Submodular Maximization with Cardinality Constraints

Niv Buchbinder* Moran Feldman† Joseph (Seffi) Naor‡ Roy Schwartz§

Abstract

We consider the problem of maximizing a (non-monotone) submodular function subject to a cardinality constraint. In addition to capturing well-known combinatorial optimization problems, *e.g.*, Max-k-Coverage and Max-Bisection, this problem has applications in other more practical settings such as natural language processing, information retrieval, and machine learning. In this work we present improved approximations for two variants of the cardinality constraint for non-monotone functions. When at most k elements can be chosen, we improve the current best $1/e - o(1)$ approximation to a factor that is in the range $[1/e + 0.004, 1/2]$, achieving a *tight* approximation of $1/2 - o(1)$ for $k = n/2$ and breaking the $1/e$ barrier for all values of k. When exactly k elements must be chosen, our algorithms improve the current best $1/4 - o(1)$ approximation to a factor that is in the range $[0.356, 1/2]$, again achieving a *tight* approximation of $1/2 - o(1)$ for $k = n/2$. Additionally, some of the algorithms we provide are very fast with time complexities of $O(nk)$, as opposed to previous known algorithms which are continuous in nature, and thus, too slow for applications in the practical settings mentioned above.

Our algorithms are based on two new techniques. First, we present a simple randomized greedy approach where in each step a random element is chosen from a set of "reasonably good" elements. This approach might be considered a natural substitute for the greedy algorithm of Nemhauser, Wolsey and Fisher [45], as it retains the same tight guarantee of $1 - 1/e$ for monotone objectives and the same time complexity of $O(nk)$, while giving an approximation of $1/e$ for general non-monotone objectives (while the greedy algorithm of Nemhauser et. al. fails to provide any constant guarantee). Second, we extend the double greedy technique, which achieves a tight $1/2$ approximation for unconstrained submodular maximization, to the continuous setting. This allows us to manipulate the natural rates by which elements change, thus bounding the total number of elements chosen.

1 Introduction

A set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is submodular if for every $A, B \in \mathcal{N}$: $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$. Such functions are ubiquitous in various disciplines, including combinatorics, optimization, economics, information theory, operations research, algorithmic game theory, and machine learning. Many well known functions, such as cut functions of graphs and hypergraphs, rank functions of matroids, entropy, mutual information, coverage functions, and budget additive functions, are submodular. An equivalent definition of submodularity, which is perhaps more intuitive, is that of diminishing returns: $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$ for every $A \subseteq B \subseteq \mathcal{N}$ and $u \notin B$. The concept of diminishing returns is widely used in economics, and thus, it should come as no surprise that utility functions in economics are often submodular.

Submodular maximization problems capture well known combinatorial optimization problems such as: Max-Cut [26, 29, 31, 33, 50], Max-DiCut [16, 26, 27], Generalized Assignment [10, 11, 19, 23], and Max-Facility-Location [1, 12, 13]. Additionally, one can find submodular maximization problems in many other settings. In machine learning, maximization of submodular functions has been used for document summarization [41, 42], sensor placement [36, 38, 35], and information gathering [37] (consult the references therein for additional applications in AI). In algorithmic game theory, calculating market expansion [14] and computing core values of certain types of games [48], are two examples where the problem can be reduced to submodular maximization. Furthermore, in social networks, viral marketing and influence [28, 32] are both calculated by submodular maximization. Image segmentation [6, 30] and speeding up satisfiability solvers [49] are additional examples in which submodular maximization is useful.

In this paper we consider the problem of maximizing a general non-negative submodular function, no necessarily monotone, subject to a cardinality constraint. Given a cardinality parameter k, the goal is to find a subset $S \subseteq \mathcal{N}$ maximizing $f(S)$ such that $|S| \leq k$. We also consider the case in which the goal is to choose *exactly* k elements from $\mathcal{N}$, *i.e.*, $|S| = k$. This problem captures several well known optimization problems including Max-k-Coverage [15, 34], Max-Bisection [5, 24], and several variants of Max-Cut in which the cut size is prespecified: undirected graphs [3, 17],

*Statistics and Operations Research Dept., Tel Aviv University, Israel. e-mail: niv.buchbinder@gmail.com. Research supported in part by ISF grant 954/11 and by BSF grant 2010426.

†School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland. e-mail: moran.feldman@epfl.ch.

‡Computer Science Dept., Technion, Haifa, Israel. e-mail: naor@cs.technion.ac.il. Research supported in part by the Google Inter-university center for Electronic Markets, by ISF grant 954/11, and by BSF grant 2010426.

§Microsoft Research, Redmond, WA. e-mail: roysch@microsoft.com

directed graphs [2] and hypergraphs [4].

Additionally, the problem of maximizing a non-monotone and non-negative submodular function subject to a cardinality constraint has applications in more practical settings. For example, consider the problem of document summarization [41, 42] whose applications span different fields such as natural language processing and information retrieval. In this problem the goal is to extract a small number of textual units from given text documents as to form a short summary. The quality of the summary is a non-monotone and non-negative submodular function as similarities between selected textual units are deducted from the total benefit these textual units contribute to the summary (the reader is referred to [41, 42] for more details). The cardinality constraint is due to real-world restrictions that limit the size of the summary. It is important to note that for the above text summarization problem in particular, and for many other applications in general, *fast* algorithms are necessary since the size of the ground set $\mathcal{N}$ is very large and even quadratic time complexity in the size of $\mathcal{N}$ is considered impractical.

The classical result of [45] states that the simple discrete greedy algorithm provides an approximation of $1 - 1/e$ for maximizing a *monotone*[1] submodular function where at most k elements can be chosen. This result is known to be tight [44], even in the case where the objective function is a coverage function [15]. However, when one considers submodular objectives which are not monotone, less is known. An approximation of 0.309 was given by [51], which was later improved to 0.325 [25] using a simulated annealing technique. Extending the continuous greedy algorithm of [9] from the case of monotone submodular objectives to the general case of non-negative submodular objectives which are not necessarily monotone, [20] obtained an improved approximation of $1/e - o(1)$. When one considers the variant of the cardinality constraint where *exactly* k elements must be chosen, an approximation of $1/4 - o(1)$ [51] is known via a fractional local search (improving upon the $1/6 - o(1)$ of [40]). For both variants of the cardinality constraint, [25] presented a hardness of 0.491 when $k = o(n)$, while a slightly weaker hardness of $1/2$ readily follows from [51] for the case of $k = n/2$. We note that the cardinality constraint is a well-studied special case of the more general matroid constraint, where one needs to maximize the objective given that the output is an independent set (or a base) of a matroid.

1.1 Our Results

Improved Approximations Subject to a Cardinality Constraint: We present improved approximations for maximizing a general non-negative submodular function subject to a cardinality constraint. Both variants of the constraint are considered. The results are summarized in theorems 1.1 and 1.2, and appear in Table 1.

For the variant in which at most k elements can be chosen (Theorem 1.1) our improved approximation guarantee is tight for $k = n/2$, achieving a guarantee of $1/2 - o(1)$, and deteriorates as k decreases. However, this guarantee never falls below an absolute constant of $1/e + 0.004$. Our algorithm, to the best of our knowledge, is the first to break the $1/e$ barrier for any value of k. The constant $1/e + 0.004$ should be considered as a "proof of concept" that the natural $1/e - o(1)$ guarantee of [20] is not the right answer.

For the variant in which exactly k elements must be chosen (Theorem 1.2) our improved approximation guarantee is tight for $k = n/2$, achieving a guarantee of $1/2 - o(1)$, and deteriorates as k decreases until it reaches a guarantee of 0.356 (when $k \approx 0.084n$). At this point, the guarantee begins improving again as k continues to decrease, approaching $1/e$ when $k = o(n)$.

THEOREM 1.1. *There exists an efficient algorithm that given a non-negative submodular function f and a cardinality parameter $k \leq n/2$, achieves an approximation of*

$$\max\left\{1/e + 0.004, \left(1 + \frac{n}{2\sqrt{(n-k)k}}\right)^{-1} - o(1)\right\}$$

for the problem: $\max\{f(S) : |S| \leq k\}$. If $k > n/2$ then the approximation ratio is $1/2 - o(1)$.

THEOREM 1.2. *There exists an efficient algorithm that given a non-negative submodular function f and a cardinality parameter $k \leq n/2$, achieves an approximation of*

$$\max\left\{\frac{1 - k/en}{e} - \varepsilon, \left(1 + \frac{n}{2\sqrt{(n-k)k}}\right)^{-1} - o(1)\right\} \geq 0.356$$

for the problem: $\max\{f(S) : |S| = k\}$ (ε is an arbitrarily small constant). If $k > n/2$ then the above approximation still applies with the cardinality parameter k replaced with $n - k$.

We note that in both the above two theorems, the approximation guarantee is obtained by taking the best out of two algorithms.

Fast Algorithms: We present fast randomized combinatorial algorithms with provable guarantees for the problems of maximizing a general non-negative submodular function subject to a cardinality constraint and a general matroid independence constraint. The results are summarized in Theorems 1.3, 1.4, 1.5 and 1.6, and appear in Table 2.

For the cardinality constraint variant in which at most k elements can be chosen (Theorem 1.3) we present a fast randomized combinatorial algorithm that retains the current best known approximation of $1/e$ [20] (and even avoid the $o(1)$ loss), while improving the time complexity from $O(n^2k^6)$[2]

[1] A set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is monotone if for every $A \subseteq B \subseteq \mathcal{N}$, $f(A) \leq f(B)$.

[2] Some of the time complexities we present for previous works are based on the improved time complexity analysis of [21].

Constraint	k	This Work	Previous Work	Hardness
$\|S\| \leq k$	k	$\max\left\{1/e + 0.004, \left(1 + \frac{n}{2\sqrt{(n-k)k}}\right)^{-1} - o(1)\right\}$	$1/e - o(1)$ [20]	–
	$n/2$	$1/2 - o(1)$		$1/2$ [51]
	$o(n)$	$1/e + 0.004$		0.491 [25]
$\|S\| = k$	k	$\max\left\{\frac{1-k/en}{e} - \varepsilon, \left(1 + \frac{n}{2\sqrt{(n-k)k}}\right)^{-1} - o(1)\right\} \geq 0.356$	$1/4 - o(1)$ [51]	–
	$n/2$	$1/2 - o(1)$		$1/2$ [51]
	$o(n)$	$1/e - \varepsilon$		0.491 [25]

Table 1: Improved approximations for submodular maximization with a cardinality constraint.

to $O(nk)$. For the cardinality constraint variant in which exactly k elements must be chosen (Theorem 1.4) we note that one of the two algorithms that comprises the guarantee in Theorem 1.2 is in fact fast and runs in time $O(nk)$. Hence, for this variant of the problem, we do not only improve the approximation guarantee from $1/4 - o(1)$ [51] to $(1 - k/en)/e - \varepsilon \geq 0.3$ (assuming $k \leq n/2$), but also improve the time complexity to $O(nk)$.

For the matroid independence constraint (Theorem 1.5) we present a fast randomized combinatorial algorithm that runs in time $O(Tk)$ and achieves a slightly worse approximation of $1/4$ than the current best known $1/e - o(1)$ [20] (T is the time needed to compute a maximum weight independent set in the matroid,[3] and it satisfies $T = O(n \log n)$). Notice that the time complexity of [20] is $O(n^2k^6)$. Additionally, we present an algorithm with a better approximation of $(1+e^{-2})/4 - \varepsilon > 0.283$ (Theorem 1.6) and a possibly worse running time of $O(Tk + Mk)$, where M is the time needed to compute a perfect matching in a bi-partite graph which has k vertices on each side. Note that $\mathbb{E}[M] = O(k^\omega)$ where ω is the exponent of matrix multiplication.

THEOREM 1.3. *There exists an algorithm that given a non-negative submodular function f and a cardinality parameter k, achieves an approximation of $1/e$ for the problem: $\max\{f(S) : |S| \leq k\}$ and runs in $O(nk)$ time.*

THEOREM 1.4. *There exists an algorithm that given a non-negative submodular function f and a cardinality parameter k, $k \leq n/2$, achieves an approximation of $(1-k/en)/e - \varepsilon$ ($\varepsilon > 0$ is an arbitrarily small constant) for the problem: $\max\{f(S) : |S| = k\}$ and runs in $O(nk)$ time. If $k > n/2$ then the above approximation still applies with the cardinality parameter k replaced with $n - k$.*

[3] Our algorithm actually needs to compute a maximum weight independent set in matroids that are constructed from the input matroid $\mathcal{M}$ by deletion and contraction operations. The problem of computing a maximum weight independent set in any such matroid can be easily reduced to the same problem in $\mathcal{M}$.

THEOREM 1.5. *There exists an algorithm that given a non-negative submodular function f and a matroid $\mathcal{M} = (\mathcal{I}, \mathcal{N})$, achieves an approximation of $1/4$ for the problem: $\max\{f(S) : S \in \mathcal{I}\}$ and runs in $O(Tk)$ time. T is the time required to compute a maximum weight independent set in $\mathcal{M}$.*

THEOREM 1.6. *There exists an algorithm that given a non-negative submodular function f and a matroid $\mathcal{M} = (\mathcal{I}, \mathcal{N})$, achieves an approximation of $\frac{1+e^{-2}}{4} - \varepsilon > 0.283$ ($\varepsilon > 0$ is an arbitrarily small constant) for the problem: $\max\{f(S) : S \in \mathcal{I}\}$ and runs in $O(Tk + Mk)$ time, where T is the time required to compute a maximum weight independent set in $\mathcal{M}$, and M is the time required to compute a perfect matching in a bipartite graph having k elements on each side.*

1.2 Techniques All the algorithms we present in this work are based on one (or both) of the following techniques.

Discrete Random Greedy: It is well known that the greedy approach provides a tight guarantee for maximizing a *monotone* submodular function subject to a cardinality constraint (where at most k elements can be chosen) [45]. In order to obtain results for non-monotone objectives, a sequence of works was needed, starting with the celebrated continuous greedy algorithm of [9] which extended [45]'s result to a general matroid constraint by using continuous techniques. Unfortunately, this continuous extension could handle only monotone objectives, and was later improved by [20] so non-monotone objectives could be handled with a general matroid constraint. It is important to note that the latter improvement is somewhat counter intuitive, as it is known that the greedy approach fails for non-monotone objectives in the discrete setting even for a cardinality constraint.

In this work we present a new and different approach in which one can extend the original discrete greedy algorithm of [45] to handle non-monotone objectives. Instead of operating continuously, one simply adds randomization. Specifically, the algorithm chooses in each step a random element

Constraint	This Work		Previous Work	
	Guarantee	Time	Guarantee	Time
$\lvert S\rvert \leq k$	$1/e$	$O(nk)$	$1/e - o(1)$ [20]	$O(n^2k^6)$
$\lvert S\rvert = k$	$\frac{1-k/en}{e} - \varepsilon > 0.3$	$O(nk)$	$1/4 - o(1)$ [51]	$poly(n,k)$
matroid independent set	$1/4$	$O(Tk)$	$1/e - o(1)$ [20]	$O(n^2k^6)$
	$\frac{1+e^{-2}}{4} - \varepsilon > 0.283$	$O(Tk + Mk)$		

Table 2: Fast algorithms. $T = O(n \log n)$ is the time required to compute a maximum weight independent set in the matroid. M is the time required to compute a perfect matching in a bipartite graph having k vertices on each side (note that $\mathbb{E}[M] = O(k^\omega)$ where ω is the exponent of matrix multiplication).

among a set of "reasonably" good elements. This approach enables us to obtain fast and simple combinatorial algorithms for maximizing a non-monotone submodular objective subject to a cardinality constraint and a matroid independence constraint.

To demonstrate the power of this approach, one only needs to consider the fast algorithm that achieves a guarantee of $1/e$ (Theorem 1.3) for maximizing a general non-negative submodular function subject to choosing at most k elements. Even though it is not stated in the theorem, as it is not necessary for the non-monotone case, the exact same algorithm also achieves an approximation of $1 - 1/e$ for maximizing a monotone submodular function given the same constraint. Hence, both the approximation guarantee and the time complexity of the algorithm are exactly the same as those of the original tight algorithm of [45]. This simple randomized greedy algorithm might be considered as a natural substitute for the greedy algorithm of [45] as it works for both monotone and non-monotone objectives simultaneously.

Continuous Double Greedy: The double greedy approach was introduced by [8] in the context of unconstrained submodular maximization. This approach maintains two evolving sets, and in every step both sets agree whether to include (or exclude) an element in the solution, up to the point where all elements are examined and both sets are identical. Despite the simplicity of this approach and the fact that it produces a tight approximation in the unconstrained case, no guarantee is provided on the number of elements in the output.

To overcome this difficulty, one must change the rule by which the two sets evolve. To this end, we present the continuous counterpart of the double greedy approach, in which two *fractional* sets are maintained. As the evolution process of these fractional sets is continuous, each element is associated with a fractional value indicating how likely it is to be included or excluded from the output. These values are changed in a continuous fashion for all elements simultaneously, according to rates defined by the algorithm. Unlike the discrete double greedy approach which makes a single irrevocable decision for each element one at a time, our continuous algorithm slowly changes the values of elements simultaneously. This careful process enables us to control the total (fraction) of elements in the output while maintaining a guarantee on the value of the solution.

1.3 Related Work The literature on submodular maximization problems is very large, and therefore, we mention below only a few of the most relevant works. For maximizing monotone submodular objectives subject to a general matroid constraint, [22] proved that the discrete greedy algorithm is a $1/2$ approximation. This was later improved to a tight $1 - 1/e$ approximation by [9], who presented the celebrated continuous greedy algorithm. A combinatorial local-search approach achieving the same tight guarantee is given in [21].

Regarding general (not necessarily monotone) submodular objectives and a general matroid constraint, [51] provided an approximation of 0.309. Using simulated annealing techniques this was improved to 0.325 [25], and shortly later was further pushed to $1/e - o(1)$ by [20] using an extension of the continuous greedy algorithm.

Paper Organization: Section 2 contains some technical preliminaries, including as to why one can assume $k \leq n/2$. Section 3 contains the two core algorithms, which form the basis of our new techniques and exemplify how they can be used. The first of the two is a fast random greedy algorithm and it can be found in Section 3.1, the second is a continuous double greedy algorithm and can be found in Section 3.2. All other algorithms are based on these two core algorithms (either as a black box or they use techniques and analysis ideas), and can be found in Section 4.

2 Preliminaries

For every set S and an element u, we denote the union $S \cup \{u\}$ by $S + u$, and the expression $S \setminus \{u\}$ by $S - u$. Given a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}$, the marginal contribution of u to S is denoted by $f_u(S) = f(S + u) - f(S)$. For a vector $x \in [0,1]^{\mathcal{N}}$, we define the random subset $\mathtt{R}(x) \subseteq \mathcal{N}$ which contains each element

$u \in \mathcal{N}$ independently with probability x_u. The multilinear extension of f is a function $F : [0,1]^{\mathcal{N}} \to \mathbb{R}$, whose value at a vector $x \in [0,1]^{\mathcal{N}}$ is the expected value of f over $\mathtt{R}(X)$. Formally, for every $x \in [0,1]^{\mathcal{N}}$, $F(x) \triangleq \mathbb{E}[\mathtt{R}(x)] = \sum_{S \subseteq \mathcal{N}} f(S) \prod_{u \in S} x_u \prod_{u \notin S} (1 - x_u)$.

We look for algorithms that are polynomial in n, the size of $\mathcal{N}$. However, the explicit representation of a submodular function might be exponential in the size of its ground set. The standard way to bypass this difficulty is to assume access to the function via an oracle. For a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}$, given a set $S \subseteq \mathcal{N}$, the oracle returns the value of $f(S)$.[4] All the algorithms we describe access submodular functions via such oracles.

To simplify the exposition of our algorithms, we assume one of the following reductions was applied (depending on the problem at hand).

REDUCTION 1. *For the problem of* $\max\{f(S) : |S| \leq k\}$*, we may assume* $2k \leq n$*, and that there is a set* $D \subseteq \mathcal{N}$ *of* $2k$ *"dummy" elements whose marginal contribution to any set is* 0*. More formally, for every set* $S \subseteq \mathcal{N}$*,* $f(S) = f(S \setminus D)$*.*

Proof. If this is not the case, one can add $2k$ such dummy elements to the ground set, and and remove them from the output of the algorithm, effecting neither OPT, nor the value of the algorithm's output.

REDUCTION 2. *For the problem of* $\max\{f(S) : |S| = k\}$*, we may assume* $2k \leq n$*.*

Proof. Follows immediately from the proof of Corollary 5.3 in [40]. The idea is that if this is not the case, then let $\bar{k} = n - k$, and $\bar{f}(S) = \mathcal{N} \setminus S$. It can be easily checked that $2\bar{k} \leq n$ and that the problem $\max\{\bar{f}(S) : |S| = \bar{k}\}$ is equivalent to the original problem.

We also assume the following reduction was applied.

REDUCTION 3. *For a cardinality constraint with parameter* k*, we may assume* k *is larger than an arbitrarily large constant at the following cost:*

- *For our non-fast algorithms, a low order term loss in the approximation ratio and a polynomial increase in the time complexity.*
- *For our fast algorithms (which evaluate* f *only on sets of size at most* k*), a multiplicative constant increase in the time complexity.*

This reduction preserves the ratio k/n*.*

[4] Such an oracle is called *value oracle*. Other, stronger, oracle types for submodular functions are also considered in the literature, but value oracles are probably the most widely used.

Proof. Assume k is smaller than an arbitrary universal constant c, and let $c' = \lceil c/k \rceil$ and $[c'] = \{1, 2, \ldots, c'\}$. We replace $\mathcal{N}$ with the ground set $\mathcal{N}' = \mathcal{N} \times [c']$, the parameter k with $c'k$ and f with the function $f' : \mathcal{N}' \to \mathbb{R}$ defined as $f'(S) = F(x(S))$, where F is the multilinear extension of f and $x_u(S) = |\{(u,i) \in S : u \in \mathcal{N}\}|/c'$ for every element $u \in \mathcal{N}$. It is easy to see that f' is submodular (see [51] for a formal proof of this) and can be evaluated to an arbitrary accuracy in polynomial time (see [9] for details). Moreover, there is an approximation ratio preserving reduction from the original problem to the new one, which results in a low order term reduction in the approximation ratio of our algorithms using this reduction.

Our fast algorithms evaluate f' only on sets of size at most $c'k = O(c)$, and therefore, for these algorithms it is possible to evaluate f' *exactly* in constant time.

Remark: The non-fast algorithms can in fact resort to exhaustive search when $k \leq c$, which requires only a polynomial time of $O(n^c)$ when c is a constant. This method does not induce any loss in the approximation ratio of the algorithms, but exhibits a worse dependence on c.

We make use of the following known lemma.

LEMMA 2.1. (LEMMA 2.2 OF [18]) *Let* $f : 2^{\mathcal{N}} \to \mathbb{R}$ *be submodular. Denote by* $A(p)$ *a random subset of* A *where each element appears with probability* p *(not necessarily independently). Then* $E[f(A(p))] \geq (1-p)f(\varnothing) + p \cdot f(A)$*.*

We also need the following close variant of the above lemma.

LEMMA 2.2. *Let* $f : 2^{\mathcal{N}} \to \mathbb{R}$ *be submodular. Denote by* $A(p)$ *a random subset of* A *where each element appears with probability at most* p *(not necessarily independently). Then,* $E[f(A(p))] \geq (1-p)f(\varnothing)$*.*

Proof. Sort the elements of A in a non-increasing order of probability to be in $A(p)$: $u_1, u_2, \ldots, u_{|A|}$. In other words, for every pair $1 \leq i \leq j \leq |A|$, $\Pr[u_i \in A(p)] \geq \Pr[u_j \in A(p)]$. Denote by X_i an indicator for the event that $u_i \in A(p)$, by p_i the probability $\Pr[u_i \in A(p)]$ and by A_i the set $\{u_1, u_2, \ldots, u_i\}$. Then:

$$\begin{aligned}
\mathbb{E}[f(A(p))] &= \mathbb{E}\left[f(\varnothing) + \sum_{i=1}^{|A|} X_i \cdot f_{u_i}(A_{i-1} \cap A(p))\right] \\
&\geq \mathbb{E}\left[f(\varnothing) + \sum_{i=1}^{|A|} X_i \cdot f_{u_i}(A_{i-1})\right] \\
&= f(\varnothing) + \sum_{i=1}^{|A|} \mathbb{E}\left[X_i\right] \cdot f_{u_i}(A_{i-1}) \\
&= f(\varnothing) + \sum_{i=1}^{|A|} p_i \cdot f_{u_i}(A_{i-1})
\end{aligned}$$

$$= (1-p_1)\cdot f(\varnothing) + \sum_{i=1}^{|A|-1}[p_{i-1}-p_i]f(A_i) + p_{|A|}\cdot f(A)$$
$$\geq (1-p)\cdot f(\varnothing) \ ,$$

where there first inequality follows from submodularity, and the second one from the order we chose for the elements of A, which guarantees $p \geq p_1 \geq p_2 \geq \ldots \geq p_{|A|}$.

3 Core Algorithms

3.1 The Discrete Random Greedy Algorithm In this section we present the fast algorithm for the problem $\max\{f(S) : |S| \leq k\}$ whose existence is guaranteed by Theorem 1.3. This is the first of the two core algorithms and it presents in the simplest way our new approach of using randomization instead of continuous techniques. As already mentioned, this simple algorithm might be considered a natural substitute for the classical algorithm of Nemhauser et. al. [45], as it retains the same tight guarantee of $1 - 1/e$ for monotone objectives and the same time complexity of $O(nk)$, while giving an approximation of $1/e$ for general non-monotone objectives.

Consider algorithm Random Greedy (Algorithm 1). Observe that the output of Random Greedy might contain less than k elements due to our assumption that Reduction 1 was applied to the problem.

Let A_i be an event fixing all the random decisions of the algorithm up to iteration i (including), and let $\mathcal{A}_i$ be the set of all possible A_i events. As a warmup, let us analyze Random Greedy in the case when f is monotone.

THEOREM 3.1. *The approximation ratio of Random Greedy (Algorithm 1) is $1 - e^{-1}$ when f is monotone.*

Proof. Fix $1 \leq i \leq k$ and an event $A_{i-1} \in \mathcal{A}_{i-1}$. All the probabilities, expectations and random quantities in the first part of this proof are implicitly conditioned on A_{i-1}. Consider a set M_i' containing the elements of $OPT \setminus S_{i-1}$ plus enough dummy elements to make the size of M_i' exactly k. Observe that:

$$\mathbb{E}[f_{u_i}(S_{i-1})] = k^{-1}\cdot\sum_{u\in M_i} f_u(S_{i-1})$$
$$\geq k^{-1}\cdot\sum_{u\in M_i'} f_u(S_{i-1}) = k^{-1}\cdot\sum_{u\in OPT\setminus S_{i-1}} f_u(S_{i-1})$$
$$\geq \frac{f(OPT\cup S_{i-1})-f(S_{i-1})}{k} \geq \frac{f(OPT)-f(S_{i-1})}{k} \ ,$$

where the first inequality follows from the definition of M_i, the second from the submodularity of f and the third from the monotonicity of f. Unfixing the event A_{i-1}, and taking an expectation over all possible such events, we get:

$$\mathbb{E}[f_{u_i}(S_{i-1})] \geq \frac{f(OPT)-\mathbb{E}[f(S_{i-1})]}{k} \ .$$

Rearranging yields:

$$f(OPT)-\mathbb{E}[f(S_i)] \leq \left(1-\frac{1}{k}\right)\cdot[f(OPT)-\mathbb{E}[f(S_{i-1})]] \ ,$$

which implies:

$$f(OPT)-\mathbb{E}[f(S_i)] \leq \left(1-\frac{1}{k}\right)^i\cdot[f(OPT)-\mathbb{E}[f(S_0)]]$$
$$\leq \left(1-\frac{1}{k}\right)^i\cdot f(OPT) \ .$$

Thus,

$$\mathbb{E}[f(S_k)] \geq \left[1-\left(1-\frac{1}{k}\right)^k\right]\cdot f(OPT)$$
$$\geq (1-1/e)\cdot f(OPT) \ .$$

Next, we consider the general case.

OBSERVATION 1. *For every $0 \leq i \leq k$, $\mathbb{E}[f(OPT\cup S_i)] \geq (1-1/k)^i\cdot f(OPT)$.*

Proof. In each iteration $1 \leq i \leq k$ of the algorithm, each element of $\mathcal{N}\setminus S_{i-1}$ stays outside of S_i with probability at least $1-1/k$. Therefore, for every $0 \leq i \leq k$, each elements belongs to S_i with probability at most $1-(1-1/k)^i$. Let $g : 2^{\mathcal{N}} \to \mathbb{R}^+$ be the function $g(S) = f(S\cup OPT)$. Observe that g is a submodular function, and by Lemma 2.2:

$$\mathbb{E}[f(OPT\cup S_i)] = \mathbb{E}[g(S_i\setminus OPT)] \geq (1-1/k)^i\cdot g(\varnothing)$$
$$= (1-1/k)^i\cdot f(OPT) \ .$$

We are now ready to prove that Random Greedy has the approximation ratio guaranteed by Theorem 1.3.

Proof. [Proof of Theorem 1.3] Fix $1 \leq i \leq k$ and an event $A_{i-1} \in \mathcal{A}_{i-1}$. All the probabilities, expectations and random quantities in the first part of this proof are implicitly conditioned on A_{i-1}. Consider a set M_i' containing the elements of $OPT \setminus S_{i-1}$ plus enough dummy elements to make the size of M_i' exactly k. Observe that:

$$\mathbb{E}[f_{u_i}(S_{i-1})] = k^{-1}\cdot\sum_{u\in M_i} f_u(S_{i-1})$$
$$\geq k^{-1}\cdot\sum_{u\in M_i'} f_u(S_{i-1}) = k^{-1}\cdot\sum_{u\in OPT\setminus S_{i-1}} f_u(S_{i-1})$$
$$\geq \frac{f(OPT\cup S_{i-1})-f(S_{i-1})}{k} \ ,$$

where the first inequality follows from the definition of M_i and the second from the submodularity of f. Unfixing the event A_{i-1}, and taking an expectation over all possible such events, we get:

$$\mathbb{E}[f_{u_i}(S_{i-1})] \geq \frac{\mathbb{E}[f(OPT\cup S_{i-1})]-\mathbb{E}[f(S_{i-1})]}{k}$$

Algorithm 1: Random Greedy(f, k)

Initialize: $S_0 \leftarrow \varnothing$.
for $i = 1$ **to** k **do**
  Let $M_i \subseteq \mathcal{N} \setminus S_{i-1}$ be a subset of size k maximizing $\sum_{u \in M_i} f_u(S_{i-1})$.
  Let u_i be a uniformly random element from M_i.
  $S_i \leftarrow S_{i-1} + u_i$.
Return S_k.

$$\geq \frac{\left(1 - \frac{1}{k}\right)^{i-1} \cdot f(OPT) - \mathbb{E}[f(S_{i-1})]}{k} \ ,$$

where the second inequality is due to Observation 1. Let us prove by induction that $\mathbb{E}[f(S_i)] \geq (i/k) \cdot (1 - 1/k)^{i-1} \cdot f(OPT)$. For $i = 0$, this is true since $f(S_0) \geq 0 = (0/k) \cdot (1 - 1/k)^{-1} \cdot f(OPT)$. Assume now that the claim holds for every $i' < i$, let us prove it for $i > 0$.

$$\begin{aligned}
&\mathbb{E}[f(S_i)] = \mathbb{E}[f(S_{i-1})] + \mathbb{E}[f_{u_i}(S_{i-1})] \\
&\geq \mathbb{E}[f(S_{i-1})] + \frac{\left(1 - \frac{1}{k}\right)^{i-1} \cdot f(OPT) - \mathbb{E}[f(S_{i-1})]}{k} \\
&= (1 - 1/k) \cdot \mathbb{E}[f(S_{i-1})] + k^{-1}(1 - 1/k)^{i-1} \cdot f(OPT) \\
&\geq (1 - 1/k) \cdot [((i-1)/k) \cdot (1 - 1/k)^{i-2} \cdot f(OPT)] \\
&\quad + k^{-1}(1 - 1/k)^{i-1} \cdot f(OPT) \\
&= [i/k] \cdot (1 - 1/k)^{i-1} \cdot f(OPT) \ .
\end{aligned}$$

In conclusion:

$$\mathbb{E}[f(S_k)] \geq \frac{k}{k} \cdot \left(1 - \frac{1}{k}\right)^{k-1} \cdot f(OPT) \geq e^{-1} \cdot f(OPT) \ .$$

3.2 The Continuous Double Greedy Algorithm In this section we present the Continuous Double Greedy Algorithm (Algorithm 2). This algorithm provides an approximation guarantee for both variants of the cardinality constraint, and is used for proving Theorems 1.1 and 1.2. This is the second of our two core algorithm and it presents how one can extend the discrete double greedy approach of [8] to the continuous setting, where the sizes of the two evolving sets are more easy to control.

To describe the algorithm, we need some notation. For two vectors $x, y \in [0,1]^{\mathcal{N}}$, we use $x \vee y$ and $x \wedge y$ to denote the coordinate-wise maximum and minimum, respectively, of x and y (formally, $(x \vee y)_u = \max\{x_u, y_u\}$ and $(x \wedge y)_u = \min\{x_u, y_u\}$). We abuse notation both in the description of the algorithm and in its analysis, and unify a set with its characteristic vector and an element with the singleton containing it. Notice that using this notation, given the multilinear extension F of any function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, element $u \in \mathcal{N}$ and a vector $x \in [0,1]^{\mathcal{N}}$, its holds that $\frac{\partial F(x)}{\partial x_u} = F(x \vee u) - F(x \wedge (\mathcal{N} - u))$. We also assume we have an oracle access to the multilinear extension F. If this is not the case, then the value of F can be approximated arbitrarily well using sampling (see, *e.g.*, [9]), which results in a low order term decrease in the approximation ratio.

Algorithm 2 is given as a continuous process executing from time $t = 0$ to time $t = 1$. At any time $t \in [0,1]$ the algorithm maintains two solutions $x^t \leq y^t \in [0,1]^n$. Initially, $x^0 \leftarrow \varnothing$, $y^0 \leftarrow \mathcal{N}$. The evolution of these two solutions over time is described using the derivatives of the coordinates of the solutions x^t and y^t. At any time t the coordinates of x^t are (weakly) increasing, while the coordinates of y^t are (weakly) decreasing. For every coordinate $u \in \mathcal{N}$, the total rate of change of x^t_u and y^t_u is 1. Thus, at the end of the execution, at time $t = 1$, $x^1 = y^1$, and this is the output of the algorithm. In order to transform the resulting fractional solution into an integral solution with the same expected cost, one may use known rounding techniques such as pipage rounding (see [9, 51]). An implementation of the algorithm should be done by a careful discretization, which reduces the approximation ratio by a low order term. We defer the details to the full version of the paper. The description of the algorithm actually includes two variants. The first is for the constraint $|S| \leq k$, and the second for $|S| = k$. The descriptions of the two variants differ only in a single line. As their proofs are (almost) the same, they are done together. We find it elegant that the same proof ideas work for both problems. To simplify notation we use x, y instead of x^t, y^t whenever the superscript t is understood from the context.

We prove below the correctness and the approximation ratio for the case $|S| = k$, and then state the minor modifications in the proof required for the case $|S| \leq k$. We start with a simple, but useful lemma that follows from the submodularity of f.

LEMMA 3.1. *For any solutions $x \leq y$ and $u \in \mathcal{N}$:*

$$a_u + b_u \triangleq \frac{\partial F(x)}{\partial x_u} - \frac{\partial F(y)}{\partial y_u} \geq 0 \ .$$

Proof. By submodularity:

$$\begin{aligned}
a_u + b_u &\triangleq \frac{\partial F(x)}{\partial x_u} - \frac{\partial F(y)}{\partial y_u} \\
&= [F(x \vee u) - F(x \wedge (\mathcal{N} - u))] \\
&\quad - [F(y \vee u) - F(y \wedge (\mathcal{N} - u))]
\end{aligned}$$

Algorithm 2: Continuous Double Greedy(F, k)

Initialize: $x^0 \leftarrow \varnothing$, $y^0 \leftarrow \mathcal{N}$.
at *any time* $t \in [0,1]$ **do**
  define *for every* $u \in \mathcal{N}$ **:**
    $a_u \leftarrow \frac{\partial F(x^t)}{\partial x_u}$, $b_u = -\frac{\partial F(y^t)}{\partial y_u}$.
    $a'_u(\ell) \leftarrow \max\{a_u - \ell, 0\}$, $b'_u(\ell) \leftarrow \max\{b_u + \ell, 0\}$.
    $\frac{dx_u}{dt}(\ell) = \frac{a'_u}{a'_u + b'_u}$, $\frac{dy_u}{dt}(\ell) = -\frac{b'_u}{a'_u + b'_u}$ (see below how to handle the case $a'_u + b'_u = 0$).
  Let ℓ' be a value such that $\sum_{u \in \mathcal{N}} \frac{dx_u}{dt}(\ell') = k$.
  In the case $|S| = k$: Let $\ell^* = \ell'$.
  In the case $|S| \leq k$**:** Let $\ell^* = \max\{\ell', 0\}$.
  Set for every $u \in \mathcal{N}$ the derivatives: $\frac{dx_u}{dt} \leftarrow \frac{dx_u}{dt}(\ell^*)$, $\frac{dy_u}{dt} \leftarrow \frac{dy_u}{dt}(\ell^*)$
Return $x^1 = y^1$.

$$\geq [F(y \vee u) - F(y \wedge (\mathcal{N} - u))] - [F(y \vee u) - F(y \wedge (\mathcal{N} - u))] = 0 \ .$$

Our next objective is to prove that the algorithm obeys the following invariants:

- At any time $t \in [0,1]$, $\sum_{u \in \mathcal{N}} \frac{dx_u}{dt} = k$.
- For any element $u \in \mathcal{N}$ and time $t \in [0,1]$, $\frac{dx_u}{dt}, -\frac{dy_u}{dt} \geq 0$ and $\frac{dx_u}{dt} - \frac{dy_u}{dt} = 1$

Observe that these invariants imply that $x^1 = y^1$ is a feasible solution to the problem. Before we can prove that the invariants are indeed maintained, we have to explain how the algorithm finds ℓ' and how does it deal with the case $a_u + b_u = 0$. By Lemma 3.1 at any time t for which $0 \leq x^t \leq y^t \leq 1$:

$$\frac{dx_u}{dt}(\ell) = \begin{cases} 1 & \ell < -b_u \\ \frac{a_u - \ell}{a_u + b_u} & -b_u \leq \ell \leq a_u \\ 0 & \ell > a_u \end{cases}$$

and

$$\frac{dy_u}{dt}(\ell) = \begin{cases} 0 & \ell < -b_u \\ -\frac{b_u + \ell}{a_u + b_u} & -b_u \leq \ell \leq a_u \\ -1 & \ell > a_u \end{cases}$$

If $a_u + b_u > 0$, then the derivative functions are continuous and defined for every $\ell \in \mathbb{R}$. If $a_u + b_u = 0$, then each derivative function has a single non-continuous point at $\ell = a_u = -b_u$. Let $g(\ell) \triangleq \sum_{u \in \mathcal{N}} \frac{dx_u}{dt}(\ell)$. $g(\ell)$ is defined everywhere except in maybe a finite set of points, is a sum of piecewise linear decreasing functions, and obeys $g(\min_{u \in \mathcal{N}}\{-b_u\} - \varepsilon) = n$ and $g(\max_u\{a_u\} + \varepsilon) = 0$. If for every $u \in \mathcal{N}$, $a_u + b_u > 0$, then $g(\ell)$ is continuous, and therefore, for any $0 \leq k \leq n$, there exists (and is easy to find) a value ℓ' for which $\sum_{u \in \mathcal{N}} \frac{dx_u}{dt}(\ell') = k$. If there are elements u for which $a_u + b_u = 0$, then $g(\ell)$ has non-continuous points. If for some k, there is no ℓ such that $g(\ell) = k$, then there is a non-continuous point ℓ' in which $g(\ell)$ is decreasing from a value larger than k to a value smaller than k. For the (non-continuous) elements u for which $a_u = -b_u = \ell'$, we define the rates in such a way that $g(\ell') = k$, $\frac{dx_u}{dt}, -\frac{dy_u}{dt} \geq 0$ and $\frac{dx_u}{dt} - \frac{dy_u}{dt} = 1$. This guarantees that it is always possible to choose a value ℓ' for which $\sum_{u \in \mathcal{N}} \frac{dx_u}{dt}(\ell') = k$.

From the above discussion, it is clear that the invariants hold at every time t in which $0 \leq x^t \leq y^t \leq 1$. Therefore, the following lemma is all that is needed to complete the proof that the invariants always hold:

LEMMA 3.2. *For every time* $t \in [0,1]$, $0 \leq x^t \leq y^t \leq 1$.

Proof. Assume otherwise, then let t' be the infimum of all the values t violating the constraint. This means that $x^t \leq y^t$ for every $t < t'$, but for every $\delta > 0$, there exists $\varepsilon \in (0, \delta]$ for which $x^{t'+\varepsilon} \not\leq y^{t'+\varepsilon}$.

Note that since $x^t \leq y^t$ for every $t < t'$, the invariants hold for all these times, and therefore, $y^t - x^t = 1 - t$. On the other hand, choosing $\delta \leq (1-t)/2$, we get that: $y_u^{t'+\varepsilon} - x_u^{t'+\varepsilon} < 0$ for some $u \in \mathcal{N}$ and $\varepsilon \in (0, (1-t)/2]$, which implies that either the derivative of x_u or of $-y_u$ must be larger than 1 somewhere in the range $[t', t' + \varepsilon]$, which is a contradiction to the definition of the algorithm.

Let OPT be the optimal solution for the problem in hand. Then, we make the following useful definition:

$$OPT(x, y) \triangleq (OPT \vee x) \wedge y \ .$$

Notice that $x \leq OPT(x,y) \leq y$ since $x \leq y$. In addition, we observe that, by the properties of the algorithm, at time $t = 0$, $OPT(x,y) = OPT$, and at time $t = 1$, $OPT(x,y) = x^1 = y^1$. Also, we have the following useful observation that follows since $|OPT| = k$ and $\frac{dx_u}{dt} - \frac{dy_u}{dt} = 1$ for every $u \in \mathcal{N}$.

OBSERVATION 2.

$$\ell^* \left(\sum_{u \notin OPT} \frac{dx_u}{dt} + \sum_{u \in OPT} \frac{dy_u}{dt} \right)$$

$$= \ell^* \left(\sum_{u \in \mathcal{N}} \frac{dx_u}{dt} - \sum_{u \in OPT} \left(\frac{dx_u}{dt} - \frac{dy_u}{dt} \right) \right)$$
$$= \ell^* \left(\sum_{u \in \mathcal{N}} \frac{dx_u}{dt} - k \right) = 0 \ .$$

Before analyzing the approximation ratio of Algorithm 2, we derive a bound on the change in the value of $OPT(x, y)$ while x and y are evolving. The proof of the following lemma is omitted from this extended abstract due to space constraints.

LEMMA 3.3. *For every element $u \notin OPT$:*

$$-\frac{\partial F(OPT(x,y))}{\partial x_u} \leq -\frac{\partial F(y)}{\partial y_u} = b_u \ ,$$

and for every element $u \in OPT$:

$$\frac{\partial F(OPT(x,y))}{\partial y_u} \leq \frac{\partial F(x)}{\partial x_u} = a_u \ .$$

We are now ready to prove:

THEOREM 3.2. $F(x^1) = F(y^1) \geq$

$$\frac{f(OPT) + \frac{1}{2}\left(\sqrt{\frac{n-k}{k}} f(\varnothing) + \sqrt{\frac{k}{n-k}} f(\mathcal{N}) \right)}{1 + \frac{1}{2} \frac{n}{\sqrt{(n-k)k}}} \ .$$

Proof. The proof follows from the set of inequalities appearing as Figure 1. In some of the inequalities we have a term of the form $\max\{0, A/(a_u + b_u)\}$, where A is an arbitrary expression. For consistency, we assume that when $a_u + b_u = 0$ this term is equal to 0.

Inequality (3.1) follows by Lemma 3.3. Equality (3.2) follows by Observation (2). Inequality (3.3) follows since for every $u \in \mathcal{N}$:

- If $\ell^* \leq -b_u$, then $(\ell^* - a_u)\frac{dy_u}{dt} = 0$ and $(b_u + \ell^*)\frac{dx_u}{dt} = (b_u + \ell^*) \cdot 1 \leq 0$.
- If $\ell^* \geq a_u$, then $(\ell^* - a_u)\frac{dy_u}{dt} = (\ell^* - a_u) \cdot (-1) \leq 0$ and $(b_u + \ell^*)\frac{dx_u}{dt} = 0$.
- If $a_u + b_u = 0$ and $\ell^* = a_u = -b_u$ then the rates $\frac{dx_u}{dt}$ and $\frac{dy_u}{dt}$ may have any value between 0 and 1. However, in this case $(\ell^* - a_u)\frac{dy_u}{dt} = (b_u + \ell^*)\frac{dx_u}{dt} = 0$.
- Otherwise, $-b_u \leq \ell^* \leq a_u$ and $a_u + b_u > 0$, and therefore, $(\ell^* - a_u)\frac{dy_u}{dt} = \frac{(a_u - \ell^*)(b_u + \ell^*)}{a_u + b_u} = (b_u + \ell^*)\frac{dx_u}{dt}$.

Inequality (3.4) follows since if $a_u - \ell^* < 0$ then $\frac{dx_u}{dt} = 0$ and $b_u + \ell^* \geq 0$ meaning that the RHS is non-negative and the LHS is zero. An analogous argument follows if $b_u + \ell^* < 0$. Otherwise, we use the fact that for every pair $r_1, r_2 \in \mathbb{R}$: $r_1 r_2 \leq \frac{1}{2}(r_1^2 + r_2^2)$. Note that if $a_u + b_u = 0$ and $\ell^* = a_u = -b_u$ then both LHS and RHS are 0. Finally, Equality (3.5) follows since $\sum_{u \in \mathcal{N}} \frac{dx_u}{dt} = k$ and $\sum_{u \in \mathcal{N}} \frac{dy_u}{dt} = k - n$, which implies: $\left(\sqrt{\frac{k}{n-k}} \sum_{u \in \mathcal{N}} \frac{dy_u}{dt} + \sqrt{\frac{n-k}{k}} \sum_{u \in \mathcal{N}} \frac{dx_u}{dt} \right) = 0$.

Finally, integrating both sides of Figure 1 from $t = 0$ to $t = 1$ we get:

$$f(OPT) - F(x^1 = y^1) \leq \frac{1}{2} \frac{n}{\sqrt{(n-k)k}} F(x^1 = y^1) - \frac{1}{2}\left(\sqrt{\frac{n-k}{k}} f(\varnothing) + \sqrt{\frac{k}{n-k}} f(\mathcal{N}) \right) \ .$$

The theorem now follows by rearranging the terms.

COROLLARY 3.1. *Algorithm 2 provides $\frac{1}{1 + \frac{1}{2}\frac{n}{\sqrt{(n-k)k}}}$-approximation for maximizing submodular function under a cardinality constraint $|S| = k$. In particular it achieves an approximation factor of $\frac{1}{2}$ when $k = \frac{n}{2}$.*

Modifications for the case $|S| \leq k$: We review here the modifications necessary for proving the same results for the case $|S| \leq k$. In general the proof is almost identical except for the following minor modifications. First, it is easy to see that if $\ell^* > 0$, then $g(0) \geq k$, and therefore, we get, as before, $\sum_{u \in \mathcal{N}} \frac{dx_u}{dt} = k$.

Second, Observation 2 is slightly different, and holds with inequality instead of equality,

$$\ell^* \left(\sum_{u \notin OPT} \frac{dx_u}{dt} + \sum_{u \in OPT} \frac{dy_u}{dt} \right)$$
$$= \ell^* \left(\sum_{u \in \mathcal{N}} \frac{dx_u}{dt} - \sum_{u \in OPT} \left(\frac{dx_u}{dt} - \frac{dy_u}{dt} \right) \right)$$
$$\geq \ell^* \left(\sum_{u \in \mathcal{N}} \frac{dx_u}{dt} - k \right) = 0 \ .$$

The inequality follows since $|OPT| \leq k$ and $\ell^* \geq 0$, and the equality holds since $\sum_{u \in \mathcal{N}} \frac{dx_u}{dt} = k$ unless $\ell^* = 0$. This change makes Equality (3.2) an inequality. Finally, in Equality (3.5) it may happen that $\sum_{u \in \mathcal{N}} \frac{dx_u}{dt} < k$ and $\sum_{u \in \mathcal{N}} \frac{dy_u}{dt} < k - n$, but in this case $\ell^* = 0$, and so the equality still holds.

Corollary 3.1 shows that Continuous Double Greedy achieves an approximation ratio equal to the second term in the max expression of Theorems 1.1 and 1.2. The first term is achieved by algorithms presented in Sections 4.2 and 4.3, respectively. Each one of the algorithms guaranteed by these theorems executes Continuous Double Greedy and one of the other algorithms, and outputs the better solution.

$$
\begin{aligned}
(3.1)\qquad & -\frac{dF(OPT(x,y))}{dt} \leq -\sum_{u\in OPT} a_u\frac{dy_u}{dt} + \sum_{u\notin OPT} b_u\frac{dx_u}{dt} \\
(3.2)\qquad & = -\sum_{u\in OPT} a_u\frac{dy_u}{dt} + \sum_{u\notin OPT} b_u\frac{dx_u}{dt} + \ell^*\left(\sum_{u\notin OPT}\frac{dx_u}{dt} + \sum_{u\in OPT}\frac{dy_u}{dt}\right) \\
& = \sum_{u\in OPT}(\ell^* - a_u)\frac{dy_u}{dt} + \sum_{u\notin OPT}(b_u+\ell^*)\frac{dx_u}{dt} \\
(3.3)\qquad & \leq \sum_{u\in\mathcal{N}} \max\left\{0, \frac{(a_u-\ell^*)\cdot(b_u+\ell^*)}{a_u+b_u}\right\} \\
& = \sum_{u\in\mathcal{N}} \max\left\{0, \left(\frac{(a_u-\ell^*)\cdot\left(\frac{n-k}{k}\right)^{1/4}}{\sqrt{a_u+b_u}}\right)\cdot\left(\frac{(b_u+\ell^*)\cdot\left(\frac{k}{n-k}\right)^{1/4}}{\sqrt{a_u+b_u}}\right)\right\} \\
(3.4)\qquad & \leq \frac{1}{2}\sum_{u\in\mathcal{N}}\left(\sqrt{\frac{n-k}{k}}\cdot(a_u-\ell^*)\frac{dx_u}{dt} - \sqrt{\frac{k}{n-k}}\cdot(b_u+\ell^*)\frac{dy_u}{dt}\right) \\
& = \frac{1}{2}\left(\sqrt{\frac{n-k}{k}}\frac{dF(x)}{dt} + \sqrt{\frac{k}{n-k}}\frac{dF(y)}{dt}\right) - \frac{\ell^*}{2}\left(\sqrt{\frac{k}{n-k}}\sum_{u\in\mathcal{N}}\frac{dy_u}{dt} + \sqrt{\frac{n-k}{k}}\sum_{u\in\mathcal{N}}\frac{dx_u}{dt}\right) \\
(3.5)\qquad & = \frac{1}{2}\left(\sqrt{\frac{n-k}{k}}\frac{dF(x)}{dt} + \sqrt{\frac{k}{n-k}}\frac{dF(y)}{dt}\right) = \frac{1}{2}\frac{n}{\sqrt{(n-k)k}}\left(\frac{n-k}{n}\frac{dF(x)}{dt} + \frac{k}{n}\frac{dF(y)}{dt}\right) \ .
\end{aligned}
$$

Figure 1: Upper bound on the rate in which $F(OPT(x,y))$ decreases.

4 Extended Algorithms

4.1 Fast Algorithms Subject to Matroid Independence

A matroid is a pair $(\mathcal{N},\mathcal{I})$, where $\mathcal{N}$ is a ground set, and $\mathcal{I}\subseteq 2^{\mathcal{N}}$ is a collection of subsets of $\mathcal{N}$. The collection $\mathcal{I}$ must obey the following three properties:

- Non-empty: $\mathcal{I}\neq\emptyset$.
- Monotone: If $A\subseteq B\in\mathcal{I}$, then $A\in\mathcal{I}$.
- Exchange: If $A,B\in\mathcal{I}$ and $|A|<|B|$, then there exists an element $u\in B$ for which $A+u\in\mathcal{I}$.

If $S\in\mathcal{I}$, we say that S is *independent*, and if S is also maximal inclusion-wise, we say it is a *base*. It is well known that all the bases of a matroid $\mathcal{M}$ have an equal size known called the rank of $\mathcal{M}$ (we denote the rank of $\mathcal{M}$ by k). Matroids capture many natural collections of subsets such as: forests in graphs, independent sets in vector spaces and the sets of nodes that appear together in legal matchings of a given graph [47, 39]. Like submodular functions, the explicit representation of a matroid might also be exponential in the size of its ground set. The standard way to bypass this difficulty is, again, using an oracle. For a matroid $M=(\mathcal{N},\mathcal{I})$, given a set $S\subseteq\mathcal{N}$, the oracle answers whether $S\in\mathcal{I}$. All the algorithms we describe access matroids via such oracles.

In this section we give two fast algorithms for the problem $\max\{f(S)\mid S\in\mathcal{I}\}$, where $\mathcal{I}$ is the collection of independent sets of a matroid $\mathcal{M}=(\mathcal{N},\mathcal{I})$. These algorithms are the algorithms guaranteed by Theorems 1.5 and Theorems 1.6. One of the algorithms is faster than the other one, but achieves a slightly worse approximation ratio. In a similar fashion to Reduction 1, we assume the ground set contains a set D of $2k$ "dummy" elements that is known to our algorithms and has two properties:

- $f(S)=f(S\setminus D)$ for every set $S\subseteq\mathcal{N}$.
- $S\in\mathcal{I}$ if and only if $S\setminus D\in\mathcal{I}$ and $|S|\leq k$.

Like in the proof of Reduction 1, we can justify our assumptions by adding such $2k$ dummy elements to the ground set, and redefining f and $\mathcal{I}$ to the extended ground set using the above properties. Observe that the existence of the set D allows us to assume also that OPT is a base of $\mathcal{M}$. We also need the following reduction which corresponds to Reduction 3.

REDUCTION 4. *We may assume the rank k of the matroid is larger than an arbitrarily large constant. For our algorithms this reduction increase the time complexity by a constant factor.*

Proof. Assume k is smaller than an arbitrary universal constant c, and let $c' = \lceil c/k \rceil$ and $[c'] = \{1, 2, \ldots, c'\}$. We replace $\mathcal{N}$ with the ground set $\mathcal{N}' = \mathcal{N} \times [c']$, and define for every set S and $i \in [c']$ the set $S_{=i} = \{u \in \mathcal{N} : (u, i) \in S\}$. We now replace the collection of independent sets with $\mathcal{I}' = \{S \subseteq \mathcal{N}' : \forall_i S_{=i} \in \mathcal{I}\}$ and f with the function $f' : \mathcal{N}' \to \mathbb{R}$ defined as $f'(S) = \sum_{i=1}^{c'} f(S_{=i})$. It is easy to see that f' is submodular, and that there exists an approximation ratio preserving reduction from the original problem to the new one.

In our algorithms one can maintain the sets $S_{=1}, S_{=2}, \ldots, S_{=c'}$ for every set S in the algorithm without increasing the time complexity by more than a constant factor. Using these sets, it is possible to evaluate f' in constant time.

Given a set $S \subseteq \mathcal{N}$, let $\mathcal{M}/S$ be the contracted matroid $(\mathcal{N} \setminus S, \mathcal{I}_{\mathcal{M}/S})$, in which a set $S' \in \mathcal{N} \setminus S$ belongs to $\mathcal{I}_{\mathcal{M}/S}$ if and only if $S' \cup S \in \mathcal{I}$.

Consider Algorithm 3. Observe that M_i can be efficiently found via the standard greedy algorithm. For every $0 \leq i \leq \lceil k/2 \rceil$, we construct a random set OPT_i for which $S_i \cup OPT_i$ is a base. For the construction we need the following lemma from [7], which can be found (with a different notation) as Corollary 39.12a in [46].

LEMMA 4.1. *If A and B are two bases of a matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, then there exists a one to one function $g : A \setminus B \to B \setminus A$ such for every $u \in B \setminus A$, $[A \cup \{u\}] \setminus \{g(u)\} \in \mathcal{I}$.*

For $i = 0$, we define $OPT_0 = OPT$. For $i > 0$, OPT_i is constructed recursively based on the algorithm's behavior. Assume OPT_{i-1} is already constructed, then let $g_i : M_i \to OPT_{i-1}$ be a one to one function mapping every element $u \in M_i$ to an element of OPT_{i-1} in such a way that $S_{i-1} \cup OPT_{i-1} + u - g_i(u)$ is a base. Observe that the existence of such function follows immediately from Lemma 4.1. We now set $OPT_i = OPT_{i-1} - g_i(u_i)$. It is important that the choice of g_i (among the possibly multiple functions obeying the required properties) is independent of the random choice of u_i, which makes $g_i(u_i)$ a uniformly random sample from OPT_{i-1}.

Let A_i be an event fixing all the random decisions of Algorithm 3 up to iteration i (including), and let $\mathcal{A}_i$ be the set of all possible A_i events.

OBSERVATION 3. *For every $0 \leq i \leq \lceil k/2 \rceil$, $\mathbb{E}[f(OPT_i \cup S_i)] \geq \frac{(k-i)(k-i-1)}{k(k-1)} \cdot f(OPT)$.*

Proof. Fix $1 \leq i \leq k$ and an event $A_{i-1} \in \mathcal{A}_{i-1}$. All the probabilities, expectations and random quantities in the first part of this proof are implicitly conditioned on A_{i-1}. Observe that:

$$\begin{aligned}
&\mathbb{E}[f_{u_i}(OPT_{i-1} \cup S_{i-1})] \\
&= (k-i+1)^{-1} \cdot \sum_{u \in M_i} f_u(OPT_{i-1} \cup S_{i-1}) \\
&\geq (k-i+1)^{-1} \cdot [f(OPT_{i-1} \cup S_{i-1} \cup M_i) \\
&\quad - f(OPT_{i-1} \cup S_{i-1})] \\
&\geq -\frac{f(OPT_{i-1} \cup S_{i-1})}{k-i+1} \ ,
\end{aligned}$$

where the first inequality follows from submodularity. Similarly,

$$\begin{aligned}
&\mathbb{E}[f_{g_i(u_i)}(OPT_{i-1} \cup S_{i-1} - g_i(u_i))] \\
&= (k-i+1)^{-1} \cdot \sum_{u \in M_i} f_{g_i(u)}(OPT_{i-1} \cup S_{i-1} - g_i(u)) \\
&\leq (k-i+1)^{-1} \cdot [f(OPT_{i-1} \cup S_{i-1}) \\
&\quad - f(OPT_{i-1} \cup S_{i-1}) \setminus M_i] \\
&\leq \frac{f(OPT_{i-1} \cup S_{i-1})}{k-i+1} \ ,
\end{aligned}$$

where the first inequality follows, again, from submodularity. Unfixing the event A_{i-1}, and taking an expectation over all possible such events, we get:

$$\mathbb{E}[f_{u_i}(OPT \cup S_{i-1})] \geq -\frac{\mathbb{E}[f(OPT \cup S_{i-1})]}{k-i+1} \ ,$$

and

$$\mathbb{E}[f_{g_i(u_i)}(OPT_{i-1} \cup S_{i-1} - g_i(u_i))] \leq \frac{\mathbb{E}[f(OPT \cup S_{i-1})]}{k-i+1} \ ,$$

We are now ready to prove the observation by induction on i. For $i = 0$, the lemma holds since $\mathbb{E}[f(OPT_0 \cup S_0)] = f(OPT) = \frac{(k-0)(k-0-1)}{k(k-1)} \cdot f(OPT)$. Assume the lemma holds for $i' < i$, and let us prove it for $i > 0$.

$$\begin{aligned}
&\mathbb{E}[f(OPT_i \cup S_i)] \\
&= \mathbb{E}[f(OPT_{i-1} \cup S_{i-1} + u_i - g_i(u_i))] \\
&\geq \mathbb{E}[f(OPT_{i-1} \cup S_{i-1} + u_i)] \\
&\quad + \mathbb{E}[f(OPT_{i-1} \cup S_{i-1} - g_i(u_i))] \\
&\quad - \mathbb{E}[f(OPT_{i-1} \cup S_{i-1})] \\
&= \mathbb{E}[f(OPT_{i-1} \cup S_{i-1})] + \mathbb{E}[f_{u_i}(OPT_{i-1} \cup S_{i-1})] \\
&\quad - \mathbb{E}[f_{g(u_i)}(OPT_{i-1} \cup S_{i-1} - g_i(u_i))] \\
&\geq \mathbb{E}[f(OPT_{i-1} \cup S_{i-1})] - \frac{\mathbb{E}[f(OPT_{i-1} \cup S_{i-1})]}{k-i+1} \\
&\quad - \frac{\mathbb{E}[f(OPT_{i-1} \cup S_{i-1})]}{k-i+1} \\
&= \frac{(k-i+1)-2}{k-i+1} \cdot \mathbb{E}[f(OPT_{i-1} \cup S_{i-1})]
\end{aligned}$$

Algorithm 3: Residual Random Greedy for Matroids$(f, \mathcal{M})$

Initialize: $S_0 \leftarrow \varnothing$.
for $i = 1$ **to** k **do**
  Let M_i be a base of $\mathcal{M}/S_{i-1}$ maximizing $\sum_{u \in M_i} f_u(S_{i-1})$, where $\mathcal{M}/S_{i-1}$ is the matroid $\mathcal{M}$ with the set S_{i-1} contracted.
  Let u_i be a uniformly random element from M_i.
  $S_i \leftarrow S_{i-1} + u_i$.
Return S_k.

$$\geq \frac{k-i-1}{k-i+1} \cdot \frac{(k-i+1)(k-i)}{k(k-1)} \cdot f(OPT)$$
$$= \frac{(k-i-1)(k-i)}{k(k-1)} \cdot f(OPT) \ .$$

The last inequality follows from the inductive assumption.

We are now ready to prove that Algorithm 3 provides the approximation ratio guaranteed by Theorem 1.5.

Proof. [Proof of Theorem 1.5] Observe that for every $1 \leq i \leq k$, there are at least k elements in $D \setminus S_{i-1}$ and every element $u \in D \setminus S_{i-1}$ obeys: $f_u(S_{i-1}) = 0$. By definition, every element of $D \setminus (S_{i-1} \cup M_i)$ can take the place of every element of M_i, and therefore, no element of M_i has a negative marginal contribution (*i.e.*, $f_u(S_{i-1}) \geq 0$ for every $u \in M_i$). Since Algorithm 3 always adds an element of M_i to S_{i-1}, we get $f(S_i) \geq f(S_{i-1})$. Hence, in order to prove the theorem, it is enough to show that $f(S_i) \geq 1/4$ for some $0 \leq i \leq k$. In the rest of this proof, we will show that $f(S_{\lceil k/2 \rceil}) \geq 1/4$.

Fix $1 \leq i \leq \lceil k/2 \rceil$ and an event $A_{i-1} \in \mathcal{A}_{i-1}$. All the probabilities, expectations and random quantities in the first part of this proof are implicitly conditioned on A_{i-1}. Observe that:

$$\begin{aligned}\mathbb{E}[f_{u_i}(S_{i-1})] &= (k-i+1)^{-1} \cdot \sum_{u \in M_i} f_u(S_{i-1}) \\ &\geq (k-i+1)^{-1} \cdot \sum_{u \in OPT_{i-1}} f_u(S_{i-1}) \\ &\geq \frac{f(OPT_{i-1} \cup S_{i-1}) - f(S_{i-1})}{k-i+1} \ ,\end{aligned}$$

where the first inequality follows from the definition of M_i and the fact that $S_{i-1} \cup OPT_{i-1}$ is a base. Unfixing the event A_{i-1}, and taking an expectation over all possible such events, we get:

$$\begin{aligned}\mathbb{E}[f_{u_i}(S_{i-1})] &\geq \frac{\mathbb{E}[f(OPT_{i-1} \cup S_{i-1})] - \mathbb{E}[f(S_{i-1})]}{k-i+1} \\ &\geq \frac{\frac{(k-i+1)(k-i)}{k(k-1)} \cdot f(OPT) - \mathbb{E}[f(S_{i-1})]}{k-i+1} \ ,\end{aligned}$$

where the second inequality is due to Observation 3. Let us prove by induction that $\mathbb{E}[f(S_i)] \geq \frac{i(k-i)}{k(k-1)} \cdot f(OPT)$ for $0 \leq i \leq \lceil k/2 \rceil$. For $i = 0$, this is true since $f(S_0) \geq 0 = \frac{0(k-0)}{k(k-1)} \cdot f(OPT)$. Assume now that the claim holds for every $i' < i$, let us prove it for $i > 0$.

$$\begin{aligned}\mathbb{E}[f(S_i)] &= \mathbb{E}[f(S_{i-1})] + \mathbb{E}[f_{u_i}(S_{i-1})] \\ &\geq \mathbb{E}[f(S_{i-1})] \\ &\quad + \frac{\frac{(k-i+1)(k-i)}{k(k-1)} \cdot f(OPT) - \mathbb{E}[f(S_{i-1})]}{k-i+1} \\ &= \frac{k-i}{k-i+1} \cdot \mathbb{E}[f(S_{i-1})] + \frac{k-i}{k(k-1)} \cdot f(OPT) \\ &\geq \frac{k-i}{k-i+1} \cdot \frac{(i-1)(k-i+1)}{k(k-1)} \cdot f(OPT) \\ &\quad + \frac{k-i}{k(k-1)} \cdot f(OPT) \\ &= \frac{(k-i)(i-1) + (k-i)}{k(k-1)} \cdot f(OPT) \\ &= \frac{i(k-i)}{k(k-1)} \cdot f(OPT) \ .\end{aligned}$$

To conclude the proof, we need to show that $\frac{i(k-i)}{k(k-1)} \geq 1/4$ for $i = \lceil k/2 \rceil$. Let us consider two cases. If k is even, then, for $i = k/2$, we get:

$$\frac{i(k-i)}{k(k-1)} = \frac{(k/2)^2}{k(k-1)} = \frac{1}{4} \cdot \frac{k}{k-1} \geq \frac{1}{4} \ .$$

If k is odd, then, for $i = k/2 + 1/2$, we get:

$$\begin{aligned}\frac{i(k-i)}{k(k-1)} &= \frac{(k/2+1/2)(k/2-1/2)}{k(k-1)} \\ &= \frac{1}{4} \cdot \frac{(k+1)(k-1)}{k(k-1)} = \frac{1}{4} \cdot \frac{k+1}{k} \geq \frac{1}{4} \ .\end{aligned}$$

Next, consider Algorithm 4. Observe that S_i is a base of $\mathcal{M}$ for every $0 \leq i \leq k$ and M_i can be efficiently found via a standard greedy algorithm. The existence of g_i is guaranteed by Lemma 4.1, and it can be found using an algorithm for finding a perfect matching in a bipartite matching. Mucha and Sankowski [43] give such an algorithm whose expected time complexity is $O(k^\omega)$, where ω is the exponent of matrix multiplication.

Algorithm 4: Random Greedy for Matroids$(f, \mathcal{M})$

Initialize: S_0 to be an arbitrary base containing only elements of D.
for $i = 1$ **to** k **do**
  Let $M_i \subseteq \mathcal{N} \setminus S_{i-1}$ be a base of $\mathcal{M}$ containing only elements of $\mathcal{N} \setminus S_{i-1}$ and maximizing $\sum_{u \in M_i} f_u(S_{i-1})$.
  Let g_i be a function mapping each element of M_i to an element of S_{i-1} obeying $S_{i-1} + u - g_i(u) \in \mathcal{I}$ for every $u \in S_{i-1}$.
  Let u_i be a uniformly random element from M_i.
  $S_i \leftarrow S_{i-1} + u_i - g_i(u_i)$.
Return S_k.

OBSERVATION 4. *For every* $0 \leq i \leq k$, $\mathbb{E}[f(OPT \cup S_i)] \geq 0.5(1 + (1 - 2/k)^i) \cdot f(OPT)$.

Proof. Let $p_{i,u}$ be the probability an element $u \in \mathcal{N} \setminus D$ belongs to S_i for some $0 \leq i \leq k$. For every $1 \leq i \leq k$, each element of $\mathcal{N} \setminus S_{i-1}$ stays outside of S_i with probability at least $1 - 1/k$, independently of what happened in previous iterations. On the other hand, an element of S_{i-1} gets into S_i with probability of only $1 - 1/k$. Therefore, by the law of total probability:

$$\begin{aligned} p_{i,u} &\leq (1 - p_{i-1,u})/k + p_{i-1,u}(1 - 1/k) \\ &= p_{i-1,u}(1 - 2/k) + 1/k \ . \end{aligned}$$

Let us prove by induction that $p_{i,u} \leq 0.5(1 - (1 - 2/k)^i) \cdot f(OPT)$. For $i = 0$, this is true since $p_{0,u} = 0 \leq 0.5(1 - (1 - 2/k)^0)$. Assume the claims holds for $i' < i$, and let us prove it for i.

$$\begin{aligned} p_{i,u} &\leq p_{i-1,u}(1 - 2/k) + 1/k \\ &\leq 0.5(1 - (1 - 2/k)^{i-1})(1 - 2/k) + 1/k \\ &= 0.5(1 - (1 - 2/k)^i) \ . \end{aligned}$$

Let $h : 2^{\mathcal{N}} \to \mathbb{R}^+$ be the function $h(S) = f(S \cup OPT)$. Observe that h is a submodular function, and by Lemma 2.2, for $1 \leq i \leq I$:

$$\begin{aligned} \mathbb{E}[f(OPT \cup S_i)] &= \mathbb{E}[h(S_i \setminus OPT)] \\ &\geq (1 - \max_{u \in \mathcal{N}} p_{i,u}) \cdot h(\varnothing) \\ &\geq 0.5(1 + (1 - 2/k)^i) \cdot f(OPT) \ . \end{aligned}$$

Let A_i be an event fixing all the random decisions of Algorithm 4 up to iteration i (including), and let $\mathcal{A}_i$ be the set of all possible A_i events. We are now ready to prove Algorithm 4 provides the approximation guaranteed by Theorem 1.6.

Proof. [Proof of Theorem 1.6] Fix $1 \leq i \leq i$ and an event $A_{i-1} \in \mathcal{A}_{i-1}$. All the probabilities, expectations and random quantities in the first part of this proof are implicitly conditioned on A_{i-1}. Consider a set M_i' containing the elements of $OPT \setminus S_{i-1}$ plus enough dummy elements to make its size exactly k. Observe that:

$$\begin{aligned} \mathbb{E}[f_{u_i}(S_{i-1})] &= k^{-1} \cdot \sum_{u \in M_i} f_u(S_{i-1}) \\ &\geq k^{-1} \cdot \sum_{u \in M_i'} f_u(S_{i-1}) \\ &= k^{-1} \cdot \sum_{u \in OPT \setminus S_{i-1}} f_u(S_{i-1}) \\ &\geq \frac{f(OPT \cup S_{i-1}) - f(S_{i-1})}{k} \ , \end{aligned}$$

where the first inequality follows from the definition of M_i and the second from the submodularity of f. Similarly,

$$\begin{aligned} &\mathbb{E}[f_{g(u_i)}(S_{i-1} \setminus \{g(u_i)\})] \\ &\quad = k^{-1} \cdot \sum_{u \in M_i} f_{g(u)}(S_{i-1} \setminus \{g(u)\}) \\ &\quad \leq \frac{f(S_{i-1}) - f(\varnothing)}{k} \leq \frac{f(S_{i-1})}{k} \ , \end{aligned}$$

where the first inequality follows from the submodularity of f. Unfixing the event A_{i-1}, and taking an expectation over all possible such events, we get:

$$\begin{aligned} \mathbb{E}[f_{u_i}(S_{i-1})] &\geq \frac{\mathbb{E}[f(OPT \cup S_{i-1})] - \mathbb{E}[f(S_{i-1})]}{k - i + 1} \\ &\geq \frac{0.5(1 + (1 - 2/k)^{i-1}) \cdot f(OPT) - \mathbb{E}[f(S_{i-1})]}{k} \end{aligned}$$

(where the second inequality is due to Observation 4), and

$$\mathbb{E}[f_{g(u_i)}(S_{i-1} \setminus \{g(u_i)\})] \leq \frac{\mathbb{E}[f(S_{i-1}])}{k} \ .$$

Let us prove by induction that $\mathbb{E}[f(S_i)] \geq 0.25[1 + (2(i+1)/k - 1)(1 - 2/k)^{i-1}] \cdot f(OPT)$. For $i = 0$, this is true since $f(S_0) \geq 0 = 0.25[1 + (2/k - 1)(1 - 2/k)^{-1}] \cdot f(OPT)$. Assume now that the claim holds for every $i' < i$, let us prove it for $i > 0$.

$$\mathbb{E}[f(S_i)] = \mathbb{E}[f([S_{i-1} \cup \{u_i\}] \setminus \{g(u_i)\})]$$

$$
\begin{aligned}
&\geq \mathbb{E}[f(S_{i-1} \cup \{u_i\}) + f(S_{i-1} \setminus \{g(u_i)\}) - f(S_{i-1})] \\
&= \mathbb{E}[f(S_{i-1})] + \mathbb{E}[f_{u_i}(S_{i-1})] - \mathbb{E}[f_{g(u_i)}(S_{i-1} \setminus \{g(u_i)\})] \\
&\geq \mathbb{E}[f(S_{i-1})] \\
&\quad + \frac{0.5(1+(1-2/k)^{i-1}) \cdot f(OPT) - \mathbb{E}[f(S_{i-1})]}{k} \\
&\quad - \frac{\mathbb{E}[f(S_{i-1}])}{k} \\
&= (1-2/k) \cdot \mathbb{E}[f(S_{i-1})] \\
&\quad + \frac{0.5(1+(1-2/k)^{i-1}) \cdot f(OPT)}{k} \\
&\geq (1-2/k) \cdot \frac{1+(2i/k-1)(1-2/k)^{i-2}}{4} \cdot f(OPT) \\
&\quad + \frac{1+(1-2/k)^{i-1}}{2k} \cdot f(OPT) \\
&= \frac{1+(2(i+1)/k-1)(1-2/k)^{i-1}}{4} \cdot f(OPT) \ .
\end{aligned}
$$

In conclusion:

$$
\begin{aligned}
\mathbb{E}[f(S_k)] &\geq \frac{1+(2(k+1)/k-1)(1-2/k)^{k-1}}{4} \cdot f(OPT) \\
&\geq \frac{1+\frac{e^{-2}(1+2/k)(1-4/k)}{1-2/k}}{4} \cdot f(OPT) \\
&\geq \left[\frac{1+e^{-2}}{4} - \varepsilon\right] \cdot f(OPT) \ ,
\end{aligned}
$$

where the last inequality holds for large enough k.

4.2 Breaking the $1/e$ Barrier Subject to $|S| \leq k$. Here we give an algorithm for $\max\{f(S) \mid |S| \leq k\}$ that has an approximation ratio better than $1/e$ for all values of k. This algorithm is used to prove Theorem 1.1. Random Greedy (Algorithm 1) chooses at each iteration a random element out of the k elements with the largest marginal values. Algorithm 5 is a variant of this algorithm that in some iterations chooses a random element out of a larger set. Recall that we assume Reduction 1 was applied to the input.

Algorithm 5 uses two parameters I and $\Sigma(i)$ defined as following:

$$
I = \lceil 0.21k \rceil \qquad \text{and} \qquad \Sigma(i) = \begin{cases} 2(k-i+1) & i \leq I \\ k & i > I \end{cases}
$$

OBSERVATION 5. *For every $0 \leq i \leq k$, $|S(i)| = i$. Hence, Algorithm 5 outputs a feasible solution.*

To simply the analysis of the algorithm, we make use of the following notation:

$$
\sigma_i = 0.5 \cdot \sum_{j=1}^{i} (k-j+1)^{-1}
$$

and

$$
\pi_i = \prod_{j=1}^{i} \left(1 - 0.5(k-j+1)^{-1}\right) \ .
$$

It is possible to present a "continuous greedy like" version of the algorithm. In this version σ_i becomes the time t, and π_i becomes e^{-t}. Like in the analysis of the random greedy algorithm, we start by deriving a lower bound on the value of $f(OPT \cup S_i)$.

LEMMA 4.2. *For every $0 \leq i \leq k$,*

$$
\frac{\mathbb{E}[f(OPT \cup S_i)]}{f(OPT)} \geq \begin{cases} \pi_i & i \leq I \\ \pi_I \cdot (1-1/k)^{i-I} & i \geq I \end{cases}
$$

Remark: Observe that the two lower bounds given by Lemma 4.2 for $\mathbb{E}[f(OPT \cup S_I)]/f(OPT)$ are identical.

Proof. In each iteration $1 \leq i \leq k$ of the algorithm, each element of $\mathcal{N} \setminus S_{i-1}$ stays outside of S_i with probability at least $1 - 1/\Sigma(i)$. Therefore, the probability of u to be in S_i is upper bounded for $0 \leq i \leq I$ by:

$$
1 - \prod_{j=1}^{i}(1-1/\Sigma(j)) = 1 - \prod_{j=1}^{i}(1-0.5/(k-j+1)) = 1-\pi_i \ ,
$$

and for $I \leq i \leq k$ by:

$$
\begin{aligned}
&1 - \prod_{j=1}^{i}(1-1/\Sigma(j)) \\
&= 1 - \left[\prod_{j=1}^{I}(1-0.5/(k-j+1))\right] \cdot (1-1/k)^{i-I} \\
&= 1 - \pi_I \cdot (1-1/k)^{i-I} \ .
\end{aligned}
$$

Let $h : 2^{\mathcal{N}} \to \mathbb{R}^+$ be the function $h(S) = f(S \cup OPT)$. Observe that h is a submodular function, and by Lemma 2.2, for $1 \leq i \leq I$:

$$
\begin{aligned}
\mathbb{E}[f(OPT \cup S_i)] &= \mathbb{E}[h(S_i \setminus OPT)] \\
&\geq \pi_i \cdot h(\varnothing) = \pi_i \cdot f(OPT) \ ,
\end{aligned}
$$

and for $I \leq i \leq k$:

$$
\begin{aligned}
\mathbb{E}[f(OPT \cup S_i)] &= \mathbb{E}[h(S_i \setminus OPT)] \\
&\geq \pi_I \cdot (1-1/k)^{i-I} \cdot h(\varnothing) \\
&= \pi_I \cdot (1-1/k)^{i-I} \cdot f(OPT) \ .
\end{aligned}
$$

In general there is no guarantee that Algorithm 5 will do well. Hence, we first consider the case when Assumption 1 holds.

ASSUMPTION 1. *For every $1 \leq i \leq I$, the element of M_i with the least marginal value still has in expectation a relatively large marginal value. More formally, let m_i be the element of M_i with the least marginal value. Notice that m_i is a random variable depending on the decisions of the algorithm in previous iterations. Then:*

$$
\mathbb{E}[f_{m_i}(S_{i-1})] \geq \ell(i)/(k-i+1) \cdot f(OPT) \ ,
$$

where $\ell(i) = 0.253 - 2.33\sigma_{i-1} + 4.5\sigma_{i-1}^2$.

Algorithm 5: Wide Random Greedy(f, k)

Initialize: $S_0 \leftarrow \varnothing$.
for i = 1 **to** k **do**
  Let $M_i \subseteq \mathcal{N} \setminus S_{i-1}$ be a subset of size $\Sigma(i)$ maximizing $\sum_{u \in M_i} f_u(S_{i-1})$.
  Let u_i be a uniformly random element from M_i.
  $S_i \leftarrow S_{i-1} + u_i$.
Return $S(k)$.

LEMMA 4.3. *Assuming Assumption 1 holds, then:* $11.583(1-\pi_i)+\sigma_i(\pi_i-11.33+4.5\sigma_i)-4.5/[4(k-i)]$ *for* $i \leq I$ *and* $(1-1/k)^{i-I-1}[(1-1/k)\cdot V(I)+\pi_I\cdot(i-I)/k]$ *for* $i \geq I$*, where* $V(I) = 11.583(1-\pi_I)+\sigma_I(\pi_I-11.33+4.5\sigma_I)-4.5/[4(k-I)]$*. Moreover, the above is true for* i' *even if Assumption 1 holds only for* $0 \leq i < i'$.

Remark: Observe that the two lower bounds given by Lemma 4.2 for $\mathbb{E}[f(S_I)]/f(OPT)$ coincide.

Proof. Let A_i be an event fixing all the random decisions of Algorithm 3 up to iteration i (including), and let $\mathcal{A}_i$ be the set of all possible A_i events. Fix $1 \leq i \leq k$ and an event $A_{i-1} \in \mathcal{A}_{i-1}$. All the probabilities, expectations and random quantities in the first part of this proof are implicitly conditioned on A_{i-1}.

Next, we transform the set M_i into a set M_i' by adding the missing elements of OPT, and removing elements of $M_i \setminus OPT$ till $|M_i'| = |M_i|$. By submodularity, for $i \leq I$:

$$\begin{aligned}
&\mathbb{E}[f_{u_i}(S_{i-1})] = \frac{1}{\Sigma(i)} \cdot \sum_{u \in M_i} f_u(S_{i-1}) \\
&\geq \frac{1}{\Sigma(i)} \cdot \sum_{u \in M_i'} f_u(S_{i-1}) \\
&\geq \frac{1}{\Sigma(i)} \cdot \left[\sum_{u \in OPT} f_u(S_{i-1}) + (k-i+1)\cdot f_{m_i}(S_{i-1}) \right] \\
&\geq \frac{f(OPT \cup S_{i-1}) - f(S_{i-1})}{\Sigma(i)} + \frac{(k-i+1)\cdot f_{m_i}(S_{i-1})}{\Sigma(i)} ,
\end{aligned}$$

where the first inequality follows by the definition of M_i, and the second one follows since there are at least $k-i+1$ elements in $M_i' \setminus OPT$, and these elements also belong to M_i, and therefore, have a marginal contribution of at least $f_{m_i}(S_{i-1})$ each (notice that $f_{m_i}(S_{i-1}) \geq 0$ due to the application of Reduction 1). Unfixing the event A_{i-1}, and taking the expectation now over all possible such events, we get:

$$\begin{aligned}
\mathbb{E}[f_{u_i}(S_{i-1})] \geq \frac{\mathbb{E}[f(OPT \cup S_{i-1}) - f(S_{i-1})]}{\Sigma(i)} \\
+ \frac{\ell(i)}{\Sigma(i)} \cdot f(OPT) \ .
\end{aligned}$$

It is easy to see that for $i > I$, the same inequality still holds without the last term $(\ell(i)/\Sigma(i))$. Using this observation, let us prove the lemma by induction. For $i = 0$, the lemma holds since:

$$\begin{aligned}
\frac{\mathbb{E}[f(S_0)]}{f(OPT)} \geq 0 > 11.583(1-\pi_0) \\
+ \sigma_0(\pi_0 - 11.33 + 4.5\sigma_0) - \frac{4.5}{4k} \ .
\end{aligned}$$

Assume the lemma holds for every $i' < i$, and let us prove it for $i > 0$. First consider the case $i \leq I$. Clearly,

$$\begin{aligned}
&\frac{\mathbb{E}[f(S_i)]}{f(OPT)} = \frac{\mathbb{E}[f(S_{i-1})]}{f(OPT)} + \frac{\mathbb{E}[f_{u_i}(S_{i-1})]}{f(OPT)} \\
&\geq \frac{\mathbb{E}[f(S_{i-1})]}{f(OPT)} + \frac{\mathbb{E}[f(OPT \cup S_{i-1}) - f(S_{i-1})]}{f(OPT)\cdot\Sigma(i)} + \frac{\ell(i)}{\Sigma(i)} \\
&= \frac{(1-1/\Sigma(i))\cdot\mathbb{E}[f(S_{i-1})]}{f(OPT)} \\
&\quad + \frac{\mathbb{E}[f(OPT \cup S_{i-1})]}{f(OPT)\cdot\Sigma(i)} + \frac{\ell(i)}{\Sigma(i)} \ .
\end{aligned}$$

Since $i \leq I$, we also have $i-1 \leq I$, and therefore, we can lower bound $\frac{\mathbb{E}[f(S_i)]}{f(OPT)}$ using the derivation appearing in Figure 2.

Consider now the case $i > I$. In this case:

$$\begin{aligned}
&\frac{\mathbb{E}[f(S_i)]}{f(OPT)} = \frac{\mathbb{E}[f(S_{i-1})]}{f(OPT)} + \frac{\mathbb{E}[f_{u_i}(S_{i-1})]}{f(OPT)} \\
&\geq \frac{\mathbb{E}[f(S_{i-1})]}{f(OPT)} + \frac{\mathbb{E}[f(OPT \cup S_{i-1}) - f(S_{i-1})]}{f(OPT)\cdot\Sigma(i)} \\
&= \frac{(1-1/\Sigma(i))\cdot\mathbb{E}[f(S_{i-1})]}{f(OPT)} + \frac{\mathbb{E}[f(OPT \cup S_{i-1})]}{f(OPT)\cdot\Sigma(i)} \ .
\end{aligned}$$

Since $i > I$, we have $i-1 \geq I$, and therefore:

$$\begin{aligned}
&\frac{\mathbb{E}[f(S_i)]}{f(OPT)} \geq (1-1/k)\cdot\big\{(1-1/k)^{i-I-2} \\
&\quad [(1-1/k)\cdot V(I) + \pi_I\cdot(i-1-I)/k]\big\} \\
&\quad + \frac{\pi_I\cdot(1-1/k)^{i-1-I}}{k} \\
&= (1-1/k)^{i-I-1}[(1-1/k)\cdot V(I) + \pi_I\cdot(i-I)/k] \ .
\end{aligned}$$

To use the bounds given by Lemma 4.3, we need bounds on π_i and σ_i.

$$\begin{aligned}
\frac{\mathbb{E}\left[f(S_i)\right]}{f(OPT)} &\geq \left(1-\frac{1}{2(k-i+1)}\right)\cdot\left\{11.583(1-\pi_{i-1})+\sigma_{i-1}(\pi_{i-1}-11.33+4.5\sigma_{i-1})-\frac{4.5}{4(k-i+1)}\right\} \\
&\quad+\frac{\pi_{i-1}}{2(k-i+1)}+\frac{0.253-2.33\sigma_{i-1}+4.5\sigma_{i-1}^2}{2(k-i+1)} \\
&\geq 11.583(1-\pi_i)+\sigma_{i-1}(\pi_i-11.33+4.5\sigma_{i-1})+\frac{-11.583+11.33\sigma_{i-1}-4.5\sigma_{i-1}^2}{2(k-i+1)} \\
&\quad+\frac{\pi_{i-1}+0.253-2.33\sigma_{i-1}+4.5\sigma_{i-1}^2}{2(k-i+1)}-\frac{4.5}{4(k-i+1)} \\
&\geq 11.583(1-\pi_i)+\sigma_{i-1}(\pi_i-11.33+4.5\sigma_{i-1})+\frac{\pi_i-11.33+9\sigma_{i-1}}{2(k-i+1)}-\frac{4.5}{4(k-i+1)} \\
&= 11.583(1-\pi_i)+\sigma_i(\pi_i-11.33+4.5\sigma_i)-\frac{4.5}{4}\cdot\left[\frac{1}{(k-i+1)^2}+\frac{1}{k-i+1}\right] \\
&> 11.583(1-\pi_i)+\sigma_i(\pi_i-11.33+4.5\sigma_i)-\frac{4.5}{4(k-i)}\ .
\end{aligned}$$

Figure 2: Lower bound on $\frac{\mathbb{E}[f(S_i)]}{f(OPT)}$ for the case $i \leq I$.

LEMMA 4.4. *For every* $0 \leq i \leq I$*:*

$$-0.5\ln(1-i/(k+1)) \leq \sigma_i \leq -0.5\ln(1-i/k)$$

and

$$\sqrt{1-i/k} \leq \pi_i \leq \sqrt{1-i/(k+1)}\ .$$

Proof. Observe that:

$$\begin{aligned}
\sigma_i &= 0.5\cdot\sum_{j=1}^{i}(k-j+1)^{-1} \\
&\leq 0.5\cdot\int_1^{i+1}\frac{dx}{k-x+1} = -0.5\ln(1-i/k)\ . \\
\sigma_i &= 0.5\cdot\sum_{j=1}^{i}(k-j+1)^{-1} \\
&\geq 0.5\cdot\int_0^{i}\frac{dx}{k-x+1} = -0.5\ln(1-i/(k+1))\ . \\
\pi_i &= \prod_{j=1}^{i}(1-0.5(k-j+1)^{-1}) \\
&\geq \prod_{j=1}^{i}\sqrt{1-(k-j+1)^{-1}} \\
&= \prod_{j=1}^{i}\sqrt{\frac{k-j}{k-j+1}} = \sqrt{1-i/k}\ .
\end{aligned}$$

Let $x_j = (k-j+1)$. For every $j \leq I$ and large enough k, $x_j \geq 1/2$, which implies:

$$(1-0.5/x_j)^2 \leq 1-1/(x_j+1),$$

and therefore,

$$\begin{aligned}
\pi_i &= \prod_{j=1}^{i}(1-0.5(k-j+1)^{-1}) \\
&\leq \prod_{j=1}^{i}\sqrt{1-(k-j+2)^{-1}} \\
&= \prod_{j=1}^{i}\sqrt{\frac{k-j+1}{k-j+2}} = \sqrt{1-i/(k+1)}\ .
\end{aligned}$$

We are now ready to analyze Algorithm 5 under Assumption 1.

COROLLARY 4.1. *Assuming Assumption 1 holds, then Algorithm 5 is a* 0.372*-approximation algorithm.*

Proof. Plugging $I = \lceil 0.21k \rceil$ into the bounds given by Lemma 4.4, we get for large enough k:

$$0.1178 \leq \sigma_I \leq 0.1179\ ,$$

and

$$0.8888 \leq \pi_I \leq 0.8889\ .$$

Therefore:

$$V(I) = 11.583(1-\pi_I)+\sigma_I(\pi_I-11.33+4.5\sigma_I)$$

$$- 4.5/[4(k-I)] > 0.1182 \ ,$$

where the inequality holds for large enough k. Thus,

$$\frac{\mathbb{E}\left[f(S_k)\right]}{f(OPT)} \geq (1-1/k)^{k-I-1}[(1-1/k)\cdot V(I) + \pi_I \cdot (k-I)/k] > 0.372 \ ,$$

where the inequality holds, again, for large enough k.

We can also deduce from Lemma 4.4 the following observation, which is used later on.

COROLLARY 4.2. *For every* $1 \leq i \leq I$, $\ell(i) \geq 0$.

Proof. Observe that for every $0 \leq j \leq I$ and large enough k:

$$\begin{aligned}\sigma_j &\leq -0.5\ln(1-j/k) \leq -0.5\ln(1-I/k) \\ &< -0.5\ln(1-0.22) = -0.5\ln 0.78 < 0.125 \ .\end{aligned}$$

Thus, $\sigma_{i-1} \in [0, 0.125)$. Recall that $\ell(i) = 0.253 - 2.33\sigma_{i-1} + 4.5\sigma_{i-1}^2$, and it can be easily checked that this expression is non-negative in the range $[0, 0.125)$.

Next, we should consider what happens when Assumption 1 does not hold. In this case the following assumption must hold.

ASSUMPTION 2. *There exists a value* $1 \leq i' \leq I$ *for which:*
- *Assumption 1 does not hold for* $i = i'$.
- *Assumption 1 holds for every* $1 \leq i < i'$.

Consider Algorithm 6. This algorithm gets the state of Algorithm 5 before iteration $i = i'$ (where i' is the value whose existence is guaranteed by Assumption 1), and uses it to output a good solution.

The feasibility of the output Z_i of the algorithm follows since $|Z_i| \leq |S_{i-1}| + |B_i| \leq (i-1) + (k-i+1) = k$. Also, observe that g_i is a submodular function, and therefore, we can use in the analysis of Algorithm 6 the approximation ratio guaranteed above for Continuous Double Greedy (Algorithm 2).

LEMMA 4.5. *Assuming Assumption 2 holds, then:*

$$\begin{aligned}\mathbb{E}[g_{i'}(B_{i'})] \geq\ & 0.5(1-(i'-1)/k)\cdot\mathbb{E}[f(S_{i'-1}\cup OPT)] \\ & + 0.25[1+2(i'-1)/k]\cdot\mathbb{E}[f(S_{i'-1})] \\ & - 0.5\ell(i')\cdot f(OPT) \ .\end{aligned}$$

Proof. Let us assume $|OPT| = k$ (if this is not the case, we can add $k - |OPT|$ dummy elements of $D \setminus OPT$ to OPT), and let OPT_i be a random subset of OPT of size $k-i+1$. We would like to prove that $g_{i'}(OPT_{i'} \cap M_{i'})$ is large in expectation. The submodularity of $g_{t'}$ implies:

$$\begin{aligned}g_{i'}(OPT_{i'} \cap M_{i'}) &\geq g_{i'}(OPT_{i'}) + g_{i'}(\varnothing) \\ &\quad - g_{i'}(OPT_{i'} \setminus M_{i'}) \\ &= g_{i'}(OPT_{i'}) - [f((OPT_{i'} \setminus M_{i'}) \cup S_{i'-1})) \\ &\quad - f(S_{i'-1})] \ .\end{aligned}$$

Since assumption 1 does not hold for i', the expected contribution of an element outside of $M_{i'}$ must be less than $\ell(i')/k \cdot f(OPT)$ (since it lower bounds the expected contribution of $m_{i'}$). By Lemma 2.1:

$$\begin{aligned}&\mathbb{E}[f(S_{i'-1} \cup (OPT_{i'} \setminus M_{i'})) - f(S_{i'-1})] \\ &\leq \mathbb{E}\left[\sum_{u \in OPT_{i'} \setminus M_{i'}} f_u(S_{i'-1})\right] \\ &\leq \mathbb{E}\left[\sum_{u \in OPT_{i'} \setminus M_{i'}} [\ell(i')/(k-i'+1)\cdot f(OPT)]\right] \\ &\leq \ell(i')\cdot f(OPT) \ ,\end{aligned}$$

where the last inequality follows due to Corollary 4.2, and the observation $|OPT_{i'} \setminus M_{i'}| \leq |OPT_{i'}| \leq k - i' + 1$. Combining the two above inequalities, we get:

$$\mathbb{E}[g_{i'}(OPT_{i'} \cap M_{i'})] \geq \mathbb{E}[g_{i'}(OPT_{i'})] - \ell(i')\cdot f(OPT) \ .$$

Notice also that $|OPT_{i'} \cap M_{i'}| \leq |OPT_{i'}| \leq k - i + 1$, and therefore, Continuous Double Greedy is guaranteed to outputs a set $B_{i'}$ such that $\mathbb{E}[g_{i'}(B_{i'}))] \geq 0.5 \cdot [\mathbb{E}[g_{i'}(OPT_{i'})] - \ell(i') \cdot f(OPT)] + 0.25 \cdot \mathbb{E}[g_{i'}(\varnothing)] = 0.5 \cdot [\mathbb{E}[g_{i'}(OPT_{i'})] - \ell(i') \cdot f(OPT)] + 0.25 \cdot \mathbb{E}[f(S_{i'-1})]$.

To complete the proof of the lemma, observe that by Lemma 2.1, $\mathbb{E}[g_{i'}(OPT_{i'})] \geq (1-(i'-1)/k) \cdot \mathbb{E}[g_{i'}(OPT)] + (i'-1)/k \cdot \mathbb{E}[g_{i'}(\varnothing)] \geq (1-(i'-1)/k) \cdot \mathbb{E}[f(S_{i'-1} \cup OPT)] + (i'-1)/k \cdot \mathbb{E}[f(S_{i'-1})]$.

COROLLARY 4.3. *Assuming Assumption 2 holds, then:*

$$\begin{aligned}\mathbb{E}[f(Z_{i'})] \geq\ & 0.5(1-(i'-1)/k)\cdot\mathbb{E}[f(S_{i'-1}\cup OPT)] \\ & + 0.25[1+2(i'-1)/k]\cdot\mathbb{E}[f(S_{i'-1})] \\ & - 0.5\ell(i')\cdot f(OPT) \ .\end{aligned}$$

Proof. Follows from Lemma 4.5 since $\mathbb{E}[f(Z_{i'})] = \mathbb{E}[f(S_{i'-1} \cup B_{i'})] = \mathbb{E}[g_{i'}(B_{i'})]$.

Using Corollary 4.3, Lemma 4.2, the second part of Lemma 4.3 and Lemma 4.4 it is strait forward to prove the following lemma.

LEMMA 4.6. *Assuming Assumption 2 holds, then* $\mathbb{E}[f(Z_{i'})] \geq 0.372 \cdot f(OPT)$.

We can now execute both Algorithms 5 and 6 (for every $1 \leq i \leq I$), and output the best solution found. Since (exactly) one of the Assumptions 1 and 2 must hold, we get by Corollary 4.1 and Lemma 4.6 that at least one of the two algorithms must find a solution of value at least $0.372 \cdot f(OPT)$.

Algorithm 6: Augmentation Procedure(f, k, i, S_{i-1})

Let $M_i \subseteq \mathcal{N} \setminus S_{i-1}$ be a subset of size $2(k-i+1)$ maximizing $\sum_{u \in M_i} f_u(S_i)$.
Apply Continuous Double Greedy to find a subset $B_i \subseteq M_i$ of size at most $k-i+1$ maximizing $g_i(B_i) = f(B_i \cup S_{i-1})$.
Let $Z_i \leftarrow S_{i-1} \cup B_i$.
Return Z_i.

COROLLARY 4.4. *There exists a* 0.372*-approximation algorithm for maximizing an independent set under a cardinality constraint.*

Notice that $1/e + 0.004 < 0.372$, and thus, Corollary 4.4 provides an approximation ratio which is as good as the first term in the $\max$ expression of Theorem 1.1. Hence, to prove Theorem 1.1 it is enough to build an algorithm that executes Continuous Double Greedy (Algorithm 2) and the algorithm of Corollary 4.4, and outputs the better solution.

4.3 Fast Algorithms Subject to $|S| = k$ In this section we are interested in analyzing the approximation that can be achieved using Random Greedy for the problem $\max\{f(S) : |S| = k\}$. To do that, we drop the assumption that Reduction 1 was applied, and assume instead that Reduction 2 was applied. Looking again at the pseudo-code of Random Greedy (Algorithm 1), one can observe that it always selects exactly k elements. However, the analysis of the algorithm in Section 3.1 assumed Reduction 1 was applied, and therefore, the output of the algorithm could contain strictly less than k elements. On the other hand, under Reduction 2 the output of Random Greedy is guaranteed to be of size exactly k.

The analysis of Random Greedy in Section 3.1 for non-monotone objectives begins with Observation 1 which lower bounds the expected value of $f(OPT \cup S_i)$. We notice that the proof of this observation does not use anything from Reduction 1, except for the guarantee $2k \leq n$. As this guarantee also follows from Reduction 2, Observation 1 still holds in our current setting. The following observation gives an upper bound on $f(OPT \cup S_i)$. Let A_i be an event fixing all the random decisions of the algorithm in the first i iterations, and let $\mathcal{A}_i$ be the set of all possible A_i events.

OBSERVATION 6. *For every* $0 \leq i \leq k$*, given any event* $A_i \in \mathcal{A}_i$*,* $f(OPT \cup S_i) \leq 2f(OPT)$.

Proof. Fix the event A_i. All the probabilities, expectations and random quantities in this proof are implicitly conditioned on A_i. Observe that once we fix A_i, the set S_i becomes constant. Let B_i be an arbitrary set containing $k-i$ elements of $OPT \setminus S_i$ (there exists such a set because $|OPT| = k$ and $|S_i| = i$). Then,

$$2f(OPT) \geq f(OPT) + f(S_i \cup B_i) \geq f(OPT \cup S_i \cup B_i) = f(OPT \cup S_i) \ ,$$

where the first inequality follows from the definition of OPT and the second from submodularity and non-negativity.

Let us now consider the series of random variables $X_1, X_2, \ldots, X_k$, where $X_i = |OPT \setminus S_i|$.

OBSERVATION 7. *For every* $1 \leq i \leq k$*, and event* $A_{i-1} \in \mathcal{A}_{i-1}$, $\Pr[X_i = X_{i-1} - 1 \mid A_{i-1}] \leq \frac{\mathbb{E}[X_{i-1} \mid A_{i-1}]}{k}$.

Proof. The set M_{i-1} contains k elements of $\mathcal{N} \setminus S_{i-1}$, and at most X_{i-1} of these belong to OPT. Therefore:

$$\Pr[X_i = X_{i-1} - 1 \mid A_{i-1}] = \Pr[u_i \in OPT \mid A_{i-1}] \leq \frac{\mathbb{E}[X_{i-1} \mid A_{i-1}]}{k} \ .$$

The next lemma bounds the expected difference between the values of S_{i-1} and S_i. Denote the term $2 \cdot \mathbb{E}[\max\{k(1-1/k)^i - X_i, 0\}] \cdot f(OPT)/(nk)$ by E_i.

LEMMA 4.7. *For every* $1 \leq i \leq k$*:*

$$\mathbb{E}[f_{u_i}(S_{i-1})] \geq \frac{[n/k - 1 + (1-1/k)^{i-1}] \cdot (1-1/k)^{i-1} \cdot f(OPT)}{n} - \frac{\mathbb{E}[f(S_{i-1})]}{k} - E_{i-1} \ .$$

Proof. Fix an event $A_{i-1} \in \mathcal{A}_i$. All the probabilities, expectations and random quantities in the first part of this proof are implicitly conditioned on A_{i-1}. Let us construct a set M_i' as following. M_i' contain the X_{i-1} elements of $OPT \setminus S_{i-1}$ plus $k - X_{i-1}$ uniformly random elements from $\mathcal{N} \setminus (OPT \cup S_{i-1})$. By Lemma 2.1:

$$\mathbb{E}[f(S_{i-1} \cup M_i')] \geq \frac{k - X_{i-1}}{n} \cdot f(\mathcal{N}) + \frac{n - k + X_{i-1}}{n} \cdot f(S_{i-1} \cup OPT) \geq \frac{n - k + X_{i-1}}{n} \cdot f(S_{i-1} \cup OPT) \ .$$

By the definition of M_i:

$$\mathbb{E}[f_{u_i}(S_{i-1})] = k^{-1} \cdot \sum_{u \in M_i} f_u(S_{i-1})$$

$$\geq k^{-1}\cdot \sum_{u\in M_i'} f_u(S_{i-1}) \geq k^{-1}\cdot[f(S_{i-1}\cup M_i') - f(S_{i-1})]$$
$$\geq k^{-1}\cdot\left[\frac{n-k+X_{i-1}}{n}\cdot f(S_{i-1}\cup OPT) - f(S_{i-1})\right]$$
$$\geq k^{-1}\cdot\left[\frac{n-k+k(1-1/k)^i}{n}\cdot f(S_{i-1}\cup OPT) - f(S_{i-1})\right]$$
$$- \frac{\max\{k(1-1/k)^i - X_{i-1}, 0\}}{nk}\cdot f(S_{i-1}\cup OPT)$$
$$\geq k^{-1}\cdot\left[\frac{n-k+k(1-1/k)^i}{n}\cdot f(S_{i-1}\cup OPT) - f(S_{i-1})\right]$$
$$- \frac{2\cdot\max\{k(1-1/k)^i - X_{i-1}, 0\}}{nk}\cdot f(OPT)\ ,$$

where the last inequality follows from Observation 6. Unfixing the event A_i, and taking now the expectation over all the random choices of the algorithm, we get:

$$\mathbb{E}[f_{u_i}(S_{i-1})]$$
$$\geq k^{-1}\cdot\left[\frac{n-k+k(1-1/k)^{i-1}}{n}\cdot\mathbb{E}[f(S_{i-1}\cup OPT)] - \mathbb{E}[f(S_{i-1})]\right]$$
$$- \frac{2\cdot\mathbb{E}[\max\{k(1-1/k)^i - X_{i-1}, 0\}]}{nk}\cdot f(OPT)$$
$$\geq \frac{[n/k - 1 + (1-1/k)^{i-1}]\cdot(1-1/k)^{i-1}\cdot f(OPT)}{n}$$
$$- \frac{\mathbb{E}[f(S_{i-1})]}{k} - E_{i-1}\ ,$$

where the last inequality follows from Observation 1.

By repeated applications of the last lemma, it is possible to prove the next lower bound on $\mathbb{E}[f(S_i)]$. Due to space constraints, we omit the formal, induction based, proof of the next lemma from this extended abstract.

LEMMA 4.8. *For every* $0 \leq i \leq k$, $\mathbb{E}[f(S_i)] \geq (1 - 1/k)^{i-1}[k/n + (1-k/n)\cdot i/k - (1-1/k)^i\cdot k/n]\cdot f(OPT) - \sum_{j=0}^{i-1} E_j$.

The lower bound given by Lemma 4.8 includes the error term $\sum_{j=0}^{i-1} E_j$ which is upper bounded by the following lemma. Due to space limitations, we omit the proof of this lemma from this extended abstract.

LEMMA 4.9. $\sum_{i=0}^{k-1} E_i \leq O(k^{2/3}n^{-1})\cdot f(OPT) = O(k^{-1/3})\cdot f(OPT)$.

Combining the last lemma with Lemma 4.8, we get $\mathbb{E}[f(S_k)] \geq e^{-1}[1 - e^{-1}k/n]\cdot f(OPT) - O(k^{-1/3})\cdot f(OPT)$, which proves that Random Greedy provides the approximation ratio guaranteed by Theorem 1.4. Moreover, this approximation ratio is also equal to the first term in the max expression of Theorem 1.2. Hence, to prove Theorem 1.2 it is enough to build an algorithm that executes Random Greedy and Continuous Double Greedy (Algorithm 2), and outputs the better solution.

Acknowledgments

The authors would like to thank Jeff Bilmes for introducing them to some of the applications mentioned in this work.

References

[1] A. A. Ageev and M. I. Sviridenko. An 0.828 approximation algorithm for the uncapacitated facility location problem. *Discrete Appl. Math.*, 93:149–156, July 1999.

[2] Alexander Ageev, Refael Hassin, and Maxim Sviridenko. A 0.5-approximation algorithm for max dicut with given sizes of parts. *SIAM J. Discret. Math.*, 14(2):246–255, February 2001.

[3] Alexander A. Ageev and Maxim Sviridenko. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In *IPCO*, pages 17–30, 1999.

[4] Alexander A. Ageev and Maxim Sviridenko. An approximation algorithm for hypergraph max k-cut with given sizes of parts. In *ESA*, pages 32–41, 2000.

[5] Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better balance by being biased: A 0.8776-approximation for max bisection. In *SODA*, pages 277–294, 2013.

[6] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *ICCV*, volume 1, pages 105–112, 2001.

[7] Richard A. Brualdi. Comments on bases in dependence structures. *Bull. of the Australian Math. Soc.*, 1(02):161–167, 1969.

[8] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization, 2012. FOCS 2012.

[9] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[10] Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.*, 35(3):713–728, September 2005.

[11] Reuven Cohen, Liran Katzir, and Danny Raz. An efficient approximation for the generalized assignment problem. *Information Processing Letters*, 100(4):162–166, 2006.

[12] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Management Sciences*, 23:789–810, 1977.

[13] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. On the uncapacitated location problem. *Annals of Discrete Mathematics*, 1:163–177, 1977.

[14] Shaddin Dughmi, Tim Roughgarden, and Mukund Sundararajan. Revenue submodularity. In *EC*, pages 243–252, 2009.

[15] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[16] Uriel Feige and Michel X. Goemans. Aproximating the value of two prover proof systems, with applications to max 2sat and max dicut. In *ISTCS*, pages 182–189, 1995.

[17] Uriel Feige and Michael Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41(2):174–211, 2001.

[18] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[19] Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of $1 - 1/e$. In *FOCS*, pages 667–676, 2006.

[20] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, 2011.

[21] Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 659–668, 2012.

[22] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions – ii. In *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 73–87. Springer Berlin Heidelberg, 1978.

[23] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *SODA*, pages 611–620, 2006.

[24] Alan M. Frieze and Mark Jerrum. Improved approximation algorithms for max k-cut and max bisection. In *IPCO*, pages 1–13, 1995.

[25] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *SODA*, pages 1098–1117, 2011.

[26] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[27] Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *SODA*, pages 1–7, 2001.

[28] Jason Hartline, Vahab Mirrokni, and Mukund Sundararajan. Optimal marketing strategies over social networks. In *WWW*, pages 189–198, 2008.

[29] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, July 2001.

[30] S. Jegelka and J. Bilmes. Submodularity beyond submodular energies: Coupling edges in graph cuts. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 0:1897–1904, 2011.

[31] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[32] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, pages 137–146, 2003.

[33] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM J. Comput.*, 37:319–357, April 2007.

[34] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[35] Andreas Krause, AjitSingh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9:235–284, January 2008.

[36] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI*, page 5, 2005.

[37] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, pages 1650–1654. AAAI Press, 2007.

[38] Andreas Krause, Jure Leskovec, Carlos Guestrin, Jeanne VanBriesen, and Christos Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, November 2008.

[39] Eugene Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rhinehart and Winston, New York, NY, USA, 1976.

[40] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing non-monotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078, 2010.

[41] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *North American chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL/HLT-2010)*, Los Angeles, CA, June 2010.

[42] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *HLT*, pages 510–520, 2011.

[43] M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. In *FOCS*, pages 248–255, 2004.

[44] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3):177–188, 1978.

[45] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14:265–294, 1978.

[46] A. Schrijver. *Combinatorial Optimization: Polyhedra and Eciency*. Springer, 2003.

[47] Alexander Schrijver. *Combinatorial Optimization, Polyhedra and Efficiency*. Springer, 2004.

[48] Andreas S. Schulz and Nelson A. Uhan. Approximating the least core value and least core of cooperative games with supermodular costs. *Discrete Optimization*, 10(2):163–180, 2013.

[49] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *NIPS*, pages 1577–1584, 2008.

[50] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. *SIAM J. Comput.*, 29:2074–2097, April 2000.

[51] Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *FOCS*, pages 651–670, 2009.