus, $p$-superseparability imposes a specific constraint on how the norm of the gradient of $v$ changes relatively to the value of $v$. Similarly, in this language the condition for the $p$-subseparability can be rewritten as $\frac{|\nabla v(S)|}{v(X) - v(S)} \leq p$.

In particular, observe that a nonincreasing and submodular function $v$ is $p$-superseparable if $\sum_{x \in X} v(\{x\}) \leq p \cdot \min_{x:\ v(\{x\})>0} v(\{x\})$. Indeed, fix $S \subseteq X$ and consider two cases. If $v(S) = 0$ then Inequality (1) is clearly satisfied as function $v$ is nondecreasing. If $v(S) > 0$, then by submodularity of $v$, there exists $y \in S$ such that $v(\{y\}) > 0$. Thus, by monotonicity of $v$ we have that $p \cdot v(S) \geq p \cdot \min_{x:\ v(\{x\})>0} v(\{x\})$, and so by our assumption $p \cdot v(S) \geq \sum_{x \in X} v(\{x\})$. In this case, the right-hand side of Inequality (1) is negative or equal to zero. Since function $v$ is nonincreasing we infer that the left-hand side of Inequality (1) is non-negative, and consequently that the Inequality (1) holds. Also, it is easy to see that each monotone and submodular function is $|X|$-superseparable and $|X|$-subseparable. Yet, in Section 4 we show that the value of parameter $p$ in many natural problems is significantly lower than $|X|$.

Further, observe that a linear combination with positive coefficients of $p$-superseparable functions is $p$-superseparable. The same comment applies to $p$-subseparability and rev-$p$-subseparability. As we will see in Section 4, this observation is very helpful in proving that certain set functions are $p$-separable.

Example 1 below illustrates how a variant of MaxCover, a fundamental problem in the theoretical computer science, can be expressed as a maximization of a non-negative, nondecreasing, $p$-separable, submodular function.

**Example 1.** In the MaxWeightCover problem, we are given a set of $n$ elements, $N = \{e_1, e_2, \dots e_n\}$, and a collection of $m$ subsets of $N$, $X = \{S_1, \dots, S_m\}$. Each element $e_i$ has a weight $w_i$, and the goal is to find a subcollection of $K$ subsets from $X$, $\mathcal{C}$, which maximizes the total weight of the covered elements: $\sum_{i: i \in \bigcup \mathcal{C}} w_i$. A *frequency* of an element $e_i$ is defined as the number of sets that contain $e_i$.

We will show that the MaxWeightCover problem with the frequency of elements upper-bounded by $p$ can be expressed as the maximization of a non-negative, nondecreasing submodular function which is (i) $p$-superseparable, and (ii) $p$-subseparable. For each set $\mathcal{C} \subseteq X$ we define $v(\mathcal{C})$ as the total weight of elements covered by the sets from $\mathcal{C}$. Such defined $v$ is non-negative and submodular. Since the weighted sum of $p$-separable set functions is also $p$-separable, it is sufficient to show $p$-separability of a function $u_i$ which returns 1 for collections of sets that cover $e_i$, and 0 for the remaining ones. Since the frequency of the elements is bounded by $p$, then $\sum_{S \in X} u_i(\{S\})$, which is the number of sets that cover $e_i$, is also bounded by $p$. Further, $p \cdot \min_{S:\ u_i(\{S\})>0} u_i(\{S\}) = p$, thus the right-hand side of Inequality (1) is negative or equal to zero: this shows that $u_i$ is $p$-superseparable. To show that $u_i$ is also $p$-subseparable, let us fix a collection of sets $\mathcal{C} \subseteq X$ and let us consider two cases. If $e_i$ is covered by $\mathcal{C}$, then $\sum_{S \in X} \big(u_i(\mathcal{C} \cup \{S\}) - u_i(\mathcal{C})\big)$ is equal to 0. The right-hand side of Inequality (2) is non-negative, thus the condition for $p$-subseparability holds. If $e_i$ is not covered by $\mathcal{C}$, then $\sum_{S \in X} \big(u_i(\mathcal{C} \cup \{S\}) - u_i(\mathcal{C})\big)$ is equal to the number of sets that cover $e_i$, thus it is upper bounded by $p$. The right-hand side of Inequality (2) is equal to $p \cdot v(X) - p \cdot v(\mathcal{C}) = p \cdot 1 - p \cdot 0 = p$. Thus the condition for $p$-subseparability also holds.

Similarly, it can be shown that the MaxWeightCover problem with the frequency of elements lower-bounded by $p$ can be expressed as the maximization of a non-negative, nondecreasing submodular function which is rev-$p$-subseparable.

[1] In fact, for all our results to hold, in Definition 1 we do not have to consider all the subsets $S$ of $X$ but only those with the size at most equal to $K$.

For a better intuition on the above definitions, we refer the reader to Section 4 where we present several further examples of natural problems which can be expressed as optimization of super/sub-separable functions for low values of the parameter $p$ or rev-subseparable functions for high values of $p$.

A different parameter of submodular functions that has been studied in the literature is the curvature. We say that a function $v$ has curvature $c$, $0 \leq c \leq 1$, if for each $S \subset X$ and $x \in X \setminus S$ it holds that $v(S \cup \{x\}) - v(S) \geq (1-c)v(\{x\})$. It is known that the greedy algorithm achieves approximation ratio of $\frac{1-e^{-c}}{c}$ for the problem of maximizing a submodular function with curvature $c$ [27]. Recently, Sviridenko et al. [28] have improved this result by designing a $(1 - c/e)$-approximation algorithm for the problem. The curvature is not comparable to $p$-separability: there are $p$-separable functions for very good values of the parameter $p$, which at the same time have curvature equal to one. For instance, consider the submodular function $v$ for MaxWeightCover from Example 1. If the set of elements $N$ can be covered with a subcollection $\mathcal{C}$, $\mathcal{C} \neq X$, then for $S \in X \setminus \mathcal{C}$ it holds that $v(\{S\}) > 0$ and $v(\mathcal{C} \cup \{S\}) - v(\mathcal{C}) = v(X) - v(X) = 0$, so the optimized function has curvature equal to one.

In this paper we investigate the problem of selecting $K$ elements from $X$ that altogether maximize the value of the set function $v$.

**Problem 1** *(BestKSubset).* For a set of elements $X$, a polynomially computable set function $v : 2^X \to \mathbb{R}$, and an integer $K$, the solution to the BestKSubset problem is a set $S \subseteq X$ with $|S| \leq K$, for which $v(S)$ is maximal.

We are specifically interested in finding approximation algorithms for the BestKSubset problem. We focus on approximating two metrics: (i) the value $v(S)$ in the maximization variant of the problem, and (ii) the value $\big(v(X) - v(S)\big)$ in its minimization variant.

**Definition 2** *(Approximation algorithms).* Let $S^*$ denote an optimal solution for BestKSubset:

1. Fix $\alpha$, $0 < \alpha < 1$. $\mathcal{A}$ is an *$\alpha$-approximation algorithm for the maximization variant* of BestKSubset, if for each instance $I$ of BestKSubset it returns a set $S$ such that $v(S) \geq \alpha v(S^*)$.
2. Fix $\alpha$, $\alpha > 1$. $\mathcal{A}$ is an *$\alpha$-approximation algorithm for the minimization variant* of BestKSubset, if for each instance $I$ of BestKSubset it returns a set $S$ such that $\big(v(X) - v(S)\big) \leq \alpha\big(v(X) - v(S^*)\big)$.
3. Fix $\alpha$, $\alpha > 1$. $\mathcal{A}$ is an *$\alpha$-approximation algorithm for the minimization-or-maximization variant* of BestKSubset, if for each instance $I$ of BestKSubset it returns a set $S$ such that $v(S) \geq \frac{1}{\alpha} v(S^*)$ or $\big(v(X) - v(S)\big) \leq \alpha\big(v(X) - v(S^*)\big)$.

The definition of an approximation algorithm for minimization-or-maximization variant of BestKSubset requires some additional comment: this definition guarantees that the algorithm finds a good solution provided a high quality solution exists. In other words, if there exists an optimal solution $S^*$ such that the value $\big(v(X) - v(S^*)\big)$ is low compared to $v(S^*)$, then a good approximation solution for the minimization variant of the problem is also a good solution for its maximization variant. For some parameters we present good approximation algorithms for the minimization-or-maximization variant of BestKSubset, even though we do not have as good algorithms neither for the minimization nor maximization variants of the problem.

We are specifically interested in FPT approximation schemes. A collection of algorithms $\mathcal{A}$ is an FPT approximation scheme for a parameter $P$, if for each constant $\alpha$ there exists an $\alpha$-approximation algorithm in $\mathcal{A}$ that runs in an FPT time for the parameter $P$.

## 3. Algorithms for maximizing p-separable submodular functions

In this section we present our approximation algorithms for the three variants of the BestKSubset problem, formally stated in Definition 2. Our methods are inspired by the algorithms of Skowron and Faliszewski [16] for the MaxCover problem. We extend these algorithms to be applicable to the problem of maximizing more general submodular functions.

We start by presenting an FPT approximation scheme for the maximization variant of BestKSubset for submodular $p$-superseparable set functions. The algorithm, formally defined as Algorithm 1, gets as an input an instance of the problem and the required approximation ratio, $\alpha$. It proceeds in two steps: first, it restricts the universe set by selecting a certain number of elements from $X$ with the highest values of the set function $v$. Second, it takes the set $\mathcal{A}$ of elements that were selected in the first step, computes the value of the set function for all $K$-element subsets of $\mathcal{A}$, and returns a subset with the highest value.

Algorithm 1 is an FPT approximation scheme for the maximization variant of the problem for the parameter $(K, p)$. Before we prove this fact, however, we note that under standard complexity theoretic assumptions, there exists no FPT exact algorithm for the problem. There even exists no FPT exact algorithm for the parameter $K$ if $p$ is a constant. This follows from our observation in Example 1, where we show that the MaxCover problem with frequencies bounded by $p$ can be expressed as maximization of a non-negative, non-decreasing, submodular, $p$-superseparable set function, and from the fact that the MaxCover problem with frequencies bounded by a constant, for the parameter $K$ belongs to the complexity class W[1] [16], and it is unlikely that W[1] $\subseteq$ FPT. Further, even for $p = 2$ there exists no polynomial-time approximation

**Algorithm 1:** An algorithm for the BestKSubset problem for non-negative, non-decreasing, submodular, and $p$-superseparable set functions.

```
Parameters:
  $X$ — the set of elements.
  $v$ — the submodular function $v : 2^X \to \mathbb{R}$ that is $p$-superseparable.
  $\alpha$ — the required approximation ratio of the algorithm.
$\mathcal{A} \leftarrow \lceil \frac{pK}{(1-\alpha)} + K \rceil$ elements $x$ from $X$ with highest values $v(\{x\})$ ;
return $K$-element subset of $\mathcal{A}$ with the highest value of $v$ ;
```

scheme (PTAS) for the problem. This is because the MaxCover problem with the frequencies of the elements equal to two is a generalization of the MaxVertexCover problem, for which there is no PTAS unless $\mathrm{P} = \mathrm{NP}$ (see, e.g., the work of Croce and Paschos [34] for a discussion on the approximability of the MaxVertexCover problem).

**Theorem 1.** *For each non-negative, non-decreasing, submodular, and $p$-superseparable set function $v : 2^X \to \mathbb{R}$ and for each $0 \le \alpha < 1$, Algorithm 1 outputs an $\alpha$-approximate solution for the maximization variant of* BestKSubset, *in time* $\mathrm{poly}(n, m) \cdot \binom{\frac{pK}{(1-\alpha)} + K}{K}$.

**Proof.** Consider an input instance $I$ of the BestKSubset problem. Let $S$ and $S^*$ be, respectively, the solution returned by Algorithm 1 and some optimal solution. We set $\mathrm{OPT} = v(S^*)$ as the value of an optimal solution.

We will show that $v(S) \ge \alpha\mathrm{OPT}$. Naturally, the value $v(S)$ might be lower than $v(S^*)$. This might happen because $\mathcal{A}$, the set of the elements considered by the algorithm in its second step, might not contain some elements from $S^*$. We will show that $\ell = |S^* \setminus \mathcal{A}|$ elements from $S^* \setminus \mathcal{A}$ might be replaced by some elements from $\mathcal{A}$ which are not present in $S^*$, in a way that decreases the value of $S^*$ by at most a small fraction. After such replacement, we will end up with the set containing the elements from $\mathcal{A}$ only. From this we will infer that the value of the best solution in $\mathcal{A}$ is lower than the value of an optimal solution by at most a small factor.

Let us order the elements from $S^* \setminus \mathcal{A}$ in some arbitrary way, and let us use the notation $S^* \setminus \mathcal{A} = \{x_1, \dots, x_\ell\}$. We will replace the elements $\{x_1, \dots, x_\ell\}$ with the elements $\{x'_1, \dots, x'_\ell\}$ from $\mathcal{A} \setminus S^*$ (we will define these elements later), one by one, in $\ell$ consecutive steps. Let $S_i$ denote the set that we get after replacing elements $x_1, \dots, x_i$ with $x'_1, \dots, x'_i$, that is let $S_i = (S^* \setminus \{x_1, \dots, x_i\}) \cup \{x'_1, \dots, x'_i\}$. In particular, $S_\ell \subseteq \mathcal{A}$. Thus, in the $i$-th step we will replace $x_i$ with $x'_i$ in set $S_{i-1}$.

The elements $x'_1, \dots, x'_\ell$ are defined by induction, in the following way. Assume that we have already found elements $x'_1, \dots, x'_{i-1}$ (for $i = 1$ it means we have not yet found any element, i.e., that we are looking for the first element in the sequence). We define $x'_i$ to be an element from $\mathcal{A} \setminus S_{i-1}$ that maximizes $v\Big((S_{i-1} \setminus \{x_i\}) \cup \{x'_i\}\Big)$, that is when selecting $x'_i$ we aim at maximizing $v(S_i)$.

It may happen that after replacing $x_i$ with $x'_i$, the value of the function $v$ for the new set decreases. Let $\Delta_i$ denote the value of such decrease (or increase if the algorithm were lucky—in such case $\Delta_i$ would be negative):

$$\Delta_i = v(S_{i-1}) - v(S_i).$$

By the construction of the set $\mathcal{A}$ and the fact that $x_i \notin \mathcal{A}$, for every $y \in \mathcal{A} \setminus S_{i-1}$ we have that $v(\{x_i\}) \le v(\{y\})$. By the way we choose the element $x'_i$, we know that for every $y \in \mathcal{A} \setminus S_{i-1}$, we have:

$$\Delta_i \le v(S_{i-1}) - v\Big((S_{i-1} \setminus \{x_i\}) \cup \{y\}\Big).$$

Using submodularity and after reformulation we get:

$$\begin{aligned}\Delta_i &\le v\Big(S_{i-1} \setminus \{x_i\}\Big) + v\Big(\{x_i\}\Big) - v(\emptyset) - v\Big((S_{i-1} \setminus \{x_i\}) \cup \{y\}\Big)\\ &\le v\Big(S_{i-1} \setminus \{x_i\}\Big) - v\Big((S_{i-1} \setminus \{x_i\}) \cup \{y\}\Big) + v\Big(\{y\}\Big) - v(\emptyset).\end{aligned}$$

For any $y \in X$ (in particular for $y \notin \mathcal{A} \setminus S_{i-1}$), by submodularity and monotonicity, we have that:

$$0 \le v\Big(S_{i-1} \setminus \{x_i\}\Big) + v\Big(\{y\}\Big) - v(\emptyset) - v\Big((S_{i-1} \setminus \{x_i\}) \cup \{y\}\Big).$$

Since the set function is non-negative, the inequalities above will still hold if we skip the fragment $v(\emptyset)$. Consequently, since the set function is $p$-superseparable, we get:

$$(|\mathcal{A}| - K)\Delta_i \le |\mathcal{A} \setminus S_{i-1}|\Delta_i = \sum_{y \in \mathcal{A} \setminus S_{i-1}} \Delta_i + \sum_{y \in X \setminus (\mathcal{A} \setminus S_{i-1})} 0$$

**Algorithm 2:** An algorithm for the BestKSubset problem for non-negative, non-decreasing, rev-$p$-subseparable set functions.

**Parameters**:
$X$ — the set of elements.
$v$ — the submodular function $v : 2^X \to \mathbb{R}$ that is rev-$p$-subseparable.

```
$S = \{\}$;
for $i \leftarrow 1$ to $K$ do
    $S \leftarrow S \cup \left\{\mathrm{argmax}_{x \in X}\left(v(S \cup \{x\} - v(S))\right)\right\}$
return C
```

$$\leq \sum_{y \in X}\left(v\left(S_{i-1} \setminus \{x_i\}\right) + v\left(\{y\}\right) - v\left((S_{i-1} \setminus \{x_i\}) \cup \{y\}\right)\right)$$
$$\leq p \cdot v\left(S_{i-1} \setminus \{x_i\}\right) \leq p\mathrm{OPT}.$$

Which leads to:

$$\Delta_i \leq \frac{p\mathrm{OPT}}{|\mathcal{A}| - K} \leq \frac{\mathrm{OPT}p(1-\alpha)}{pK} = \frac{\mathrm{OPT}(1-\alpha)}{K}.$$

Since $\ell \leq K$, we conclude that:

$$\sum_{i=1}^{\ell} \Delta_i \leq (1-\alpha)\mathrm{OPT}.$$

That is, after replacing the elements from $S^*$ that do not appear in $\mathcal{A}$ with sets from $\mathcal{A}$, the optimal value decreases by at most $(1-\alpha)$OPT. This means that there are $K$ elements in $\mathcal{A}$ for which the function $v$ achieves the value equal to at least $\alpha$OPT. Since the algorithm tries all size-$K$ subsets of $\mathcal{A}$, it finds a solution with such a value. $\square$

Now, we move to optimizing $p$-subseparable set functions. We know that the simple greedy algorithm (Algorithm 2) achieves approximation ratio of $1 - 1/e$ for the maximization variant of the BestKSubset problem [18]. Below, we show that if the optimized set function is rev-$p$-subseparable then for some values of the parameter $p$ the analysis of the approximation guarantees of this simple greedy algorithm can be further improved. We note that in this case we do not even require the set function to be submodular.

**Proposition 1.** *The greedy algorithm (Algorithm 2) is a polynomial-time $\left(1 - e^{-\frac{pK}{|X|}}\right)$-approximation algorithm for the maximization variant of* BestKSubset *problem with a non-negative, non-decreasing, rev-$p$-subseparable set function.*

**Proof.** The algorithm clearly runs in polynomial time and so we focus only on proving its approximation ratio.

We prove by induction that for each $i$, $0 \leq i \leq K$, after the $i$'th iteration of the greedy algorithm's "for" loop, the value $\left(v(X) - v(S)\right)$ is at most equal to $\left(v(X) - v(\emptyset)\right)\left(1 - \frac{p}{|X|}\right)^i$. Naturally, the assumption is true for $i = 0$. Suppose that the inductive assumption holds for some $(i-1)$, $1 \leq i < K$. Let $S$ be the partial solution at the beginning of the $i$-th iteration of the algorithm's "for" loop and let $x_b$ be the element added to the partial solution in this iteration. Since the set function $v$ is rev-$p$-subseparable, it holds that:

$$\sum_{x \in X}\left(v(S \cup \{x\}) - v(S)\right) \geq p \cdot v(X) - p \cdot v(S).$$

From the pigeonhole principle we get that:

$$\left(v(S \cup \{x_b\}) - v(S)\right) \geq \frac{p}{|X|}\left(v(X) - v(S)\right).$$

Thus, we get that:

$$v(X) - v(S \cup \{x_b\}) \leq v(X) - v(S) + v(S) - v(S \cup \{x_b\})$$
$$\leq v(X) - v(S) - \frac{p}{|X|}\left(v(X) - v(S)\right)$$
$$= \left(v(X) - v(S)\right)\left(1 - \frac{p}{|X|}\right) \leq \left(v(X) - v(\emptyset)\right)\left(1 - \frac{p}{|X|}\right)^i.$$

Let $C$ be the solution returned by Algorithm 2.

**Algorithm 3:** An algorithm for the minimization variant of the BestKSubset problem with a non-negative, non-decreasing, submodular, and $p$-subseparable set function.

```
Parameters:
  $X$ — the set of elements.
  $v$ — the submodular function $v : 2^X \to \mathbb{R}$ that is $p$-subseparable.
  $\alpha$ — the required approximation ratio of the algorithm
  $\epsilon$ — the allowed probability of achieving worse than $\alpha$ approximation ratio
SingleRun():
    $S \leftarrow \emptyset$;
    for $i \leftarrow 0$ to $K$ do
        $x_r \leftarrow$ randomly select an element from $X \setminus S$
                with probability of selecting $x$ proportional to $v(S \cup \{x\}) - v(S)$ ;
        $S \leftarrow S \cup \{x_r\}$;
    return $S$;
Main(): run SingleRun() for $\left\lceil (-\ln \epsilon) / \left(\frac{\alpha-1}{p\alpha}\right)^K \right\rceil$ times; return the best solution;
```

$$v(X) - v(C) \leq \Big(v(X) - v(\emptyset)\Big)\Big(1 - \frac{p}{|X|}\Big)^{\frac{|X|}{p} \cdot \frac{pK}{|X|}} \leq \Big(v(X) - v(\emptyset)\Big)e^{-\frac{pK}{|X|}}.$$

Since the function $v$ is monotonic, and thus $\mathrm{OPT} \leq v(X)$, and since $v$ is non-negative, and thus $v(\emptyset) \geq 0$, we have that:

$$v(C) \geq v(X) - \Big(v(X) - v(\emptyset)\Big)e^{-\frac{pK}{|X|}} \geq \mathrm{OPT}\Big(1 - e^{-\frac{pK}{|X|}}\Big).$$

This completes the proof. □

If we additionally assume that the set function is submodular, then naturally, the standard approximation ratio of $1 - 1/e$ of the greedy algorithm for maximizing submodular functions still applies and we can strengthen approximation guarantee from Proposition 1 to $\Big(1 - e^{-\max\left(\frac{pK}{|X|}, 1\right)}\Big)$. Proposition 1 has interesting implication: if we restrict our problem to the class of instances for which $\frac{p}{|X|}$ is lower-bounded by some constant, then there exists a polynomial time approximation scheme (PTAS) for such a restricted problem. This observation is interesting, because for some real-life problems it is more natural to express the value of the parameter $p$ as the fraction of the size of the universe set $X$ rather than as the absolute value.

**Corollary 1.** *Let $\gamma \in \mathbb{R}$ be a constant. There exists a polynomial time approximation scheme for the maximization variant of the* BestKSubset *problem with non-negative, non-decreasing, and rev-$(\gamma|X|)$-subseparable set function.*

**Proof.** For each $\epsilon$, $0 < \epsilon < 1$, there exists such a constant $c$ that for $K > c$ we have $\big(1 - e^{-\gamma K}\big) \geq 1 - \epsilon$. For $K > c$ we run the greedy algorithm (and the approximation ratio follows from Proposition 1), and for $K \leq c$, we invoke a brute-force algorithm that tries all $K$-element subsets of $X$. □

Finally, we consider the minimization variant of BestKSubset for the case of *$p$-subseparable* submodular set functions. In Algorithm 3 we present a randomized algorithm for the problem: the algorithm performs several independent runs. Each run, in Algorithm 3 described by the `SingleRun` procedure, builds the solution by selecting random elements in $K$ consecutive steps. In each step, an element $x$ is selected with the probability proportional to the marginal increase of the value of the set function caused by adding $x$ to the partial solution. Theorem 2 below shows that if we repeat such a procedure a sufficient number of times, we are very likely to find a solution with the required approximation ratio. Before we state Theorem 2 we will prove a more technical lemma which will appear useful in proving the theorem, but also in our further analysis.

**Lemma 1.** *Consider a single iteration of the "for" loop within the function* `SingleRun` *in Algorithm 3. Let $S^*$ be some optimal solution for $I$, let $S$ be the value of the partial solution at the beginning of this iteration and let $p_{hit}$ denote the probability that in this iteration any element from $S^*$ is added to the partial solution (thus, using notation from Algorithm 3, $p_{hit}$ is the probability that $x_r \in S^*$). We have that:*

$$p_{hit} \geq \frac{v(S^*) - v(S)}{\sum_{x \in X}\Big(v(S \cup \{x\}) - v(S)\Big)}.$$

**Proof.**

$$\begin{aligned}
p_{hit} &= \frac{\sum_{x\in S^*}\Big(v(S\cup\{x\})-v(S)\Big)}{\sum_{x\in X}\Big(v(S\cup\{x\})-v(S)\Big)} \\
&\geq \frac{\Big(v(S\cup\{x_1\})-v(S)\Big)+\Big(v(S\cup\{x_1,x_2\})-v(S\cup\{x_1\})\Big)+\dots}{\sum_{x\in X}\Big(v(S\cup\{x\})-v(S)\Big)} && \text{submodularity} \\
&= \frac{v(S\cup S^*)-v(S)}{\sum_{x\in X}\Big(v(S\cup\{x\})-v(S)\Big)} \\
&\geq \frac{v(S^*)-v(S)}{\sum_{x\in X}\Big(v(S\cup\{x\})-v(S)\Big)} && \text{non-decreasing} \qquad \square
\end{aligned}$$

**Theorem 2.** *For each non-negative, non-decreasing, submodular, $p$-subseparable set function $v : 2^X \to \mathbb{R}$ and for each $0 \leq \alpha < 1$, Algorithm 3 outputs an $\alpha$-approximate solution for the minimization variant of* BestKSubset*, with probability $(1-\epsilon)$. The time complexity of the algorithm is* $\mathrm{poly}(n, m)\cdot\left\lceil(-\ln\epsilon)/\left(\frac{\alpha-1}{p\alpha}\right)^K\right\rceil$.

**Proof.** Let $I$ be an instance of the BestKSubset problem with $v : 2^X \to \mathbb{R}$ being a non-negative, submodular, $p$-subseparable function. Let $\alpha$, $\alpha > 1$, and $\epsilon$, $0 < \epsilon < 1$ be the parameters of Algorithm 3. Let $S^*$ be some optimal solution for $I$.

Let us consider a single call to `SingleRun` from function `Main`. Let $p_s$ denote the probability that such a single invocation of function `SingleRun` returns an $\alpha$-approximate solution. We will prove the lower-bound of $\left(\frac{\alpha-1}{p\alpha}\right)^K$ for the value of $p_s$. Let $Ev$ denote the event that during such an invocation, at the beginning of some iteration of the "for" loop within the function `SingleRun`, it holds that:

$$v(X)-v(S) \leq \alpha\Big(v(X)-v(S^*)\Big). \tag{4}$$

If this event occurs, then there is an iteration where we obtain an $\alpha$-approximate solution and the value of the solution can only increase in further iterations. Thus, in such a case `SingleRun` definitely returns an $\alpha$-approximate solution. A condition reverse to Inequality (4), $v(X)-v(S) < \alpha\Big(v(X)-v(S^*)\Big)$, can be equivalently formulated as follows:

$$\frac{v(S^*)-v(S)}{v(X)-v(S)} > \frac{\alpha-1}{\alpha}. \tag{5}$$

Now, let us consider a single iteration of the "for" loop within the function `SingleRun`, and let us use the notation from Lemma 1. If $S$ (the value of the partial solution at the beginning of the considered iteration) satisfies Inequality (5), then we have:

$$\begin{aligned}
p_{hit} &\geq \frac{v(S^*)-v(S)}{\sum_{x\in X}\Big(v(S\cup\{x\})-v(S)\Big)} && \text{Lemma 1} \\
&\geq \frac{v(S^*)-v(S)}{p\Big(v(X)-v(S)\Big)} && p\text{-subseparability} \\
&\geq \frac{\alpha-1}{p\alpha}. && \text{Inequality (5)}
\end{aligned}$$

In each iteration of an invocation of the `SingleRun` function one of the two things can happen. The even $Ev$ can occur—in such a case we are sure that the function will return an $\alpha$-approximate solution—or it may not occur, and so at the beginning of the next iteration Inequality (4) would hold. In the latter case, by our previous reasoning we know that the probability of selecting an element from $S^*$ in this next iteration is at least equal to $\frac{\alpha-1}{p\alpha}$. Clearly, if in each iteration an element from $S^*$ is selected, then we eventually get an optimal solution, which in particular is an $\alpha$-approximate solution. Consequently, we can lower-bound $p_s$ by $\left(\frac{\alpha-1}{p\alpha}\right)^K$.

To conclude, we use the standard argument that if we make $x = \lceil\frac{-\ln\epsilon}{p_s}\rceil$ independent calls to `SingleRun`, then the best output from these calls is an $\alpha$-approximate solution with probability at least equal to:

$$1-\big(1-p_s\big)^x \geq 1-e^{\ln\epsilon} = 1-\epsilon.$$

This completes the proof. $\square$

Interestingly, we can slightly modify the proof of Theorem 2 so that it would apply with the parameter $\frac{\sum_{x \in X} v(\{x\})}{v(X)}$ (for this parameter we do not use the notion of $p$-separability). On the other hand, for this parameter we give weaker approximation guarantees, by approximating the minimization-or-maximization instead of the minimization variant of the problem.

**Theorem 3.** *For each non-negative, non-decreasing and submodular set function $v : 2^X \to \mathbb{R}$ there exists a randomized FPT approximation scheme for the minimization-or-maximization variant of* BestKSubset *problem with the parameter $(K, \frac{\sum_{x \in X} v(\{x\})}{v(X)})$.*

**Proof.** Let us fix $\alpha$, $\alpha > 1$, the required approximation ratio. Let $p = \frac{\alpha}{\alpha - 1} \cdot \frac{\sum_{x \in X} v(\{x\})}{v(X)}$. We will show that Algorithm 3 with such value of the parameter $p$ (this parameter is used to determine the number of iterations of the algorithm) is an $\alpha$-approximation algorithm for the minimization-or-maximization variant of the problem. We repeat the reasoning from the proof of Theorem 2, with the following small modification. In the proof of Theorem 2 we defined $Ev$ to denote the event that during a single invocation of the `SingleRun` function from Algorithm 3, at the beginning of some iteration of the "for" loop, it holds that: $v(X) - v(S) \leq \alpha\Big(v(X) - v(S^*)\Big)$. In this proof we modify this definition saying that $Ev$ denotes the event when at the beginning of some iteration of the "for" loop within the function `SingleRun`, at least one the following two conditions hold:

$$v(X) - v(S) \leq \alpha\Big(v(X) - v(S^*)\Big),$$
$$v(S) \geq \frac{1}{\alpha} v(S^*).$$

Naturally, if this event occurs, then `SingleRun` definitely returns an $\alpha$-approximate solution for the minimization-or-maximization variant of the problem. Similarly as in the proof of Theorem 2, we will assess $p_{hit}$, the probability that in a given iteration of the "for" loop, the algorithm selects an element from $S^*$ and adds it to the partial solution $S$, under the condition that $Ev$ has not yet occurred. Assuming that $Ev$ does not hold, we have that $v(X) - v(S) > \alpha\Big(v(X) - v(S^*)\Big)$ and $v(S) < \frac{1}{\alpha} v(S^*)$. From the first condition we have that $v(X) > \alpha\Big(v(X) - v(S^*)\Big)$ and so $v(X)(\alpha - 1) < \alpha(S^*)$.

$$\begin{aligned}
p_{hit} &\geq \frac{v(S^*) - v(S)}{\sum_{x \in X}\Big(v(S \cup \{x\}) - v(S)\Big)} && \text{Lemma 1} \\
&\geq \frac{v(S^*) - v(S)}{\sum_{x \in X}\Big(v(\{x\}) - v(\emptyset)\Big)} && \text{submodularity} \\
&\geq \frac{v(S^*) - v(S)}{\sum_{x \in X} v(\{x\})} = \frac{1}{p} \cdot \frac{\alpha}{\alpha - 1} \cdot \frac{v(S^*) - v(S)}{v(X)} \\
&\geq \frac{1}{p} \cdot \frac{\alpha}{\alpha - 1} \cdot \frac{v(S^*) - \frac{1}{\alpha} v(S^*)}{v(X)} && Ev \text{ has not occurred} \\
&= \frac{1}{p} \cdot \frac{v(S^*)}{v(X)} \geq \frac{1}{p} \cdot \frac{\alpha - 1}{\alpha}. && Ev \text{ has not occurred}
\end{aligned}$$

We get exactly the same estimation as in the proof of Theorem 2. Thus, with these modifications the proof of Theorem 2 can be used in this case. □

Algorithm 3 can be applied to yet another variant of the problem. Let BestSubset be defined similarly to BestKSubset, with the following difference. In BestSubset we are not putting any constraints on the size of the solution, but we rather look for the smallest possible set $S$ such that $v(S) = v(X)$. Interestingly, Algorithm 3 can be used to find *exact* solutions to BestSubset for non-negative, non-decreasing, submodular, $p$-subseparable set functions, and it will run in FPT time for the parameter $(K, p)$.

**Theorem 4.** *For each non-negative, non-decreasing, submodular, $p$-subseparable set function $v : 2^X \to \mathbb{R}$, the algorithm that runs Algorithm 3 for consecutive values of the parameter $K$ until it finds a solution $S$, such that $v(S) = v(X)$, is a randomized FPT exact algorithm for the* BestSubset *problem for the parameter $(K, p)$.*

**Proof.** Let $S^*$ be an optimal solution for the problem. Let us consider a single iteration of the "for" loop within the function `SingleRun`, and let $S$ be the value of the partial solution at the beginning of this iteration. We define $p_{hit}$ as the probability that in this iteration, an element from $S^*$ is added to the partial solution. We can use a very similar estimation as in the proof of Theorem 2:

$$\begin{aligned}
p_{hit} &\geq \frac{v(S^*) - v(S)}{\sum_{x \in X}\Big(v(S \cup \{x\}) - v(S)\Big)} && \text{Lemma 1} \\
&\geq \frac{v(S^*) - v(S)}{p\Big(v(X) - v(S)\Big)} && p\text{-subseparability} \\
&= \frac{1}{p}. && \text{since } v(S^*) = v(X)
\end{aligned}$$

Thus, we can lower bound the probability that $S^*$ will be found by a single run of the function `SingleRun`, by $\frac{1}{p^K}$. If we invoke `SingleRun` a sufficient number of times, then the probability of not finding an optimal solution can be decimated. □

## 4. Applications of the algorithms

In this section we show that the assumption about $p$-separability of submodular set functions is plausible. We provide several examples of known computational problems that can be expressed as maximization of $p$-separable, submodular functions. Consequently, we show that the algorithms from Section 3 are applicable to these problems.

### 4.1. Item selection in multi-agent systems

Let $N = \{1, 2, \dots n\}$ be the set of agents and let $C = \{a_1, a_2, \dots, a_m\}$ be the set of *items*. Each agent $i \in N$ is endowed with a *utility function* $u_i : C \to \mathbb{R}$ that measures how much $i$ desires each of the items. Our goal is to select $K$ items, called *winners*, that in some sense would make the agents most satisfied. An ordered weighted average (OWA) vector $\lambda$ is a vector of $K$ reals, $\lambda = \langle \lambda_1, \dots, \lambda_K \rangle$. Given an OWA vector $\lambda$, for each agent $i$ and for each set of $K$ items $S$, we define $u_i(S)$, the satisfaction of $i$ from $S$, in the following way. Let $z_{i,j}(S)$ denote the $j$-th most preferred, from the perspective of agent $i$, element of $S$, that is $z_{i,1}(S), z_{i,2}(S), \dots z_{i,K}(S)$ are the utilities from $\{u_i(x) : x \in S\}$ sorted in the descending order. Then $u_i(S) = \sum_{j=1}^{K} \lambda_j z_j(S)$. The satisfaction of all agents from $S$ is defined as the sum of satisfactions of all the individuals from $S$. For a fixed OWA vector $\lambda$, the problem is, for a given profile of utility functions and for a given integer $K$, to select a subset of $K$ items $S$ that maximizes the total satisfaction of the agents from $S$.

This model captures various natural problems, from winner determination in multiwinner election systems, through recommendation systems, to location problems. For instance the problem of selecting $K$ items under the OWA vector $\lambda = \langle 1, 0, \dots, 0 \rangle$ boils down to the problem of winner determination under Chamberlin and Courant rule [35], or to the facility location problem [36]. The problem for $\lambda = \langle 1, 1/2, \dots, 1/K \rangle$ is equivalent to winner determination in the Proportional Approval Voting (PAV) system [37]. For more examples of applications of this generic model we refer the reader to the original work of Skowron et al. [2].

We say that the agents have $k$-approval utilities if each agent assigns utility equal to 1 to exactly $k$ items, and utility equal to 0 to the remaining ones. Such $k$-approval utilities are very popular in the context of social choice, in particular in case of multi-winner election rules [37–39,16,40].

Even if the agents have $k$-approval utilities and even for the simple OWA vector $\lambda = \langle 1, 0, \dots, 0 \rangle$ the problem of selecting $K$ items is NP-hard [41] and hard from the perspective of parameterized complexity theory [42]. Also, it is known that even under these simplifying assumptions, no polynomial-time algorithm can approximate the problem with a better ratio than $1 - 1/e$. Thus, this is interesting to see that Algorithm 1 and Algorithm 3 are applicable to the problem of selecting $K$ items in the generic OWA-based framework with $k$-approval utilities.

**Lemma 2.** *For a non-decreasing OWA vector and for k-approval utilities, the problem of selecting K best items can be expressed as the maximization of a non-negative, nondecreasing submodular function which is (i) k-superseparable, and (ii) k-subseparable.*

**Proof.** We start from defining an appropriate set function $v$. It is easy to extend the definition of the utility of an agent $i$ from a set of items $S$, to be applicable to sets with a different number of elements than $K$: $u_i(S) = \sum_{j=1}^{\min(K,|S|)} \lambda_j z_{i,j}(S)$. For each set $S \subseteq C$, we define $v(S)$ as the total satisfaction of the agents from $S$: $v(S) = \sum_{i \in N} u_i(S)$.

Skowron et al. [16] showed that such defined function is submodular. We will show that it is also $k$-superseparable and $k$-subseparable. Since the sum of $p$-superseparable (respectively, $p$-subseparable) set functions is also $p$-superseparable (respectively, $p$-subseparable), it suffices to show that our hypothesis is true for an arbitrary $k$-approval utility function of a single agent $i$, $u_i$.

We fix $S \subseteq C$: let $\ell$ denote the number of elements in $S$ which are approved of by $i$. Since, the utilities of the agents are $k$-approval, there are only $(k - \ell)$ elements in $C \setminus S$ for which $u_i(\{x\}) > 0$. For each such an element $x$, we have $\big(v(S \cup \{x\}) - v(S)\big) = \lambda_{\ell+1}$. Thus:

$$\sum_{x \in X} \big(u_i(S \cup \{x\}) - u_i(S)\big) = (k-\ell)\lambda_{\ell+1}.$$

If $\ell \geq 1$, then:

$$\sum_{x \in X} \big(u_i(S \cup \{x\}) - u_i(S)\big) = (k-\ell)\lambda_{\ell+1} \geq 0 = k\lambda_1 - k\lambda_1 \geq \sum_{x \in X} u_i(\{x\}) - ku_i(S).$$

If $\ell = 0$, then:

$$\sum_{x \in X} \big(u_i(S \cup \{x\}) - u_i(S)\big) = k\lambda_1 = \sum_{x \in X} u_i(\{x\}) \geq \sum_{x \in X} u_i(\{x\}) - ku_i(S).$$

Which shows that $u_i$ is $k$-superseparable, and thus that $v$ is $k$-superseparable. Further,

$$\sum_{x \in X} \big(u_i(S \cup \{x\}) - u_i(S)\big) = (k-\ell)\lambda_{\ell+1} \leq (k-\ell)\Big(\sum_j \lambda_j - \sum_{j \leq \ell} \lambda_j\Big)$$
$$= (k-\ell)\big(u_i(X) - u_i(S)\big) \leq k\big(u_i(X) - u_i(S)\big).$$

Which shows that $u_i$ is $k$-subseparable. This completes the proof. □

As the corollary we get that Algorithm 1 and Algorithm 3 are applicable to the problem.

**Corollary 2.** *For each non-increasing OWA vector there exists FPT approximation schemes for the maximization and minimization variants of the problem of selecting $K$ items with $k$-approval utilities for the parameter $(K, k)$.*

There are numerous consequences of Corollary 2. First, Algorithm 1 and Algorithm 3 are applicable to the facility location problem [43]. Further, since the item selection problem is a model for multi-winner election systems (with candidates corresponding to items and voters to agents), Algorithm 1 and Algorithm 3 are applicable to the problem of finding winners under Chamberlin and Courant [35] and Proportional Approval Voting [37] election systems. In all these cases the assumption that the number of approved items is small is realistic. For instance, in case of elections, in some countries, the voting procedure imposes constraints on how many candidates a voter can approve of (for instance, in the Polish parliamentary elections these are only three candidates). Our results can be also extended to cover geometric utilities, that is the case where the set of utilities of each agent has the form $\{d^{m-1}, d^{m-2}, \dots, 1\}$, for some $d > 1$.

### *4.2. Matching and assignment problems*

The algorithm for $p$-superseparable set functions, i.e., Algorithm 1, is also applicable to variants of assignment problems in bipartite graphs. We provide an example by showing how to apply our results to the WEIGHTED-B-$K$-MATCHING problem, which is similar in spirit to the item selection problem from the previous subsection. Here, however, we introduce additional capacity constraints, which says that items cannot be shared among too many agents.

In the WEIGHTED-B-$K$-MATCHING problem we are given a set of vertices $X \cup Y$, a set of edges $E$ (there are no edges neither between the vertices from $X$ nor between the vertices from $Y$), a weight function $w : E \to \mathbb{R}$, and a capacity function $c : X \to \mathbb{Z}$. The goal is to find a subset of edges with the maximal total weight, such that each vertex $x \in X$ belongs to at most $c(x)$ of the selected edges, each vertex $y \in Y$ belongs to at most one of the selected edges, and altogether there are at most $K$ vertices from $X$ which belong to some of the selected edges.

Let us explain the relation between the WEIGHTED-B-$K$-MATCHING problem and the item selection problem from the previous subsection. We observe that vertices from $X$ can be thought of as items and the vertices from $Y$ can be identified with agents. A weight $w$ of an edge $(x, y)$, $x \in X$ and $y \in Y$, corresponds to the utility that the agent $y$ assigns to the item $x$. In the WEIGHTED-B-$K$-MATCHING problem we need to select $K$ items from $X$ and assign each agents to the selected items so that the number of agents assigned to a single selected item does not exceed its capacity.

This matching problem is at least as hard as the item selection problem from the previous subsection for the fixed OWA vector $\lambda = \langle 1, 0, \dots, 0 \rangle$. In particular, it is NP-hard, for many natural parameters it is hard from the point of view of the parameterized complexity theory, and there exists no polynomial-time algorithm that would approximate it with a better ratio than $1 - 1/e$. Yet, our algorithms are also applicable to this problem.

**Lemma 3.** *The* WEIGHTED-B-$K$-MATCHING *problem with the degree of vertices from $Y$ bounded by $p$ can be expressed as the maximization of a non-negative, nondecreasing submodular function which is $p$-superseparable.*

**Proof.** For each set $S \subseteq X$ we define $v(S)$ as the maximum weight of a matching for the graph induced by the set of vertices $S \cup Y$. Such defined $v$ is non-negative and submodular ([3], Theorem 9), and can be evaluated in polynomial time.

First, we show that $v$ is $p$-superseparable. For each $S$, let $\mathcal{M}$ denote some optimal matching for the graph induced by the set of vertices $S \cup Y$. For each $S$, and for each $x \in X$, $\big(v(S) + v(\{x\}) - v(S \cup \{x\})\big)$ is no greater than the weight of such edges $(x', y') \in \mathcal{M}$ that there exists an edge between $x$ and $y'$. This holds because we can construct a feasible matching for the graph induced by $S \cup \{x\} \cup Y$ in the following way. Let us denote the matchings for the graphs induced by $S \cup Y$ and by $\{x\} \cup Y$, as $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively. If we merge $\mathcal{M}_1$ and $\mathcal{M}_2$ and drop each edge $(x', y')$ from $\mathcal{M}_1$ that connects a vertex $y' \in Y$ which is already connected in $\mathcal{M}_2$ (i.e., each edge $(x', y')$ from $\mathcal{M}_1$ such that there exists an edge $(x, y')$ in $\mathcal{M}_2$), we get a feasible matching for the graph induced by $S \cup \{x\} \cup Y$. Consequently, $\sum_{x \in X} \Big(v(S) + v(\{x\}) - v(S \cup \{x\})\Big)$ is no greater than the weight of $\mathcal{M}_1$ times the bound on the degree of vertices from $Y$. That is:

$$\sum_{x \in X} \Big(v(S) + v(\{x\}) - v(S \cup \{x\})\Big) \le p \cdot v(S),$$

which is equivalent to the condition for $p$-superseparability. This proves the thesis. □

**Corollary 3.** *There exists an FPT approximation scheme for the maximization variant of the* WEIGHTED-B-$K$-MATCHING *for the parameter $p$, the bound on the degree of vertices from $Y$.*

Since finding winners under the Monroe election system [44] is computationally equivalent to solving WEIGHTED-B-$K$-MATCHING with specific weights and specific capacities [3], Corollary 3 implies that for winner determination under the Monroe election system with $k$-approval utilities, there exists an FPT approximation scheme for the parameter $(k, K)$.

### *4.3. The* MAXWEIGHTCOVER *problem*

In this subsection we explain that all our algorithms are applicable to MAXWEIGHTCOVER, a generalized variant of the MAXCOVER problem.

Let us recall that in the MAXWEIGHTCOVER problem, we are given a set $N = \{e_1, e_2, \dots e_n\}$ of $n$ elements and a collection $X = \{S_1, \dots, S_m\}$ of $m$ subsets of $N$. Each element $e_i$ has a weight $w_i$. The goal is to find a subcollection $\mathcal{C}$ of $X$ of size at most $K$ that maximizes the total weight of covered elements: $\sum_{i: i \in \bigcup \mathcal{C}} w_i$. Let us also recall that a *frequency* of an element $e_i$ is the number of sets that contain $e_i$. Frequency of elements is a natural parameter considered in the context of approximability of covering problems [45]. To the best of our knowledge, for polynomial-time algorithms, there exists no better guarantee for the MAXCOVER problem with bounded frequencies of elements than $(1 - 1/e)$. This is specifically interesting, since such better approximation algorithms exists for the very similar problem SETCOVER [45].

In Example 1 we have shown that the MAXWEIGHTCOVER problem with the frequency of elements upper-bounded by $p$ can be expressed as the maximization of a non-negative, non-decreasing submodular function which is (i) $p$-superseparable, and (ii) $p$-subseparable, and that the MAXWEIGHTCOVER problem with the frequency of elements lower-bounded by $p$ can be expressed as the maximization of a non-negative, non-decreasing submodular function which is rev-$p$-subseparable. As a corollary of these observations we get that all our algorithms can be applied to certain natural variants of the MAXWEIGHTCOVER problem.

**Corollary 4.** *There exists an FPT approximation scheme both for the maximization and for the minimization variant of the* MAXWEIGHTCOVER *for the parameter $(K, p)$, where p is the upper-bound on the frequency of the elements.*

As an implication from Corollary 1, we also get that:

**Corollary 5.** *Let $\gamma$ be a constant and let $\gamma$-*MAXWEIGHTCOVER *be the variant of the* MAXWEIGHTCOVER *problem with additional assumption that each element is covered by at least $\gamma$ fraction of the sets from $X$. There exists a PTAS for $\gamma$-*MAXWEIGHTCOVER.

Corollary 4 and Corollary 5 extend the recent results for the MAXCOVER problem [16]. Further, interestingly Theorem 3 says that there exists a randomized FPT approximation scheme for the minimization-or-maximization variant of the MAXWEIGHTCOVER problem, for the parameter $(K, p_{av})$, where $p_{av}$ is an average frequency of an element.

## 5. Conclusions

In this paper we have considered approximation algorithms for the problem of maximizing submodular set functions. Since it is known that the greedy algorithm achieves approximation ratio of $(1 - 1/e)$ and that no polynomial-time algorithm can approximate the problem better, we have initiated the study on algorithms for the problem that run in super-polynomial time. In our study we have followed the approach of parameterized complexity theory. We have identified three new properties of set functions, called $p$-separability properties, and we have considered the parameter $p$ from the definition of $p$-separability. For $p$ combined with $K$, the size of the solution, we have shown that the problem can be arbitrarily well approximated by algorithms that run in FPT time. We have justified our choice of the parameters by providing numerous examples of computational problems which can be expressed as maximization of submodular $p$-separable set functions, and by arguing that in many real-life problems the value of $p$ is small.

We have exposed the difference between approximating the maximization and minimization variants of the problem and we have provided a new weaker definition of approximability: approximation of minimization-or-maximization variant of the problem. We have shown that it is possible to approximate our problem arbitrarily well using algorithms that run in FPT time for the more general parameter $\left(K, \frac{\sum_{x \in X} v(\{x\})}{v(X)}\right)$.

There are many natural ways in which this research can be extended. We believe that one of the promising approaches is to consider the problem with additional constraints, such as knapsack constraints or matroid constraints.

## Acknowledgments

The author thanks Piotr Faliszewski, the anonymous reviewers of WINE-2016 and of Information & Computation for their helpful comments.

This research has been supported by Europe Research Grant ERC-StG 639945 and by a Humboldt Research Fellowship for Postdoctoral Researchers.

## References

[1] T. Lu, C. Boutilier, Budgeted social choice: from consensus to personalized decision making, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-2011), 2011, pp. 280–286.
[2] P. Skowron, P. Faliszewski, J. Lang, Finding a collective set of items: from proportional multirepresentation to group recommendation, Artif. Intell. 241 (2016) 191–216.
[3] P. Skowron, P. Faliszewski, A. Slinko, Achieving fully proportional representation: approximability result, Artif. Intell. 222 (2015) 67–103.
[4] S. Jegelka, J. Bilmes, Submodularity beyond submodular energies: coupling edges in graph cuts, in: Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2011), 2011, pp. 1897–1904.
[5] G. Kim, E. Xing, L. Fei-Fei, T. Kanade, Distributed cosegmentation via submodular optimization on anisotropic diffusion, in: Proceedings of IEEE International Conference on Computer Vision (ICCV-2011), 2011, pp. 169–176.
[6] s. Jegelka, F. Bach, S. Sra, Reflection methods for user-friendly submodular optimization, in: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (Eds.), Proceedings of Advances in Neural Information Processing Systems 26 (NIPS-2013), Curran Associates, Inc., 2013, pp. 1313–1321.
[7] Y. Yue, C. Guestrin, Linear submodular bandits and their application to diversified retrieval, in: Proceedings of Advances in Neural Information Processing Systems 24 (NIPS-2011), 2011, pp. 2483–2491.
[8] H. Lin, J. Bilmes, Multi-document summarization via budgeted maximization of submodular functions, in: Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HTL-2010), Association for Computational Linguistics, Stroudsburg, PA, USA, 2010, pp. 912–920.
[9] A. Krause, C. Guestrin, Near-optimal value of information in graphical models, in: Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI-2005), 2005.
[10] A. Krause, A. Singh, C. Guestrin, Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies, J. Mach. Learn. Res. 9 (2008) 235–284.
[11] M. Narasimhan, J. Nebojsa, B. Jeff, Q-clustering, in: Y. Weiss, B. Schölkopf, J. Platt (Eds.), Proceedings of Advances in Neural Information Processing Systems 18 (NIPS-2010), MIT Press, 2006, pp. 979–986.
[12] H. Lin, J. Bilmes, How to select a good training-data subset for transcription: submodular active selection for sequences, in: Proceedings of 10th Annual Conference of the International Speech Communication Association (INTERSPEECH-2009), 2009, pp. 2859–2862.
[13] F. Bach, Structured sparsity-inducing norms through submodular functions, in: Proceedings of Advances in Neural Information Processing Systems 23 (NIPS-2010), 2010, pp. 118–126.
[14] A. Das, D. Kempe, Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 1057–1064.
[15] V. Chvatal, A greedy heuristic for the set-covering problem, Math. Oper. Res. 4 (3) (1979) 233–235.
[16] P. Skowron, P. Faliszewski, Fully proportional representation with approval ballots: approximating the MMaxCover problem with bounded frequencies in FPT time, in: Proceedings of the 29th Conference on Artificial Intelligence (AAAI-2015), 2015, pp. 2124–2130.
[17] S. Iwata, L. Fleischer, S. Fujishige, A combinatorial strongly polynomial algorithm for minimizing submodular functions, J. ACM 48 (4) (2001) 761–777.
[18] G. Nemhauser, L. Wolsey, M. Fisher, An analysis of approximations for maximizing submodular set functions, Math. Program. 14 (1) (1978) 265–294.
[19] R. Kumar, B. Moseley, S. Vassilvitskii, A. Vattani, Fast greedy algorithms in mapreduce and streaming, in: Proceedings of the Twenty-Fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA-2013), 2013, pp. 1–10.
[20] M. Streeter, D. Golovin, An online algorithm for maximizing submodular functions, in: Proceedings of Advances in Neural Information Processing Systems 21 (NIPS-2008), 2008, pp. 1577–1584.
[21] U. Feige, V. Mirrokni, J. Vondrák, Maximizing non-monotone submodular functions, SIAM J. Comput. 40 (4) (2011) 1133–1153.
[22] G. Calinescu, C. Chekuri, M. Pál, J. Vondrák, Maximizing a submodular set function subject to a matroid constraint (extended abstract), in: Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO-2007), 2007, pp. 182–196.
[23] M. Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, Oper. Res. Lett. 32 (1) (2004) 41–43.
[24] J. Lee, V. Mirrokni, V. Nagarajan, M. Sviridenko, Non-monotone submodular maximization under matroid and knapsack constraints, in: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing (STOC-2009), 2009, pp. 323–332.
[25] J. Vondrák, C. Chekuri, R. Zenklusen, Submodular function maximization via the multilinear relaxation and contention resolution schemes, in: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing (STOC-2011), 2011, pp. 783–792.
[26] A. Krause, D. Golovin, Submodular Function Maximization, Tech. rep., 2012.
[27] M. Conforti, G. Cornuéjols, Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the rado-edmonds theorem, Discrete Appl. Math. 7 (3) (1984) 251–274.
[28] M. Sviridenko, J. Vondrák, J. Ward, Optimal approximation for submodular and supermodular optimization with bounded curvature, in: Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms (SODA-2015), 2015, pp. 1134–1148.
[29] U. Feige, A threshold of $\ln n$ for approximating set cover, J. ACM 45 (4) (1998) 634–652.
[30] R. Downey, M. Fellows, Parameterized Complexity, Springer-Verlag, 1999.
[31] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.
[32] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer-Verlag, 2006.

[33] M. Cygan, F. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015.
[34] F. Croce, V. Paschos, Efficient algorithms for the max *k*-vertex cover problem, J. Comb. Optim. 28 (3) (2014) 674–691.
[35] B. Chamberlin, P. Courant, Representative deliberations and representative decisions: proportional representation and the borda rule, Am. Polit. Sci. Rev. 77 (3) (1983) 718–733.
[36] F. Farahani, M. Hekmatfar (Eds.), Facility Location: Concepts, Models, and Case Studies, Springer, 2009.
[37] D. Kilgour, Approval balloting for multi-winner elections, in: J. Laslier, R. Sanver (Eds.), Handbook on Approval Voting, Springer, 2010, pp. 105–124.
[38] D. Baumeister, G. Erdélyi, E. Hemaspaandra, L. Hemaspaandra, J. Rothe, Computational aspects of approval voting, in: J. Laslier, R. Sanver (Eds.), Handbook of Approval Voting, Springer, 2010, pp. 199–251.
[39] H. Aziz, M. Brill, V. Conitzer, E. Elkind, R. Freeman, T. Walsh, Justified representation in approval-based committee voting, Soc. Choice Welf. 48 (2) (2017) 461–485.
[40] H. Aziz, S. Gaspers, J. Gudmundsson, S. Mackenzie, N. Mattei, T. Walsh, Computational aspects of multi-winner approval voting, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2015), 2015, pp. 107–115.
[41] A. Procaccia, J. Rosenschein, A. Zohar, On the complexity of achieving proportional representation, Soc. Choice Welf. 30 (3) (2008) 353–362.
[42] N. Betzler, A. Slinko, J. Uhlmann, On the computation of fully proportional representation, J. Artif. Intell. Res. 47 (2013) 475–519.
[43] K. Jain, V. Vazirani, Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation, J. ACM 48 (2) (2001) 274–296.
[44] B. Monroe, Fully proportional representation, Am. Polit. Sci. Rev. 89 (4) (1995) 925–940.
[45] V. Vazirani, Approximation Algorithms, Springer-Verlag New York, Inc., New York, NY, USA, 2001.