A Unified Continuous Greedy Algorithm for Submodular Maximization

(Extended Abstract)

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

Abstract

The study of combinatorial problems with a submodular objective function has attracted much attention in recent years, and is partly motivated by the importance of such problems to economics, algorithmic game theory and combinatorial optimization. Classical works on these problems are mostly combinatorial in nature. Recently, however, many results based on continuous algorithmic tools have emerged. The main bottleneck of such continuous techniques is how to approximately solve a non-convex relaxation for the submodular problem at hand. Thus, the efficient computation of better fractional solutions immediately implies improved approximations for numerous applications. A simple and elegant method, called "continuous greedy", successfully tackles this issue for monotone submodular objective functions, however, only much more complex tools are known to work for general non-monotone submodular objectives.

In this work we present a new unified continuous greedy algorithm which finds approximate fractional solutions for both the non-monotone and monotone cases, and improves on the approximation ratio for many applications. For general non-monotone submodular objective functions, our algorithm achieves an improved approximation ratio of about $1/e$. For monotone submodular objective functions, our algorithm achieves an approximation ratio that depends on the density of the polytope defined by the problem at hand, which is always at least as good as the previously known best approximation ratio of $1-1/e$. Some notable immediate implications are an improved $1/e$-approximation for maximizing a non-monotone submodular function subject to a matroid or $O(1)$-knapsack constraints, and information-theoretic tight approximations for Submodular Max-SAT and Submodular Welfare with k players, for *any* number of players k.

A framework for submodular optimization problems, called the *contention resolution framework*, was introduced recently by Chekuri et al. The improved approximation ratio of the unified continuous greedy algorithm implies improved approximation ratios for many problems through this framework. Moreover, via a parameter called *stopping time*, our algorithm merges the relaxation solving and re-normalization steps of the framework, and achieves, for some applications, further improvements. We also describe new monotone balanced contention resolution schemes for various matching, scheduling and packing problems, thus, improving the approximations achieved for these problems via the framework.

*Computer Science Dept., Technion, Haifa, Israel. e-mail: moranfe@cs.technion.ac.il

†Computer Science Dept., Technion, Haifa, Israel. e-mail: naor@cs.technion.ac.il

‡Computer Science Dept., Technion, Haifa, Israel. e-mail: schwartz@cs.technion.ac.il

1 Introduction

The study of combinatorial problems with submodular objective functions has attracted much attention recently, and is motivated by the principle of economy of scale, prevalent in real world applications. Moreover, submodular functions are commonly used as utility functions in economics and algorithmic game theory. From a theoretical perspective, submodular maximization plays a major role in combinatorial optimization since many optimization problems can be represented as constrained variants of submodular maximization. Two such well studied problems are Max-Cut and Max-k-Cover [23, 24, 27, 29, 30, 32, 41, 42, 47]. In this paper, we consider the basic problem of maximizing a non-negative submodular function[1] $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$ over a ground set $\mathcal{N}$ under the constraint that the solution must belong to a down-monotone family[2] of subsets $\mathcal{I} \subseteq 2^{\mathcal{N}}$. This basic (constrained) submodular maximization problem generalizes the above two mentioned classic combinatorial optimization problems.

The techniques used to compute approximate solutions to various (constrained) submodular maximization problems can be partitioned into two main approaches. The first approach is combinatorial in nature, and is mostly based on local search techniques and greedy rules. This approach has been used as early as the late 70's for maximizing monotone submodular functions under the constraint that the solution should be an independent set of one of several specific matroids [13, 21, 25, 26, 28, 33, 43, 44]. Lately, this approach has been extended to include both non-monotone submodular objective functions [17, 20, 22, 49] and additional constraint sets $\mathcal{I}$ [38] (*e.g.*, independent sets of matroids intersection). Though for some problems this approach yields the current state of the art solutions [38], or even tight results [46], these solutions are usually tailored for the specific structure of the problem at hand, making extensions quite difficult.

The second approach for approximating (constrained) submodular maximization problems overcomes the above obstacle. This approach resembles a common paradigm for designing approximation algorithms and is composed of two steps. In the first step, a fractional solution is found for a relaxation of the problem. In the second step, the fractional solution is rounded to obtain an integral one while incurring only a small loss in the objective. This approach has been used to obtain improved approximations to various problems [8,10–12,35,37]. Most notable of these results is an *asymptotically tight* approximation for maximizing a monotone submodular function given a single matroid constraint [8, 43, 44]. Two issues arise when using this approach. First, since the objective function is not linear, it is not clear how to formulate a relaxation which can be solved or even approximated efficiently. Second, given a fractional solution, one needs a rounding procedure which outputs an integral solution without losing too much in the objective function.

Let us elaborate on the first issue, namely how to find good fractional solutions to (constrained) submodular maximization problems. The standard relaxation for such a problem has a variable for every element of the groundset $\mathcal{N}$ taking values from the range $[0, 1]$. As with linear programming relaxations, the family $\mathcal{I}$ is replaced by a set of linear inequality constraints on the variables which define a down-monotone polytope[3] $\mathcal{P}$. Unlike the linear case, the formulation of an objective function for the relaxation is not obvious. A good objective function is a continuous extension of the given integral objective f which allows for efficient computation of a good fractional solution. The extension commonly used to overcome this difficulty, in the context of (constrained) submodular maximization problems, is the *multilinear extension* of f, denoted by F. The multilinear extension $F(x)$ for any $x \in [0, 1]^{\mathcal{N}}$ is the expected value of f over a random subset $R(x) \subseteq \mathcal{N}$. Each element

[1] A function $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ is *submodular* if for every $A \subseteq B \subseteq \mathcal{N}$ and $e \in \mathcal{N}$: $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$. An equivalent definition is that for every $A, B \subseteq \mathcal{N}$: $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.

[2] A family of subsets $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is *down-monotone* if $B \in \mathcal{I}$ and $A \subseteq B$ imply $A \in \mathcal{I}$. Note that many natural families of subsets $\mathcal{I}$ are down-monotone, *e.g.*, families induced by matroid and knapsack constraints.

[3] A polytope $\mathcal{P} \subseteq [0, 1]^{\mathcal{N}}$ is *down-monotone* if $x \in \mathcal{P}$ and $0 \leq y \leq x$ imply $y \in \mathcal{P}$.

$e \in \mathcal{N}$ is chosen independently to be in $R(x)$ with probability x_e. Formally, for every $x \in [0,1]^{\mathcal{N}}$, $F(x) \triangleq \mathbb{E}[R(x)] = \sum_{S \subseteq \mathcal{N}} f(S) \prod_{e \in S} x_e \prod_{e \notin S} (1 - x_e)$. Such relaxations are very common since first introduced by [7] (see [8, 37, 48, 49] for several additional examples).

Even though the objective function defined by the multilinear extension is neither convex nor concave, it is still possible to efficiently compute an approximate feasible fractional solution for the relaxation, assuming its feasibility polytope $\mathcal{P}$ is down monotone and solvable[4]. The first method proposed for computing such a solution is the *continuous greedy* method [7]. It is simple and quick, and its analysis is rather short and intuitive. However, it is only known to work for the multilinear extensions of *monotone*[5] submodular functions f. For non-monotone functions f and specific polytopes, other methods are known for solving the multilinear extension, *e.g.*, for a constant number of knapsack constraints [37] and for a single matroid [22, 49]. These methods use extensions of the local search approach, as opposed to the simple continuous greedy method, making the analysis quite involved. Recently, three algorithms for the non-monotone case and general down-monotone solvable polytopes were suggested by [12]. Similarly to [22, 37], these three algorithms are also based on extensions of the local search approach. The best of the three (with respect to its approximation guarantee) uses a simulated annealing technique [22]. Therefore, these algorithms, and especially the best of the three, have quite a complex analysis.

1.1 Our Results

We present a new unified continuous greedy algorithm which finds approximate fractional solutions for both the non-monotone and monotone cases, and improves on the approximation ratio for many applications. For general non-monotone submodular objective functions, our algorithm achieves an improved approximation ratio of about $1/e$. For monotone submodular objective functions, our algorithm achieves an approximation ratio that depends on the density of the polytope defined by the problem at hand, which is always at least as good as the previously known best approximation ratio of $1 - 1/e$. Some notable immediate applications are an improved $1/e$-approximation for maximizing a non-monotone submodular function subject to a matroid or $O(1)$-knapsack constraints, and information-theoretic tight approximations for Submodular Max-SAT and Submodular Welfare with k players, for *any* number of players k.

It turns out that the unified continuous greedy algorithm works very well with a framework presented by [12] for solving submodular optimization problems via a relaxation. Naïvely plugging our algorithm into the framework, immediately produces improved results due to its approximation ratio. Moreover, we prove that a careful use of our algorithm can further improve the framework's performance. We also show how to extend the framework to various scheduling, matching and packing problems, thus, improving upon the current best known results for these problems.

1.1.1 Measured Continuous Greedy

Though our algorithm is quite intuitive, it is based on a simple but crucially useful insight on which we now elaborate. The continuous greedy algorithm of [8] starts with an empty solution and at each step moves by a small δ in the direction of a feasible point $x \in \mathcal{P}$. Let y be the current position of the algorithm. Then x is chosen greedily (hence the name "continuous greedy") by solving $x = \operatorname{argmax} \{w(y) \cdot x \mid x \in \mathcal{P}\}$ where the weight vector $w(y) \in \mathbb{R}^{\mathcal{N}}$ is $w(y)_e = F(y \vee \mathbf{1}_e) - F(y)$, for every $e \in \mathcal{N}$. Thus, x is chosen according to the *residual increase* of each element e, *i.e.*, $F(y \vee \mathbf{1}_e) - F(y)$. However, one would intuitively expect that the step should be chosen according

[4] A polytope $\mathcal{P}$ is *solvable* if linear functions can be maximized over it in polynomial time. Using the ellipsoid algorithm, one can prove $\mathcal{P}$ is solvable by giving a polynomial-time algorithm that given x determines whether $x \in \mathcal{P}$.

[5] A function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is *monotone* if $A \subseteq B \subseteq \mathcal{N}$ implies $f(A) \leq f(B)$.

to the *gradient* of $F(y)$. Our unified algorithm compensates for the difference between the residual increase of elements at point y, and $\nabla F(y)$, by *distorting* the direction x as to mimic the value of $\nabla F(y)$. This is done by decreasing x_e, for every $e \in \mathcal{N}$, by a multiplicative factor of $1 - y_e$. Therefore, our unified continuous greedy algorithm is called *measured continuous greedy*.

The measured continuous greedy algorithm, unlike local search based algorithms, has a parameter called *stopping time*. The stopping time controls a tradeoff between two important properties of the fractional solution found by the algorithm. The first property is the value of the solution: a larger stopping time implies a better fractional solution. The second property is how much slack does the fractional solution has: a smaller stopping time implies more slack (refer to Section 1.1.3 for uses of the second property).

For monotone submodular objectives, the dependance of the approximation ratio on the stopping time T is identical for both our algorithm and the continuous greedy algorithm of [8]. This is somewhat counter intuitive, since our algorithm makes a "smaller" step in each iteration (recall that the movement in direction e is reduced by a multiplicative factor of $1 - y_e$). This seems to suggest that the known continuous greedy algorithm is a bit wasteful. The smaller steps of our algorithm prevent this waste, keep its fractional solution within the polyope for a longer period of time, and thus, allow the use of larger stopping times in some settings.

The following two theorems quantify the guaranteed performance of the measured continuous greedy algorithm for non-monotone and monotone submodular functions. We denote by OPT the optimal integral solution. Note that the first claim of Theorem 1.2, $x/T \in \mathcal{P}$, repeats, in fact, the guarantee of the continuous greedy algorithm of [8]. However, the second claim of the same theorem enables us to obtain improved approximation guarantees for several well studied problems. This property states that one can use stopping times larger than 1. The maximal stopping time that can be used depends on the density of the underlaying polytope[6] (notice that the density resembles the width parameter used by [3]).

Theorem 1.1. *For any non-negative submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, down-monotone solvable polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ and stopping time $T \in [0,1]$, the* measured continuous greedy *algorithm finds a point $x \in [0,1]^{\mathcal{N}}$ such that $F(x) \geq [Te^{-T} - o(1)] \cdot f(OPT)$ and $x/T \in \mathcal{P}$.*

Theorem 1.2. *For any normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, down-monotone solvable polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ and stopping time $T \geq 0$, the* measured continuous greedy *algorithm finds a point $x \in [0,1]^{\mathcal{N}}$ such that $F(x) \geq [1 - e^{-T} - o(1)] \cdot f(OPT)$. Additionally,*

1. *$x/T \in \mathcal{P}$.*
2. *Let $T_{\mathcal{P}} = -\ln(1 - d(\mathcal{P}) + n\delta)/d(\mathcal{P})$. Then, $T \leq T_{\mathcal{P}}$ implies $x \in \mathcal{P}$.*

Theorem 1.2 gives an approximation ratio of $1 - e^{-T_{\mathcal{P}}} \approx 1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})}$. In some cases one can get a cleaner approximation ratio of exactly $1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})}$ by guessing the most valuable single element of OPT (the technique of guessing the most valuable single element of OPT is not new, and can be found, *e.g.*, in [8]). The following theorem examplifies that. A *binary* polytope $\mathcal{P}$ is a polytope defined by constraints with only $\{0,1\}$ coefficients.

Theorem 1.3. *Given a binary down-monotone solvable polytope $\mathcal{P}$ with a bounded $T_{\mathcal{P}}$ and a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, there is a polynomial time algorithm outputting a point $x \in \mathcal{P}$ with $F(x) \geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})}] \cdot f(OPT)$.*

1.1.2 Main Applications

Theorems 1.1, 1.2 and 1.3 immediately provide improved approximations for various problems. We elaborate now on a few of these, starting with the non-monotone case. Theorem 1.1, gives an

[6] Let $\sum_{e \in \mathcal{N}} a_{i,e} x_e \leq b_i$ denote the i^{th} inequality constraint of the polytope. The *density* of $\mathcal{P}$ is defined by: $d(\mathcal{P}) = \min_{1 \leq i \leq m} \frac{b_i}{\sum_{e \in \mathcal{N}} a_{i,e}}$. See Section 2 and Appendix A for details.

improved $(1/e - o(1))$-approximation for finding a fractional solution for any down-monotone and solvable polytope $\mathcal{P}$. Examples of some well-studied problems for which this provides improved approximation are maximization of a non-monotone submodular function f over a single matroid [12, 22, 49] and over $O(1)$ knapsack constraints [12, 34, 37]. For both we provide an improved approximation of about $1/e$. Note that both problems are known to have an approximation of roughly ≈ 0.325 [12] via the simulated annealing technique of [22].

For the monotone case, Theorems 1.2 and 1.3 are used to immediately obtain improved approximations for various problems. Most notable is the well studied Submodular Welfare problem (refer to [8, 9, 14–16, 18, 19, 31, 39, 45, 48, 49] for previous results on Submodular Welfare and additional closely related variants of the problem). The above theorems provide *tight* approximations for *any* number of players k, which exactly matches the $(1 - (1 - 1/k)^k)$-hardness result [48, 49]. This improvement is most significant for small values of k. Another problem we consider is Submodular Max-SAT. Submodular Max-SAT is a generalization of both Max-SAT and Submodular Welfare with two players, in which a monotone submodular function f is defined over the clauses of a CNF formula, and the goal is to find an assignment maximizing the value of f over the set of satisfied clauses. For Submodular Max-SAT we get a 3/4 approximation. The above main applications are summarized in Table 1.

Table 1: Main applications.

Problem \ Constraint	**This Paper**	**Previous Result**	**Hardness***
Matroid (non-monotone)	$1/e - o(1)$	≈ 0.325 [12]	≈ 0.478 [22]
$O(1)$-Knapsacks (non-monotone)	$1/e - \varepsilon$	≈ 0.325 [12]	$1/2$**
Submodular Welfare (k players)	$1 - (1 - 1/k)^k$	$\max\left\{1 - \frac{1}{e}, \frac{k}{(2k-1)}\right\}$ [8, 15]	$1 - (1 - 1/k)^k$ [49]
Submodular Max-SAT	3/4	2/3*** [2]	3/4 [49]

* All hardness results are for the value oracle model, and are information theory based.
** Can be derived from the method of [49].
*** The results of [2] were achieved independently of ours.

1.1.3 Framework Extension

As previously mentioned, though the rounding approach to (constrained) submodular maximization problems is flexible, there are two issues that need addressing. The first one is to approximately solve a relaxation for the problem, and the other is to round the fractional solution. Building upon [3], [12] proposes a general *contention resolution framework* for rounding fractional solutions. Intuitively, the scheme works as follows. First, an approximate fractional solution x is found for the multilinear extension relaxation. Second, x is re-normalized, and a random subset of elements is sampled according to probabilities determined by x. Third and last, some of the sampled elements are discarded to ensure the feasibility of the solution.

The first step can be performed by any algorithm for finding approximate fractional solutions for the multilinear relaxation. Let α be its approximation guarantee. The re-normalization factor and the decision which elements to discard are determined by a constraint specific *contention resolution scheme*. Formally, a (b, c)-balanced contention resolution scheme for a constraint family $\mathcal{I}$ is an algorithm that gets a vector $x \in b\mathcal{P}(\mathcal{I})$ (where $\mathcal{P}(\mathcal{I})$ is the convex hull of $\mathcal{I}$), picks a random set $R(x)$ according to probabilities determined by x, and then outputs a set $S \in \mathcal{I}$ obeying $\Pr[e \in S | e \in R(x)] \geq c$ for every $e \in \mathcal{N}$. If the contention resolution scheme is monotonic, *i.e.*, the probability of e to be in S only increases when elements are removed from $R(x)$, then the framework guarantees αbc approximation for maximizing a submodular function subject to the constraint family $\mathcal{I}$. One advantage of this framework is the ease by which it deals with intersections

of constraints of different types (*e.g.*, matroids, knapsack constraints and matchoids).

We extend the framework of [12] by showing that finding a fractional solution for the relaxation and the re-normalization step, can both be done simultaneously using the measured continuous greedy algorithm. Equipped with this observation, we can replace the expression αbc for the approximation ratio with an improved one for both the non-monotone and the monotone cases. The improvement achieved by the new expression is most significant for small values of b, as can be seen by applications such as *k-sparse packing*.

The idea behind our method is to use b as the stopping time of Theorems 1.1 and 1.2, hence, directly getting a re-normalized fractional solution (as both theorems ensure). The following theorem presents the improved expressions for the approximation ratio, and its proof appears in Appendix F.

Theorem 1.4. *If there is a monotone (b,c)-balanced contention resolution scheme for $\mathcal{I}$, then there is an approximation of $\left(e^{-b}bc - o(1)\right)$ for $\max_{S\in\mathcal{I}}\{f(S)\}$ assuming f is non-negative and submodular, and an approximation of $\left((1-e^{-b})c - o(1)\right)$ assuming f is monotone.*

Note that the results of Theorem 1.4 are better then the (αbc)-approximation of [12]. This is true, since for the non-monotone case $e^{-b} > 0.325$ for every $b \in (0,1]$, and for the monotone case $1 - e^{-b} \geq (1-1/e)b$ for every $b \in (0,1]$.

For example, consider the *k-sparse packing* problem presented by [3], who provided an approximation of $(e-1)/(e^2 \cdot k) - o(1)$ in case f is monotone. Using Theorem 1.4 we can improve this approximation factor and obtain a guarantee of $1/(e \cdot k) - o(1)$ (details are deferred for a full version of this paper). In fact, we are able to improve the approximation guarantee for any constraint family $\mathcal{I}$ which contains, for example, constraints for the intersection of k matroids or k-matchoids. Additional details regarding such problems appear in Appendix F.

We also note that a simple proof of the framework which does not use the FKG inequality, but rather relays on a coupling argument of random subsets, can be presented. We defer details of this proof for a full version of this paper.

1.1.4 Balanced Contention Resolution Schemes

We provide monotone balanced contention resolution schemes for various matching, scheduling and packing problems. Using these schemes and Theorem 1.4, we are able to improve the known approximation ratios for these problems. A comprehensive list of our schemes and the problems for which they provide an improvement appears in Appendix G. Among the results of Appendix G, there are two that are especially notable:

- For job interval scheduling with k-identical machines, and a *linear* objective function, we get an approximation ratio approaching 1 for large values of k. The previously best approximation ratio for this problems approaches $1 - e^{-1}$ for large k's [6].
- For broadcast scheduling with a monotone submodular objective function, we get an approximation ratio of $1/4$. This matches the best known approximation for the linear variant [5].

2 Preliminaries

In addition to the multilinear extension, we make use of the Lovász extension (introduced in [40]). Let $T_\lambda(z)$ be the set of elements whose coordinate in z is at least λ. The Lovász extension of a submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}$ is defined as $\hat{f}(x) = \int_0^1 f(T_\lambda(x))d\lambda$. This definition can also be interpreted in probabilistic terms as the expected value of f over the set $T_\lambda(x)$, where $\lambda \sim Unif[0,1]$. Beside its use in relaxations for minimization problems, the Lovász extension can also be used to lower bound the multilinear extension via the following theorem.

Theorem 2.1 (Lemma A.4 in [49])**.** *Let $F(x)$ and $\hat{f}(x)$ be the multilinear and Lovász extensions, respectively, of a submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}$. Then, $F(x) \geq \hat{f}(x)$ for every $x \in [0,1]^{\mathcal{N}}$.*

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, $(x \vee y)_e = \max\{x_e, y_e\}$ and $(x \wedge y)_e = \min\{x_e, y_e\}$). We also make use of the notation $\partial_e F(x) = F(x \vee \mathbf{1}_e) - F(x \wedge \mathbf{1}_{\bar{e}})$, where $\mathbf{1}_e$ and $\mathbf{1}_{\bar{e}}$ are the characteristic vectors of the sets $\{e\}$ and $\mathcal{N} - \{e\}$, respectively. The multilinear nature of F yields the following useful observation, relating these terms to each other.

Observation 2.2. *Let $F(x)$ be the multilinear extension of a submodular function $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$. Then, for every $e \in \mathcal{N}$,*

$$\partial_e F(x) = \frac{F(x \vee \mathbf{1}_e) - F(x)}{1 - x_e} = \frac{F(x) - F(x \wedge \mathbf{1}_{\bar{e}})}{x_e} \ .$$

Consider a down-monotone polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ defined by positive sign constraints ($x \geq 0$) and additional m inequality constraints. Let $\sum_{e \in \mathcal{N}} a_{i,e} x_e \leq b_i$ denote the i^{th} inequality constraint. The *density* of $\mathcal{P}$ is defined by: $d(\mathcal{P}) = \min_{1 \leq i \leq m} \frac{b_i}{\sum_{e \in \mathcal{N}} a_{i,e}}$. Since $\mathcal{P}$ is a down monotone polytope within the hypercube $[0,1]^{\mathcal{N}}$, one can assume all coefficients $a_{i,e}$ and b_i are non-negative, and $0 < d(\mathcal{P}) \leq 1$. See Appendix A for details.

Given a matroid $M = (\mathcal{N}, \mathcal{I})$, its matroid polytope $\mathcal{P}(M)$ is the convex-hall of all its independent sets. The polytope $\mathcal{P}(M)$ is down-monotone since the family of independent sets $\mathcal{I}$ is down-monotone. Also, $\mathcal{P}(M)$ is solvable because the greedy algorithm can be used to maximize a linear function over $\mathcal{P}(M)$. The following theorem shows that it is possible to round a fractional point in $\mathcal{P}(M)$ without any loss, even when the objective function is a general non-negative submodular function. This theorem is originally based on the method of [1].

Theorem 2.3 (Lemma A.8 in [49])**.** *Given a matroid $M = (\mathcal{N}, \mathcal{I})$, a point $x \in \mathcal{P}(M)$ and a submodular function $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$ with its multilinear extension $F : [0,1]^{\mathcal{N}} \rightarrow \mathbb{R}^+$. There is a polynomial time algorithm, called* pipage rounding*, outputting a random independent set $S \in \mathcal{I}$ such that $\mathbb{E}[f(S)] \geq F(x)$.*

The explicit representation of both submodular functions and matroids might be exponential in the size of their ground set. The standard way to bypass this difficulty is to assume access to these objects via oracles. For a submodular function $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$, given a set $S \subseteq \mathcal{N}$, the oracle returns the value of $f(S)$.[7] For a matroid $M = (\mathcal{N}, \mathcal{I})$, given a set $S \subseteq \mathcal{N}$, the oracle answers whether $S \in \mathcal{I}$.

In some proofs we use the value W defined as $W = n \cdot \max_{e \in \mathcal{N}} f(e)$. We assume throughout the paper that $\{e\} \in \mathcal{I}$ for every element $e \in \mathcal{N}$. Any element which violates this assumption belongs to a non-feasible solution, and can be removed. It is clear that this assumption implies $f(S) \leq W \leq n \cdot f(OPT)$, for every set $S \subseteq \mathcal{N}$.

3 Measured Continuous Greedy

In this section we describe the unified measured continuous greedy algorithm that works for both non-monotone and monotone cases. We analyze it for general non-monotone submodular functions and then refine the analysis to get improved results for monotone submodular functions. The parameter T of the algorithm is the *stopping time* mentioned in Theorems 1.2 and 1.1.

It is important to note the differences of Algorithm 1 with respect to the known continuous greedy algorithm of [7]. As mentioned before, we distort the direction y which the algorithm goes to at each step. This can bee seen in line 8 of Algorithm 1 as we multiply $I_e(t)$ with $1 - y_e(t)$.

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

Algorithm 1: Measured Continuous Greedy$(f, \mathcal{P}, T)$

```
// Initialization
Set: n ← |𝒩|, δ ← T(⌈n^5 t⌉)^{-1}.
Initialize: t ← 0, y(0) ← 1_∅.
// Main loop
while t < T do
    foreach e ∈ 𝒩 do
        w_e(t) ← F(y(t) ∨ 1_e) − F(y(t)).
    Let I(t) ∈ 𝒫 be a vector maximizing I(t) · w(t).
    foreach e ∈ 𝒩 do
        y_e(t + δ) ← y_e(t) + δ I_e(t) · (1 − y_e(t)).
    t ← t + δ.
Return y(T).
```

Remarks

- The way δ is defined implies that δ^{-1} has two properties: it is at least n^5, and it is dividable by T^{-1}. The last property guarantees that after $T\delta^{-1}$ iterations, t will be exactly T.
- In some applications, the calculation of $w_e(t)$ can be done efficiently. In cases where it is not true, $w_e(t)$ can be estimated by averaging its value for enough independent random samples of $R(y(t))$. This is a standard practice (see, *e.g.*, [8]), and we omit details from this extended abstract.

Due to space limitations, many proofs of this section are deferred to Appendix C.

3.1 Analysis for Non-Monotone f

In this subsection we analyze the measured continuous greedy algorithm for general non-negative submodular functions, and prove Theorem 1.1.

Lemma 3.1. *For every $T \geq 0$, the measured continuous greedy algorithm produces a solution x such that $x/T \in \mathcal{P}$.*

The following two lemmata give together a lower bound on the improvement achieved by the algorithm in each iteration. This lower bound is stated explicitly in Corollary 3.4.

Lemma 3.2. *For every time $0 \leq t < T$, $\sum_{e \in \mathcal{E}} (1 - y_e(t)) \cdot I_e(t) \cdot \partial_e F(y(t)) \geq F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t))$.*

Lemma 3.3. *Consider two vectors $x, x' \in [0,1]^{\mathcal{N}}$ such that for every $e \in \mathcal{N}$, $|x_e - x'_e| \leq \delta$. Then, $F(x') - F(x) \geq \sum_{e \in \mathcal{N}} (x'_e - x_e) \cdot \partial_e F(x) - O(n^3 \delta^2) \cdot f(OPT)$.*

The proof of Lemma 3.3 uses standard methods, and, for completeness, appears in Appendix E.

Corollary 3.4. *For every time $0 \leq t < T$, $F(y(T+\delta)) - F(y(T)) \geq \delta \cdot [F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t))] - O(n^3 \delta^2) \cdot f(OPT)$.*

Proof. Follows immediately from Lemmata 3.2 and 3.3. □

The last corollary gives a lower bound in terms of $F(y(t) \vee \mathbf{1}_{OPT})$. To make this lower bound useful, we need to lower bound the term $F(y(t) \vee \mathbf{1}_{OPT})$. This is done by the following two lemmata.

Lemma 3.5. *Consider a vector $x \in [0,1]^{\mathcal{N}}$. Assuming $x_e \leq a$ for every $e \in \mathcal{N}$, then for every set $S \subseteq \mathcal{N}$, $F(x \vee \mathbf{1}_S) \geq (1-a) f(S)$.*

Lemma 3.6. *For every time $0 \leq t \leq T$ and element $e \in \mathcal{N}$, $y_e(t) \leq 1 - (1-\delta)^{t/\delta} \leq 1 - e^{-t} + O(\delta)$.*

Proof. We prove the first inequality by induction on t. For $t = 0$, the inequality holds because $y_e(0) = 0 = 1 - (1-\delta)^{0/\delta}$. Assume the inequality holds for some t, let us prove it for $t + \delta$.

$$\begin{aligned} y_e(t+\delta) &= y_e(t) + \delta I_e(t)(1 - y_e(t)) = y_e(t)(1 - \delta I_e(t)) + \delta I_e(t) \\ &\leq (1 - (1-\delta)^{t/\delta})(1 - \delta I_e(t)) + \delta I_e(t) = 1 - (1-\delta)^{t/\delta} + \delta I_e(t)(1-\delta)^{t/\delta} \\ &\leq 1 - (1-\delta)^{t/\delta} + \delta(1-\delta)^{t/\delta} = 1 - (1-\delta)^{(t+\delta)/\delta} \ . \end{aligned}$$

We complete the proof by deriving the second inequality: $1 - (1-\delta)^{t/\delta} \leq 1 - [e^{-1}(1-\delta)]^t = 1 - e^{-t}(1-\delta)^t \leq 1 - e^{-t}(1 - T\delta) = 1 - e^{-t} + O(\delta)$, where the last inequality holds since $t \in [0, T]$. □

Corollary 3.7. *For every time $0 \leq t < T$, $F(y(T+\delta)) - F(y(T)) \geq \delta \cdot [(e^{-t} - O(\delta)) \cdot f(OPT) - F(y(t))] - O(n^3\delta^2) \cdot f(OPT) = \delta \cdot [e^{-t} \cdot f(OPT) - F(y(t))] - O(n^3\delta^2) \cdot f(OPT)$.*

Proof. By Lemma 3.6, every coordinate in $y(t)$ is at most $1 - e^{-t} + O(\delta)$. Therefore, by Lemma 3.5, $F(y(t) \vee \mathbf{1}_{OPT})] \geq [e^{-t} - O(\delta)] \cdot f(OPT)$. Plugging this into Corollary 3.4 completes the proof. □

At this point we have a lower bound on the improvement achieved in each iteration in terms of t, $f(OPT)$ and $F(y(t))$. In order to complete the analysis of the algorithm, we need to derive from it a bound on the value of $F(y(t))$ for every time t. Let $g(t)$ be defined as following. $g(0) = 0$ and $g(t+\delta) = g(t) + \delta[e^{-t} f(OPT) - g(t)]$. The next lemma shows that a lower bound on $g(t)$ also gives a lower bound on $F(y(t))$

Lemma 3.8. *For every $0 \leq t \leq T$, $g(t) \leq F(y(t)) + O(n^3\delta) \cdot t f(OPT)$.*

The function g is given by a recursive formula, thus, evaluating it is not immediate. Instead, we show that the function $h(t) = te^{-t} \cdot f(OPT)$ lower bounds g within the range $[0, 1]$.

Lemma 3.9. *Given that $T \in [0, 1]$, for every $0 \leq t \leq T$, $g(t) \geq h(t)$.*

We can now use the last result to lower bound the quality of the algorithm's output.

Corollary 3.10. *For $T \in [0, 1]$, $F(y(T)) \geq [Te^{-T} - o(1)] \cdot f(OPT)$.*

Proof. By Lemmata 3.8 and 3.9, $F(y(T)) \geq g(T) - O(n^3\delta) \cdot T f(OPT) \geq h(T) - O(n^3\delta) \cdot f(OPT) = [Te^{-T} - O(n^3\delta)] \cdot f(OPT)$. Recall that $\delta \leq n^{-5}$, hence, $O(n^3\delta) = o(1)$, and the proof is complete. □

Theorem 1.1 now follows immediately from Lemma 3.1 and Corollary 3.10.

3.2 Analysis for Monotone f

In this subsection we analyze the measured continuous greedy algorithm for normalized monotone submodular functions, and prove Theorem 1.2. The proof of Theorem 1.3 builds on the proofs in this section, and is deferred to Appendix B due to space limitations. Observe that we can use all claims of Subsection 3.1 because a normalized monotone submodular function is a specific case of a non-negative submodular function.

Our proof has two parts. In the first part we modify the proof from Subsection 3.1 to show that $F(y(T)) \geq [(1 - e^T) - o(1)] \cdot f(OPT)$. This part of the proof essentially reproduce the result Calinescu et al. [8] achieve using their continuous greedy algorithm. The novel part of the proof is the second part showing that if $\mathcal{P}$ is a packing polytope and $T \leq T_{\mathcal{P}}$, then $y(T) \in \mathcal{P}$. Let us begin with the first part of the proof. The following observation logically replace Lemma 3.5.

Observation 3.11. *Consider a vector $x \in [0, 1]^{\mathcal{N}}$, then for every set S, $\mathbb{E}[f(R(x) \cup S)] \geq f(S)$.*

Proof. Follows immediately from the monotonicity of f. □

Using Observation 3.11 instead of Lemma 3.5 in the proof of Corollary 3.7, we get the following improved corollary.

Corollary 3.12. *For every time $0 \leq t < T$, $F(y(T+\delta)) - F(y(T)) = \delta \cdot [f(OPT) - F(y(t))] - O(n^3\delta^2) \cdot f(OPT)$.*

We now define $\tilde{g}(t)$, the counterpart of g from Subsection 3.1, as following. $\tilde{g}(0) = 0$ and $\tilde{g}(t+\delta) = g(t) + \delta[f(OPT) - g(t)]$. Using $\tilde{g}$, we get the following counterpart of Lemma 3.8. The proof of the lemma is omitted since it is completely analog to the proof of Lemma 3.8.

Lemma 3.13. *For every $0 \leq t \leq T$, $\tilde{g}(t) \leq F(y(t)) + O(n^3\delta)tf(OPT)$.*

Let $\tilde{h}(t)$ be the function $\tilde{h}(t) = (1 - e^{-t}) \cdot f(OPT)$. $\tilde{h}$ is the counterpart of h, and using it we can write the following counterpart of Lemma 3.9.

Lemma 3.14. *For every $0 \leq t \leq T$, $\tilde{g}(t) \geq \tilde{h}(t)$.*

Finally, we can prove a counterpart of Corollary 3.10.

Corollary 3.15. *$F(y(T)) \geq [1 - e^{-T} - o(1)] \cdot f(OPT)$.*

Proof. By Lemmata 3.13 and 3.14, $F(y(T)) \geq g(T) - O(n^3\delta) \cdot Tf(OPT) \geq h(T) - O(n^3\delta) \cdot Tf(OPT) = [1 - e^{-T} - O(n^3\delta) \cdot T] \cdot f(OPT)$. Recall that $\delta \leq n^{-5}$, hence, $O(n^3\delta) \cdot T = O(n^{-2}) \cdot T = o(1)$, completing the proof of the corollary. □

The first part of the proof is now complete. We are left to prove that if $\mathcal{P}$ is a packing constraint and $T \leq T_{\mathcal{P}}$, then $y(T) \in \mathcal{P}$. Consider some general constraint $\sum_{e \in \mathcal{N}} a_e x_e \leq b$ of $\mathcal{P}$. We assume $a_e > 0$ for some $e \in \mathcal{N}$, otherwise, the constraint holds always and can be ignored. Let $I_e^t = \delta \cdot \sum_{i=0}^{t/\delta - 1} I_e(\delta \cdot i)$, *i.e.*, I_e^t is the scaled sum of I_e over all times up to time t. The following two lemmata prove some properties of I_e^t.

Lemma 3.16. *$\sum_{e \in \mathcal{N}} a_e \cdot I_e^T \leq Tb$.*

Lemma 3.17. *For every $0 \leq t \leq T$, $y_e(t) \leq 1 - e^{-I_e^t} + O(\delta) \cdot t$.*

The following lemma is a mathematical observation needed to combine the last two lemmata.

Lemma 3.18. *Let $c_1, c_2 > 0$, and let z_1, z_2 be two variables whose values obey $c_1 z_1 + c_2 z_2 = s$ for some constant s. Then, the expression $c_1(1 - e^{-z_1}) + c_2(1 - e^{-z_2})$ is maximized when $z_1 = z_2$.*

The following lemma upper bounds the left hand side of our general constraint $\sum_{e \in \mathcal{N}} a_e \cdot x_e \leq b$ at time T. The proof of the lemma uses the last three lemmata.

Lemma 3.19. *Let $\mathcal{N}' \subseteq \mathcal{N}$ be the set of elements with a strictly positive a_e. Then, $\sum_{e \in \mathcal{N}'} a_e \cdot y_e(T) \leq \frac{b}{d(\mathcal{P})} \cdot (1 - e^{-Td(\mathcal{P})} + O(\delta) \cdot T)$.*

Proof. By Lemma 3.17:

$$\sum_{e \in \mathcal{N}'} a_e \cdot y_e(T) \leq \sum_{e \in \mathcal{N}'} a_e \cdot (1 - e^{-I_e^T} + O(\delta) \cdot T) \leq \sum_{e \in \mathcal{N}'} a_e \cdot (1 - e^{-I_e^T}) + O(\delta) \cdot Tb/d(\mathcal{P}) \ .$$

The second term of the right hand side is independent of the values taken by the I_e^T's, therefore, we can upper bound the entire right hand side by assigning to the I_e^t's values maximizing the first term. Let us determine these values.

- Since the summand is an increasing function of I_e^T, the sum $\sum_{e \in \mathcal{N}'} a_e \cdot I_e^T$ should have its maximal value, which is Tb by Lemma 3.16.
- By Lemma 3.18, the maximum is attained when I_e^T is identical for all elements $e \in \mathcal{N}'$.

It can be easily seen that the sole solution satisfying these conditions is $I_e^T = Tb / \sum_{e \in \mathcal{N}'} a_e$. Plugging this solution to into the previous bound on $\sum_{e \in \mathcal{N}'} a_e \cdot y_e(T)$, we get:

$$\begin{aligned} \sum_{e \in \mathcal{N}'} a_e \cdot y_e(T) &\leq \sum_{e \in \mathcal{N}'} a_e \cdot (1 - e^{-Tb/\sum_{e \in \mathcal{N}'} a_e}) + O(\delta) \cdot Tb/d(\mathcal{P}) \\ &= (1 - e^{-Tb/\sum_{e \in \mathcal{N}'} a_e}) \cdot \sum_{e \in \mathcal{N}'} a_e + O(\delta) \cdot Tb/d(\mathcal{P}) \ . \end{aligned}$$

Let us denote by Σ the sum $\sum_{e\in\mathcal{N}'} a_e$. The first term of the last expression can now be rewritten as $\Sigma(1-e^{-Tb/\Sigma})$, and its derivative by Σ is:

$$\frac{d[S(1-e^{-Tb/\Sigma})]}{d\Sigma} = (1-e^{-Tb/\Sigma}) - S\cdot\frac{Tb}{S^2}e^{-Tb/\Sigma} \geq 1 - e^{Tb/\Sigma}\cdot e^{-Tb/\Sigma} = 0 \ .$$

Hence, increasing the value of Σ only worsens the bound we have on $\sum_{e\in\mathcal{N}'} a_e\cdot y_e(T)$. Therefore, we can plug $\Sigma = b/d(\mathcal{P})$, which is an upper bound on Σ, and get:

$$\sum_{e\in\mathcal{N}'} a_e\cdot y_e(T) \leq (1-e^{-Tb/(b/d(\mathcal{P}))})\cdot b/d(\mathcal{P}) + O(\delta)\cdot Tb/d(\mathcal{P}) = \frac{b}{d(\mathcal{P})}\cdot(1-e^{-Td(\mathcal{P})}+O(\delta)\cdot T) \ . \ \square$$

As long as the upper bound proved in the last lemma is at most b, the constraint $\sum_{e\in\mathcal{N}} a_e x_e \leq b$ is not violated. The next corollary shows that if $T\leq T_{\mathcal{P}}$, then this is the case.

Corollary 3.20. *Given that $T\leq T_{\mathcal{P}}$, the solution $y(T)$ respects the constraint $\sum_{e\in\mathcal{N}} a_e x_e \leq b$. Moreover, since $\sum_{e\in\mathcal{N}} a_e x_e \leq b$ is an arbitrary constraint of $\mathcal{P}$, $y(T)\in\mathcal{P}$.*

Proof. By Lemma 3.19,

$$\begin{aligned}\sum_{e\in\mathcal{N}} a_e\cdot y_e(T) &= \sum_{e\in\mathcal{N}'} a_e\cdot y_e(T) \leq \sum_{e\in\mathcal{N}'} a_e\cdot y_e(T_{\mathcal{P}}) \leq \frac{b}{d(\mathcal{P})}\cdot(1-e^{-T_{\mathcal{P}}d(\mathcal{P})}+O(\delta)\cdot T_{\mathcal{P}})\\ &= \frac{b}{d(\mathcal{P})}\cdot(1-e^{\ln(1-d(\mathcal{P})+n\delta)}+O(\delta)\cdot T_{\mathcal{P}}) = \frac{b}{d(\mathcal{P})}\cdot(d(\mathcal{P})-n\delta+O(\delta)\cdot T_{\mathcal{P}}) \leq b \ .\square\end{aligned}$$

Theorem 1.2 now follows from Lemma 3.1 and Corollaries 3.15 and 3.20.

4 Applications

This section describes the immediate applications of the measured continuous greedy algorithm. For other applications involving our algorithm and the framework of [12], see Appendixes F and G. Due to space limitations, many proofs of this section are deferred to Appendix D.

4.1 Non-Monotone Submodular Function Subject to Matroid Constraint

In this section we consider the problem of maximizing a non-monotone submodular function subject to a matroid constraint. Formally, given a matroid $M=(\mathcal{N},\mathcal{I})$ and a non-negative submodular function $f:2^{\mathcal{N}}\to\mathbb{R}^+$, the objective is to find an independent set $S\in\mathcal{I}$ maximizing $f(S)$.

Theorem 4.1. *There is a polynomial time $(1/e-o(1))$-approximation algorithm for maximizing a general non-negative submodular function subject to a matroid constraint.*

Proof. Let $\mathcal{P}(M)$ be the matroid polytope corresponding to M. Since $\mathcal{P}(M)$ is a solvable down monotone polytope, by Theorem 1.1, applying the measured continuous greedy to it, with stopping time $T=1$, produces a point $x\in\mathcal{P}(M)$ such that $F(x)\geq[1/e-o(1)]\cdot f(OPT)$.

The point x can then be rounded using pipage rounding. By Theorem 2.3, this rounding returns a random set S which is an independent set of M and $\mathbb{E}[f(S)]\geq F(x)\geq[e^{-1}-o(1)]\cdot f(OPT)$. $\square$

4.2 Non-Monotone Submodular Function Subject to Knapsack Constraints

In this section we consider the problem of maximizing a non-monotone submodular function subject to a constant number of knaspack constraints. Formally, we are given a ground set $\mathcal{N}$, a set of d knapsack constraints over this ground set (where d is considered to be a constant) and a non-negative submodular function $f:2^{\mathcal{N}}\to\mathbb{R}^+$. The objective is to find a set $S\subseteq\mathcal{N}$ satisfying all knapsack constraints and maximizing $f(S)$.

Let $\mathcal{P}$ be the polytope defined by the d knapsack constraints and the cube $[0,1]^{\mathcal{N}}$. Observe that $\mathcal{P}$ is a down monotone solvable polytope. The following theorem shows that it is possible to round fractional points in $\mathcal{P}$.

Theorem 4.2 (Theorem 2.6 in [36]). *Suppose there is a polynomial time α-approximation algorithm for finding a point $x \in \mathcal{P}$ maximizing $F(x)$. Then, for every constant $\varepsilon > 0$, there is a polynomial time randomized $(\alpha-\varepsilon)$-approximation algorithm for maximizing a non-monotone submodular function subject to k knapsack constraints.*

Theorem 4.2 assumes the existence of a fractional approximation algorithm. We observe that the measured continuous greedy algorithm can be used as this algorithm.

Corollary 4.3. *For any constant $\varepsilon > 0$, there is a polynomial time $(1/e - \varepsilon)$-approximation algorithm for maximizing a general non-negative submodular function subject to knapsack constraints.*

Proof. By Theorem 1.1, applying the measured continuous greed to $\mathcal{P}$, with $T = 1$, produces a point $x \in \mathcal{P}$ such that $F(x) \geq [1/e - o(1)] \cdot f(OPT)$. The corollary now follows by Theorem 4.2. □

4.3 The Submodular Welfare and Submodular Max-SAT Problems

In this section consider the Submodular Welfare and Submodular Max-SAT problems. Both problems are generalized by the (d, r)-Submodular Partition Problem. A (d, r)-partition matroid (considered by [2]) is a matroid defined over a groundset $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2 \cup \ldots \cup \mathcal{N}_m$, where $|\mathcal{N}_i| = r$ for every $1 \leq i \leq m$. A set $S \subseteq \mathcal{I}$ is independent if it contains up to d elements of each subset $\mathcal{N}_i$. In the (d, r)-Submodular Partition problem, given a (d, r)-partition matroid M and a normalized monotone submodular function f, the objective is to find an independent set S maximizing $f(S)$.

Observation 4.4. *Let M be a (d, r)-partition matroid, then $\mathcal{P}(M)$ is a binary packing matroid with density $d(\mathcal{P}(\mathcal{M})) = d/r$.*

Proof. $\mathcal{P}(M)$ is a polytope defining by m constraints, one for each subset $\mathcal{N}_i$. The constraint of $\mathcal{N}_i$ is $\sum_{e \in \mathcal{N}_i} x_e \leq d$. Observe that all coefficients in this constraint belong to $\{0, 1\}$. The number of terms in the sum is r, and therefore, the density imposed by this constraint is: $d / \sum_{e \in \mathcal{E}_i} 1 = d/r$. □

Lemma 4.5. *There is a polynomial time $(1 - (1 - d/r)^{r/d})$-approximation algorithm for (d, r)-Submodular Partition.*

Proof. Observation 4.4 together with Theorem 1.3 give a polynomial time approximation algorithm that finds a point $x \in \mathcal{P}(M)$ with $F(x) \geq [1 - (1 - d/r)^{r/d}] \cdot f(OPT)$.

The point x can then be rounded using pipage rounding. By Theorem 2.3, this rounding returns a independent random set S of M, and $\mathbb{E}[f(S)] \geq F(x) \geq [1 - (1 - d/r)^{r/d}] \cdot f(OPT)$. □

In Submodular Welfare there are k players and n elements $\mathcal{E}$. Each player is associated with a normalized monotone submodular utility function f_i. The objective is to partition the elements among the players, and maximize the total utility of the players for the items that were assigned to them. Formally, we need to partition $\mathcal{E}$ into: $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k$ (where $\mathcal{E}_i$ is the set of elements assigned to player i), and maximize $\sum_{i=1}^{k} f_i(\mathcal{E}_i)$.

Lemma 4.6. *An α-approximation for $(1, k)$-Submodular Partition implies an α-approximation for Submodular Welfare with k players.*

Corollary 4.7. *There is a polynomial time $1-(1-1/k)^k$-approximation algorithm for Submodular Welfare.*

Proof. Follows immediately from Lemmata 4.5 and 4.8. □

In Submodular Max-SAT we are given a CNF formula and a normalized monotone submodular function over the set of clauses in the formula. The goal is to find an assignment ϕ to the variables maximizing the value of f over the set of clauses satisfied by ϕ. Let $\mathcal{X}$, $\mathcal{C}$ and m denote the set of variables, set of clauses and number of clauses in the formula, respectively. Using this notation, the objective function is a function $f : 2^{\mathcal{C}} \to \mathbb{R}^+$, and an assignment is a function $\phi : \mathcal{X} \to \{0, 1\}$.

Lemma 4.8. *There is an α-approximation for Submodular Max-SAT if and only if $(1, 2)$-Submodular Partition has.*

Corollary 4.9. *There is a polynomial time $3/4$-approximation algorithm for Submodular Max-SAT.*

Proof. Follows immediately from Lemmata 4.5 and 4.8. □

References

[1] Alexander A. Ageev and Maxim Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *J. Comb. Optim.*, 8(3):307–328, 2004.

[2] Yossi Azar, Iftah Gamzu, and Ran Roth. Beating the $(1-1/e)$-threshold: Submodular function maximization under certain matroid constraints, 2011. Manuscript.

[3] Nikhil Bansal, Nitish Korula, Viswanath Nagarajan, and Aravind Srinivasan. On k-column sparse packing programs. In *Integer Programming and Combinatorial Optimization*, volume 6080 of *Lecture Notes in Computer Science*, pages 369–382. 2010.

[4] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, 2001.

[5] Amotz Bar-Noy, Sudipto Guha, Yoav Katz, Joseph (Seffi) Naor, Baruch Schieber, and Hadas Shachnai. Throughput maximization of real-time scheduling with batching. *ACM Trans. Algorithms*, 5(2):1–17, 2009.

[6] Amotz Bar-Noy, Sudipto Guha, Joseph (Seffi) Naor, and Baruch Schieber. Approximating the throughput of multiple machines under real-time scheduling. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 622–631, New York, NY, USA, 1999. ACM.

[7] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *Proceedings of the 12th Conference on Integer Programming and Combinatorial Optimization*, IPCO '07, pages 182–196, 2007.

[8] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. *to appear in SIAM Journal on Computing, Special Issue for STOC 2008*, 2008.

[9] Deeparnab Chakrabarty and Gagan Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and gap. In *In Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science-Volume 00*, pages 687–696, 2008.

[10] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Multi-budgeted matchings and matroid intersection via dependent rounding. In *to appear in SODA 2011*.

[11] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 575–584, Washington, DC, USA, 2010. IEEE Computer Society.

[12] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *to appear in STOC '11*, 2011.

[13] M. Conforti and G. Cornuèjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Disc. Appl. Math.*, 7(3):251–274, 1984.

[14] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, STOC '05, pages 610–618, 2005.

[15] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 1064–1073, 2006.

[16] Uriel Feige. On maximizing welfare when utility functions are subadditive. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 41–50, 2006.

[17] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 461–471, Washington, DC, USA, 2007. IEEE Computer Society.

[18] Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 667–676, Washington, DC, USA, 2006. IEEE Computer Society.

[19] Uriel Feige and Jan Vondrák. The submodular welfare problem with demand queries. *Theory of Computing*, 6(1):247–290, 2010.

[20] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Nonmonotone submodular maximization via a structural continuous greedy algorithm. *Submitted*, 2011.

[21] M. Fisher, G. Nemhauser, and L. Wolsey. An analysis of approximations for maximizing submodular set functions - ii. *Math. Prog. Study*, 8:73–87, 1978.

[22] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *to appear in SODA 2011*, 2011.

[23] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115–1145, November 1995.

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

[25] D. Hausmann and B. Korte. K-greedy algorithms for independence systems. *Oper. Res. Ser. A-B*, 22(1):219–228, 1978.

[26] D. Hausmann, B. Korte, and T. Jenkyns. Worst case analysis of greedy type algorithms for independence systems. *Math. Prog. Study*, 12:120–131, 1980.

[27] Dorit S. Hochbaum. *Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems*, pages 94–143. PWS Publishing Co., Boston, MA, USA, 1997.

[28] T. Jenkyns. The efficacy of the greedy algorithm. *Cong. Num.*, 17:341–350, 1976.

[29] 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.

[30] 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.

[31] Subhash Khot, Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. In *in Proceedings of WINE 2005*, pages 92–101, 2005.

[32] Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70:39–45, April 1999.

[33] B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. *Annals of Discrete Math.*, 2:65–74, 1978.

[34] Ariel Kulik and Hadas Shachnai. Improved approximations for non-monotone submodular maximization with knapsack constraints. Manuscript, 2010.

[35] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 545–554, Philadelphia, PA, USA, 2009.

[36] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Approximations for monotone and non-monotone submodular maximization with knapsack constraints, 2011. Manuscript.

[37] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 323–332, New York, NY, USA, 2009. ACM.

[38] Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010.

[39] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, EC '01, pages 18–28, 2001.

[40] László Lovász. Submodular functions and convexity. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming: the State of the Art*, pages 235–257. Springer, 1983.

[41] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[42] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[43] G. Nemhauser and L. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978.

[44] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions - i. *Math. Prog.*, 14(1):265–294, 1978.

[45] Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129:192–224, 2006.

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

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

[48] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 67–74, New York, NY, USA, 2008. ACM.

[49] Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 651–670, Washington, DC, USA, 2009. IEEE Computer Society.

Appendix

A Down Monotone Polytopes in the Hypercube $[0,1]^{\mathcal{N}}$

Let $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$ be a down monotone polytope defined by positive sign constraints and m additional inequality constraints. Let $\sum_{e\in\mathcal{N}} a_{i,e}x_e \leq b_i$ be the i^{th} constraint defining $\mathcal{P}$. In this section we justify some assumptions on the coefficients of the inequality constraints.

Observation A.1. *We can assume $\mathcal{P}$ has a positive sign constraint $x_e \geq 0$, and an inequality constraint of the form $x_e \leq 1$ for every $e \in \mathcal{N}$.*

Proof. Since $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$, we can add such these constraints for every variable missing them, without modifying $\mathcal{P}$. □

Lemma A.2. *For every $1 \leq i \leq m$, b_i is non-negative.*

Proof. Since $\mathcal{P}$ is down monotone, $\mathbf{1}_{\varnothing} \in \mathcal{P}$. The point $\mathbf{1}_{\varnothing}$ induce the value of 0 on the left hand side of all constraints, and therefore, the free coefficients must be non-negative. □

Lemma A.3. *We can assume that for every $1 \leq i \leq m$ and $e \in \mathcal{N}$, $a_{i,e}$ is non-negative.*

Proof. Assume this is not the case, then let $a_{j,e'}$ be a negative coefficient. Let $\mathcal{P}'$ be the polytope defined by the same of constraints with the sole modification that $a_{j,e'}$ is changed to 0. By Observation A.1, the change we made only tightens the constraint, and therefore, $\mathcal{P}' \subseteq \mathcal{P}$. Let us show that the last containment is in fact an equality. Let x be a point in $\mathcal{P}$, and let $\sum_{e\in\mathcal{N}} a'_{i,e}x_e \leq b'_i$ be the i^{th} constraint of $\mathcal{P}'$.

For every constraint $i \neq j$, $\mathcal{P}$ and $\mathcal{P}'$ share the same coefficients, and therefore,

$$\sum_{e\in\mathcal{N}} a'_{i,e}x_e = \sum_{e\in\mathcal{N}} a_{i,e}x_e \leq b_i = b'_i \ .$$

Observe now that $x' = x \wedge \mathbf{1}_{\mathcal{N}-\{e'\}} \in \mathcal{P}$ due to the down monotonicity of $\mathcal{P}$. Also, $\mathcal{P}$ and $\mathcal{P}'$ share all coefficients of the j^{th} constraint except for $a_{j,e'}$, and therefore,

$$\sum_{e\in\mathcal{N}} a'_{j,e}x_e = \sum_{e\in\mathcal{N}-\{e'\}} a'_{j,e}x_e = \sum_{e\in\mathcal{N}-\{e'\}} a_{j,e}x'_e = \sum_{e\in\mathcal{N}} a_{j,e}x'_e \leq b_j = b'_j \ .$$

Where the inequality holds since $x' \in \mathcal{P}$. We proved that $x \in \mathcal{P}$ implies $x \in \mathcal{P}'$, and therefore, $\mathcal{P} = \mathcal{P}'$. Hence, replacing a negative coefficient $a_{j,e'}$ by 0 does not change the polytope. The lemma now follows by repeating the argument for every negative coefficient. □

Lemma A.4. *We can assume that for every constraint i, $b_i > 0$.*

Proof. By Lemma A.2, $b_i \geq 0$. Let $\mathcal{N}' = \{e \in \mathcal{N} | a_{e,i} > 0\}$. If $b_i = 0$, the constraint implies that every $e \in \mathcal{N}'$ must be zero everywhere in $\mathcal{P}$, and therefore, the elements of $\mathcal{N}'$ can be removed from the groundset. After the removal of $\mathcal{N}'$'s elements, we are left with a constraint of the form:

$$\sum_{e \in \mathcal{N}} 0 \cdot x_e \leq 0 \ ,$$

and such constraints are meaningless, and can be removed without effecting $\mathcal{P}$. □

Lemma A.5. *We can assume that for every constraint i, $\sum_{e \in \mathcal{N}} a_{e,i} > 0$.*

Proof. By Lemma A.3, $\sum_{e \in \mathcal{N}} a_{e,i} \geq 0$. If $\sum_{e \in \mathcal{N}} a_{e,i} = 0$, the constraints is satisfied for every assignment, and therefore, can be removed. □

Lemma A.6. *We can assume that for every constraint i, $\sum_{e \in \mathcal{N}} a_{e,i} \leq b_i$.*

Proof. For every point $x \in [0,1]^{\mathcal{N}}$ it holds that $\sum_{e \in \mathcal{N}} a_{e,i} x_e \leq \sum_{e \in \mathcal{N}} a_{e,i}$. Hence, if $\sum_{e \in \mathcal{N}} a_{e,i} \leq b_i$, then the i^{th} constraint is redundant for points in $[0,1]^{\mathcal{N}}$. By Observation A.1, the removal of this constraint will not introduce to $\mathcal{P}$ points outside of $[0,1]^{\mathcal{N}}$, and therefore, will not modify $\mathcal{P}$. □

Corollary A.7. *We can assume $0 < d(\mathcal{P}) \leq 1$.*

Proof. For every inequality constraint i, Lemma A.4 guarantees that $b_i > 0$, and Lemma A.5 guarantees that $\sum_{e \in \mathcal{N}} a_{e,i} > 0$. Hence, $b_i / \sum_{e \in \mathcal{N}} a_{e,i}$ is positive. On the other hand, Lemma A.6 implies that $\sum_{e \in \mathcal{N}} a_{e,i} \leq 1$. Since this is true for every inequality constraint of $\mathcal{P}$, its density must be in the range $(0,1]$. □

B Proof of Theorem 1.3

Notice that we can use all claims of Sections 3.1 and 3.2 because these sections deal with more general settings than this one. Corollary 3.20 states that if $T \leq T_{\mathcal{P}}$, then it outputs a feasible solution. The next lemma lower bounds the value of this solution by plugging $T = T_{\mathcal{P}}$ into the lower bound of Corollary 3.15..

Lemma B.1. $F(y(T_{\mathcal{P}})) \geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - O(n^{-2})] \cdot f(OPT) = [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - o(1)] \cdot f(OPT)$.

Proof. Let us derive $1 - (1 - d(\mathcal{P}) + c)^{1/d(\mathcal{P})}$ by c (assuming c is small).

$$\frac{d[1 - (1 - d(\mathcal{P}) + c)^{1/d(\mathcal{P})}]}{dc} = -\frac{(1 - d(\mathcal{P}) + c)^{1/d(\mathcal{P})-1}}{d(\mathcal{P})} \geq -\frac{1}{d(\mathcal{P})(1 - d(\mathcal{P}))} \ .$$

Hence, for small enough c, $1 - (1 - d(\mathcal{P}) + c)^{1/d(\mathcal{P})} \geq 1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - c/(d(\mathcal{P})(1 - d(\mathcal{P})))$. By the proof of Corollary 3.15:

$$\begin{aligned}
F(y(T_{\mathcal{P}})) &\geq [1 - e^{-T_{\mathcal{P}}} - O(n^{-2}) \cdot T_{\mathcal{P}}] \cdot f(OPT) = [1 - (1 - d(\mathcal{P}) + n\delta)^{1/d(\mathcal{P})} - O(n^{-2}) \cdot T_{\mathcal{P}}] \cdot f(OPT) \\
&\geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - \frac{n\delta}{d(\mathcal{P})(1 - d(\mathcal{P}))} - O(n^{-2}) \cdot T_{\mathcal{P}}] \cdot f(OPT) \\
&\geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - n^2\delta - O(n^{-2}) \cdot T_{\mathcal{P}}] \cdot f(OPT) \\
&= [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - O(n^{-2})] \cdot f(OPT) \ .
\end{aligned}$$

Where the third inequality holds for $n \geq (d(\mathcal{P})(1 - d(\mathcal{P}))^{-1}$, and the last equality follows since $\delta \leq n^{-5}$. □

Algorithm 2: Measured Continuous Greedy with Enumeration$(f, \mathcal{P})$

// Guess

Guess an element $e^* \in OPT$ such that $f(e^*) \geq f(OPT)/n$.

// Definitions

Let $\mathcal{P}'$ be the polytope formed from $\mathcal{P}$ by setting x_{e^*} to be identically 1.

Let $\mathcal{E}_0$ be the set of variables in $\mathcal{E}$ which are identically 0 in $\mathcal{P}'$.

Set $\mathcal{E}' \leftarrow \mathcal{E} - \mathcal{E}_0 - \{e^*\}$.

Define $f'(A) = f(A \cup \{e^*\}) - f(\{e^*\})$.

// Solve

Use the measured continuous greedy algorithm with ground set $\mathcal{E}'$, submodular function f', polytope $\mathcal{P}'$ and stopping time $T_{\mathcal{P}}$.

Let $y(T_{\mathcal{P}})$ be the output of the measured continuous greedy algorithm.

Output $y(T_{\mathcal{P}}) \vee \mathbf{1}_{e^*}$.

We now need to show how to get rid of the $o(1)$ term in the guarantee of the last lemma. Consider the following algorithm.

First, let us make sure that there is an element e^* that the algorithm can guess.

Lemma B.2. *There is an element $e^* \in OPT$ satisfying the requirement of the algorithm, i.e., $f(e^*) \geq f(OPT)/n$.*

Proof. Assume for the sake of contradiction that every element $e \in OPT$ has $f(e) < f(OPT)/n$. Then, we get:

$$f(OPT) \leq \sum_{e \in OPT} f(e) < \sum e \in OPT \frac{f(OPT)}{n} = \frac{|OPT| \cdot f(OPT)}{n} \leq f(OPT) \ .$$

And this is of course a contradiction. □

Algorithm 2 applies the measured continuous greedy algorithm with the objective function f'. The following lemma shows that f' has the properties required by Theorem 1.2.

Lemma B.3. *f' is a normalized monotone submodular function over the ground set $\mathcal{E}'$.*

Proof. f is submodular because for every two subsets $A, B \in \mathcal{E}'$:

$$\begin{aligned} f'(A) + f'(B) &= f(A \cup \{e^*\}) - f(\{e^*\}) + f(B \cup \{e^*\}) - f(\{e^*\}) \\ &\geq f(A \cup B \cup \{e^*\}) + f((A \cup \{e^*\}) \cap (B \cup \{e^*\})) - 2f(\{e^*\}) \\ &= f(A \cup B \cup \{e^*\}) - f(\{e^*\}) + f((A \cap B) \cup \{e^*\}) - f(\{e^*\}) \\ &= f'(A \cup B) + f'(A \cap B) \ . \end{aligned}$$

f is monotone because for every $A \subseteq B \subseteq \mathcal{E}'$:

$$f'(A) = f(A \cup \{e^*\}) - f(\{e^*\}) \leq f(B \cup \{e^*\}) - f(\{e^*\}) = f'(B) \ .$$

And finally, f' is normalized because:

$$f'(\varnothing) = f(\varnothing \cup \{e^*\}) - f(\{e^*\}) = 0 \ .$$ □

In order for us to use the full power of Theorem 1.2, we need to demonstrate that $\mathcal{P}$ is a packing polytope and bound its density.

Lemma B.4. *The polytope $\mathcal{P}'$ can be represented as a packing polytope over $\mathcal{E}'$ with $d(\mathcal{P}') \geq d(\mathcal{P})$.*

Proof. Consider a general constraint $\sum_{e\in\mathcal{E}} a_e x_e \leq b$ of $\mathcal{P}$. The corresponding constraint in $\mathcal{P}'$ is $\sum_{e\in\mathcal{E}'} a_e x_e \leq b - a_{e^*}$. If all the coefficients a_e in this constraint are 0, the constraint always holds, and therefore, can be removed. Hence, we can assume this is not the case, and there exists $e' \in \mathcal{E}'$ such that $a_{e'} = 1$.

Clearly $a_{e^*} \leq b$ because otherwise e^* could not have been in a feasible solution, contradicting our assumption that $e^* \in OPT$. Therefore, the free coefficient of the constraint is either 0 or 1. If $b - a_{e^*} = 0$, then this constraint implies $x_{e'} = 0$, contradicting the fact $e' \in \mathcal{E}'$. Hence, $b - a_{e^*} = 1$, which implies $a_{e^*} = 0$. Thus,

$$\frac{b - a_{e^*}}{\sum_{e\in\mathcal{E}'} a_e} \geq \frac{b}{\sum_{e\in\mathcal{E}} a_e} \ .$$

The last inequality holds for a general constraint of $\mathcal{P}$, and therefore, $d(\mathcal{P}') \geq d(\mathcal{P})$. □

Let us denote by x be the output of Algorithm 2.

Corollary B.5. *$x \in \mathcal{P}$.*

Proof. Lemmata B.3 and B.4 imply that $T_{\mathcal{P}'} \geq T_{\mathcal{P}}$, and therefore, $y(T_{\mathcal{P}}) \in \mathcal{P}'$ by Theorem 1.2. Hence, by the definition of $\mathcal{P}'$, $x = y(T_{\mathcal{P}}) \vee 1_{e^*} \in \mathcal{P}$. □

To complete the proof of Theorem 1.3, we are only left to lower bound $F(x)$.

Lemma B.6. *$F(x) \geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})}] \cdot f(OPT)$.*

Proof. Let F' be the multilinear extension of f'. Observe that $OPT - \{e^*\}$ is a feasible integral point in $\mathcal{P}'$, and therefore, by Lemmata B.1 and B.4, $F'(y(T_{\mathcal{P}})) \geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - cn^{-2}] \cdot f'(OPT - \{e^*\})$ for some constant c (that depends on $d(\mathcal{P})$). Hence,

$$\begin{aligned}
F(x) &= \mathbb{E}[f(\{e^*\} \cup R(y(T_{\mathcal{P}})))] = \mathbb{E}[f(\{e^*\}) + f'(R(y(T_{\mathcal{P}})))] = f(\{e^*\}) + F'(y(\ell_z)) \\
&\geq f(\{e^*\}) + [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - cn^{-2}] \cdot f'(OPT - \{e^*\}) \\
&= ((1 - d(\mathcal{P}))^{1/d(\mathcal{P})} + cn^{-2}) \cdot f(\{e^*\}) + (1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - cn^{-2}) \cdot f(OPT) \\
&\geq ((1 - d(\mathcal{P}))^{1/d(\mathcal{P})} + cn^{-2}) \cdot f(OPT)/n + (1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - cn^{-2}) \cdot f(OPT) \\
&\geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})} + n^{-1}((1 - d(\mathcal{P}))^{1/d(\mathcal{P})} - cn^{-1})] \cdot f(OPT) \\
&\geq [1 - (1 - d(\mathcal{P}))^{1/d(\mathcal{P})}] \cdot f(OPT)
\end{aligned}$$

Where the last inequality follows since c and z are bounded while n is arbitrarily large. □

C Omitted Proofs of Section 3

Lemma 3.1. For every $T \geq 0$, the measured continuous greedy algorithm produces a solution x such that $x/T \in \mathcal{P}$.

Proof. Notice that x is coordinate-wise upper bounded by $x' = \delta \cdot \sum_{i=0}^{T/\delta-1} I(i \cdot \delta)$. Since $\mathcal{P}$ is a down-monotone polytope, it is enough to show that $x'/T \in \mathcal{P}$. x'/δ is the sum of T/δ points in $\mathcal{P}$, and therefore, $x'/T = (x'/\delta)/(T/\delta) \in \mathcal{P}$. □

Lemma 3.2. For every time $0 \leq t < T$, $\sum_{e\in\mathcal{E}}(1-y_e(t)) \cdot I_e(t) \cdot \partial_e F(y(t)) \geq F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t))$.

Proof. Recall that $R(x)$ is a random set containing every element $e \in \mathcal{N}$ with probability x_e, and that $F(x) = \mathbb{E}[R(x)]$. Let us calculate the weight of OPT according to weight function $w(t)$.

$$\begin{aligned} w(t) \cdot \mathbf{1}_{OPT} &= \sum_{e \in OPT} w_e(t) = \sum_{e \in OPT} [F(y(t) \vee \mathbf{1}_e) - F(y(t))] \\ &= \mathbb{E}\left[\sum_{e \in OPT} f(R(y(t)) \cup \{e\}) - f(R(y(t)))\right] \\ &\geq \mathbb{E}\left[f(R(y(t)) \cup OPT) - f(R(y(t)))\right] = F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t)) \ . \end{aligned}$$

Where the inequality follows from submodularity. Since $\mathbf{1}_{OPT} \in \mathcal{P}$, we get:

$$w(t) \cdot I(t) \geq F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t)) \ .$$

Hence,

$$\begin{aligned} \sum_{e \in \mathcal{E}} (1 - y_e(t)) \cdot I_e(t) \cdot \partial_e F(y(t)) &= \sum_{e \in \mathcal{E}} I_e(t) \cdot [F(y(t) \vee \mathbf{1}_e) - F(y(t))] = I(t) \cdot w(t) \\ &\geq F(y(t) \vee \mathbf{1}_{OPT}) - F(y(t)) \ . \end{aligned}$$

□

Lemma 3.5. Consider a vector $x \in [0,1]^{\mathcal{N}}$. Assuming $x_e \leq a$ for every $e \in \mathcal{N}$, then for every set $S \subseteq \mathcal{N}$, $F(x \vee \mathbf{1}_S) \geq (1-a)f(S)$.

Proof. Notice that if $\lambda > a$, then $T_\lambda(x) = \varnothing$. By Theorem 2.1, we have:

$$\begin{aligned} F(x \vee \mathbf{1}_S) \geq \hat{f}(x \vee \mathbf{1}_S) &= \int_0^1 f(T_\lambda(x \vee \mathbf{1}_S)) d\lambda = \int_0^1 f(T_\lambda(x) \cup S) d\lambda \\ &\geq \int_a^1 f(T_\lambda(x) \cup S) d\lambda = \int_a^1 f(S) d\lambda = (1-a) \cdot f(S) \ . \end{aligned}$$

□

Lemma 3.8. For every $0 \leq t \leq T$, $g(t) \leq F(y(t)) + O(n^3\delta) \cdot t f(OPT)$.

Proof. Let c be the constant hiding behind the big O notation in Corollary 3.7. We prove by induction on t that $g(t) \leq F(y(t)) + cn^3\delta t f(OPT)$. For $t = 0$, $g(0) = 0 \leq F(y(0))$. Assume now that the claim holds for some t, and let us prove it for $t + \delta$. Using Corollary 3.7, we get:

$$\begin{aligned} g(t+\delta) &= g(t) + \delta[e^{-t} f(OPT) - g(t)] = (1-\delta)g(t) + \delta e^{-t} f(OPT) \\ &\leq (1-\delta)[F(y(t)) + cn^3\delta t f(OPT)] + \delta e^{-t} f(OPT) \\ &= F(y(t)) + \delta[e^{-t} f(OPT) - F(y(t))] + c(1-\delta)n^3\delta t f(OPT) \\ &\leq F(y(t+\delta)) + cn^3\delta^2 f(OPT) + c(1-\delta)n^3\delta t f(OPT) \\ &\leq F(y(t+\delta)) + cn^3\delta(t+\delta) f(OPT) \ . \end{aligned}$$

□

Lemma 3.9. Given that $T \in [0,1]$, for every $0 \leq t \leq T$, $g(t) \geq h(t)$.

Proof. The proof is by induction on t. For $t = 0$, $g(0) = 0 = 0 \cdot e^{-0} \cdot f(OPT) = h(0)$. Assume now that the lemma holds for some t, and let us prove it holds for $t + \delta$.

$$\begin{aligned} h(t+\delta) &= h(t) + \int_t^{t+\delta} h'(\tau) d\tau = h(t) + f(OPT) \cdot \int_t^{t+\delta} e^{-\tau}(1-\tau) d\tau \\ &\leq h(t) + f(OPT) \cdot \delta e^{-t}(1-t) = (1-\delta)h(t) + \delta e^{-t} \cdot f(OPT) \\ &\leq (1-\delta)g(t) + \delta e^{-t} \cdot f(OPT) = g(t) + \delta \cdot [e^{-t} \cdot f(OPT) - g(t)] = g(t+\delta) \ . \end{aligned}$$

□

Lemma 3.14. For every $0 \leq t \leq T$, $\tilde{g}(t) \geq \tilde{h}(t)$.

Proof. The proof is by induction on t. For $t = 0$, $\tilde{g}(0) = 0 = (1 - e^{-0}) \cdot f(OPT) = \tilde{h}(0)$. Assume now that the lemma holds for some t, and let us prove it holds for $t + \delta$.

$$\begin{aligned}\tilde{h}(t+\delta) &= \tilde{h}(t) + \int_t^{t+\delta} \tilde{h}'(\tau)d\tau = \tilde{h}(t) + f(OPT) \cdot \int_t^{t+\delta} e^{-\tau}d\tau \leq \tilde{h}(t) + f(OPT) \cdot \delta e^{-t} \\ &= (1-\delta)\tilde{h}(t) + \delta \cdot f(OPT) \leq (1-\delta)\tilde{g}(t) + \delta \cdot f(OPT) = \tilde{g}(t+\delta) \ . \end{aligned}$$ □

Lemma 3.16. $\sum_{e \in \mathcal{N}} a_e \cdot I_e^T \leq Tb$.

Proof. For every time $t \in [0, T)$, $I(t)$ is a feasible solution, and therefore, $\sum_{e \in \mathcal{N}} a_i \cdot I_e(t) \leq b$. Summing over all times in this range, we get:

$$\sum_{i=0}^{T/\delta - 1} \sum_{e \in \mathcal{N}} a_e \cdot I_e(\delta \cdot i) \leq \sum_{i=0}^{T/\delta - 1} b \ .$$

Since there are T/δ different times in the above range, the right hand side of the last expression is Tb/δ. The lemma now follows by switching the order of summation in the left hand side, and plugging in the definition of I_e^t. □

Lemma 3.17. For every $0 \leq t \leq T$, $y_e(t) \leq 1 - e^{-I_e^t} + O(\delta) \cdot t$.

Proof. We prove by induction on t that $y_e(t) \leq 1 - e^{-I_e^t} + 0.5\delta t$. For $t = 0$ the lemma holds since $y_e(t) = 0 = 1 - e^{-0} + 0 \cdot \delta$. Assume that the lemma holds for some time t, and let us prove it for time $t + \delta$.

$$\begin{aligned} y_e(t+\delta) &= y_e(t) + \delta I_e(t) \cdot (1 - y_e(t)) \leq (1 - e^{-I_e^t} + 0.5t\delta)(1 - \delta I_e(t)) + \delta I_e(t) \\ &\leq 1 - e^{-I_e^t} \cdot (1 - \delta I_e(t)) + 0.5t\delta \leq 1 - e^{-I_e^t} \cdot [e^{-\delta I_e(t)} - 0.5(\delta I_e(t))^2] + 0.5t\delta \\ &\leq 1 - e^{-I_e^t - \delta I_e(t)} + 0.5\delta^2 + 0.5t\delta = 1 - e^{-I_e^{t+\delta}} + 0.5(t+\delta)\delta \ . \end{aligned}$$ □

Lemma 3.18. Let c_1, c_2 be two positive numbers, and let z_1, z_2 be two variables whose values obey $c_1 z_1 + c_2 z_2 = s$ for some constant s. Then, the expression $c_1(1 - e^{-z_1}) + c_2(1 - e^{-z_2})$ is maximized when $z_1 = z_2$.

Proof. The value of z_2 is given, in terms of z_1, by $z_2 = (s - c_1 z_1)/c_2$. Hence, we can derive the expression $c_1(1 - e^{-z_1}) + c_2(1 - e^{-z_2})$ by z_1 as following.

$$\begin{aligned} \frac{d[c_1(1 - e^{-z_1}) + c_2(1 - e^{z_2})]}{dz_1} &= \frac{d[c_1(1 - e^{-z_1}) + c_2(1 - e^{(c_1 z_1 - s)/c_2})]}{dz_1} \\ &= c_1 e^{-z_1}[1 - e^{z_1 + (c_1 z_1 - s)/c_2}] \ . \end{aligned}$$

Observe that the first part of the derivative is always positive. The second part is a decreasing function of z_1, and therefore, the original function has a global maximum when the right hand side equals 0, which happens when:

$$e^{z_1 + (c_1 z_1 - s)/c_2} = 1 \Leftrightarrow z_1 + (c_1 z_1 - s)/c_2 = 0 \Leftrightarrow z_1 = z_2 \ .$$ □

D Omitted Proofs of Section 4

Lemma 4.6. An α-approximation for $(1, k)$-Submodular Partition implies an α-approximation for Submodular Welfare with k players.

Proof. Given an instance I of Submodular Welfare with k players, let us transform it into an equivalent instance I' of $(1,k)$-Submodular Partition. Let $e_1,\ldots,e_n$ denote the elements of $\mathcal{E}$. The ground set of I' is $\mathcal{N} = \cup_{i=1}^{n}\mathcal{N}_i$, where $\mathcal{N}_i = \{e_i\} \times \{1,2,\ldots,k\}$. The normalized monotone submodular function for I' is $\bar{f}(S) = \sum_{i=1}^{k} f_i(\{e|(e,i) \in S\})$. It is easy to see that $\bar{f}$ is normalized and monotone, however, proving it is submodular requires some work. Consider two sets $A, B \subseteq \mathcal{N}$. Using the submodularity of f, we get:

$$\begin{aligned}
\bar{f}(A) + \bar{f}(B) &= \sum_{i=1}^{k} f_i(\{e|(e,i) \in A\}) + \sum_{i=1}^{k} f_i(\{e|(e,i) \in B\}) \\
&\geq \sum_{i=1}^{k} f_i(\{e|(e,i) \in A\} \cup \{e|(e,i) \in B\}) + \sum_{i=1}^{k} f_i(\{e|(e,i) \in A\} \cap \{e|(e,i) \in B\}) \\
&= \sum_{i=1}^{k} f_i(\{e|(e,i) \in A \cup B\}) + \sum_{i=1}^{k} f_i(\{e|(e,i) \in A \cap B\}) = \bar{f}(A \cup B) + \bar{f}(A \cap B) \ .
\end{aligned}$$

We now show how to transform any solution of I to a feasible set of I', and vice versa. Consider a solution $\mathcal{E}_1,\ldots,\mathcal{E}_k$ of I, and construct from it the set: $S = \cup_{i=1}^{k}\{(e,i)|e \in \mathcal{E}_i\}$. Observe that:

$$\bar{f}(S) = \sum_{i=1}^{k} f_i(\{e|(e,i) \in S\}) = \sum_{i=1}^{k} f_i(\mathcal{E}_i) \ .$$

Thus, S has exactly the same value as $\mathcal{E}_1,\ldots,\mathcal{E}_k$. On the other hand, given a set S which is a feasible solution for I', let us create a solution $\mathcal{E}_1,\ldots,\mathcal{E}_k$ for I.

$$\mathcal{E}_i = \{e|(e,i) \in S\} \ .$$

It is clear by construction of $\bar{f}$ that $\bar{f}(S)$ is equal to the value of the solution $\mathcal{E}_1,\ldots,\mathcal{E}_k$. We are now ready to use the above conversion of I into I' to transform any α-approximation algorithm for $(1,k)$-Submodular Partition into an α-approximation algorithm for Submodular Welfare with k players. Given an instance of Submodular Welfare with k players, transform it into an instance of $(1,k)$-Submodular Partition, find an approximate solution for the resulting instance, and then use the procedure described above to convert it into an approximate solution for the original Submodular Welfare instance. □

Lemma 4.8. There is an α-approximation for Submodular Max-SAT if and only if $(1,2)$-Submodular Partition has.

Proof. Given an instance I of Submodular Max-SAT, let us transform it into an equivalent instance I' of $(1,2)$-Submodular Partition. Let $x_1,\ldots,x_n$ denote the variables of $\mathcal{X}$. The ground set of I' is $\mathcal{N} = \cup_{i=1}^{n}\mathcal{N}_i$, where $\mathcal{N}_i = \{(x_i,0),(x_i,1)\}$. Let $C_{x,v}$ be the set of clauses of I that are satisfied if variable $x \in \mathcal{X}$ is set to value $v \in \{0,1\}$. The normalized monotone submodular function for I' is $\bar{f}(S) = f(\cup_{(x,v)\in S} C_{x,v})$. It is easy to see that $\bar{f}$ is normalized and monotone, however, proving it is submodular requires some work. Consider two sets $A, B \subseteq \mathcal{X} \times \{0,1\}$. Using the submodularity

and monotonicity of f, we get:

$$\begin{aligned}\bar{f}(A)+\bar{f}(B) &= f\left(\bigcup_{(x,v)\in A} C_{x,v}\right)+f\left(\bigcup_{(x,v)\in B} C_{x,v}\right)\\ &\geq f\left(\bigcup_{(x,v)\in A} C_{x,v}\cup\bigcup_{(x,v)\in B} C_{x,v}\right)+f\left(\bigcup_{(x,v)\in A} C_{x,v}\cap\bigcup_{(x,v)\in B} C_{x,v}\right)\\ &\geq f\left(\bigcup_{(x,v)\in A\cup B} C_{x,v}\right)+f\left(\bigcup_{(x,v)\in A\cap B} C_{x,v}\right)=\bar{f}(A\cup B)+\bar{f}(A\cap B)\ .\end{aligned}$$

We now show how to transform any assignment of I to a feasible set of I', and vice versa. Consider an assignment ϕ of I, we construct from it the following set: $S=\{(x,v)|\phi(x)=v\}$. The set of clauses ϕ satisfies is $\cup_{\phi(x)=v}C_{x,v}$, hence, the value of ϕ is $f(\cup_{\phi(x)=v}C_{x,v})$. On the other hand, the value of S is:

$$\bar{f}(S)=f(\cup_{(x,v)\in S}C_{x,v})=f(\cup_{\phi(x)=v}C_{x,v})\ .$$

Thus, S has exactly the same value as ϕ. On the other hand, given a set S which is a feasible solution for I', let us create an assignment ϕ for I.

$$\phi(x)=\begin{cases}0 & \text{if } (x,0)\in S\\ 1 & \text{otherwise}\end{cases}$$

Again, the value of ϕ is $f(\cup_{\phi(x)=v}C_{x,v})$. By the construction of ϕ, $(x,v)\in S$ implies $\phi(x)=v$, although the reverse is not necessarily true. Using this observation and the monotonicity of f, we can upper bound the value of S as following:

$$\bar{f}(S)=f(\cup_{(x,v)\in S}C_{x,v})\leq f(\cup_{\phi(x)=v}C_{x,v})\ .$$

Hence, the value of S is at most the value of ϕ. We are now ready to use the above conversion of I into I' to transform any α-approximation algorithm for $(1,2)$-Submodular Partition into an α-approximation algorithm for Submodular Max-SAT. Given an instance of Submodular Max-SAT, transform it into an instance of $(1,2)$-Submodular Partition, find an approximate solution for the resulting instance, and then use the procedure described above to convert it into an approximate solution for the original Submodular Max-SAT instance.

Consider now an instance I' of $(1,2)$-Submodular Partition with a ground set $\mathcal{N}=\cup_{i=1}^{n}\mathcal{N}_i$ and objective function $f:2^{\mathcal{N}}\rightarrow\mathbb{R}^+$. Let us rename the elements of $\mathcal{N}_i$, and call them $(x_i,0)$ and $(x_i,1)$. We can now construct an instance I of SSAT. There are n variables $x_1,\ldots,x_n$, and each variable x_i has two clauses $C_{x_i,0}$ and $C_{x_i,1}$. The first clause contains only a negative literal of x_i, and the other contains only a positive literal of x_i. The normalized monotone submodular function for I is $\bar{f}(\mathcal{C}')=f(\{(x,v)|C_{x,v}\in\mathcal{C}'\})$. Observe that if one converts I to an instance of $(1,2)$-Submodular Partition using the above method, the resulting instance is exactly I'. Hence, assignments ϕ of I can be transformed into feasible sets S of I', and vice versa, using the methods described above. The rest of the proof of the second direction of the lemma is analog to the first one. □

E Proof of Lemma 3.3

Let A be the set of elements e with $x'_e>x_e$, and let B be the set of elements with $x'_e<x_e$. Recall that $R(x)$ is a random set containing every element $e\in\mathcal{N}$ with probability x_e. For the sake of the

proof, we assume $R(x')$ is formed from $R(x)$ using the following process. Every element of $A - R(x)$ is added to a set D with probability of $1 - (1 - x'_e)/(1 - x_e)$, and every element of $B \cap R(x)$ is added to D with probability $1 - x'_e/x_e$. Then, $R(x')$ is chosen as $R(x) \oplus D$. Observe that every element $e \in \mathcal{N}$ gets into D with probability $|x_e - x'_e| \leq \delta$, independently. We now bound the value of $F(x') - F(x) = \mathbb{E}[f(R(x')) - f(R(x))]$, given various constraints on D.

Lemma E.1. $\sum_{e\in\mathcal{N}} \Pr[D = \{e\}] \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}] \geq \sum_{e\in\mathcal{N}} (x'_e - x_e) \cdot \partial_e F(x) - O(n^3\delta^2) f(OPT)$.

Proof. Let $\mathcal{N}^+$ be the set of elements from $\mathcal{N}$ which have $(x'_e - x_e) \cdot \partial_e F(x) \geq 0$. Observe that for every $e \in \mathcal{N}^+$:

$$\begin{aligned}
(x'_e - x_e) \cdot \partial_e F(z) = |x'_e - x_e| \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}] &= \frac{\Pr[D = \{e\}] \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}]}{\prod_{e'\in\mathcal{N}-e} (1 - |x'_{e'} - x_{e'}|)} \\
&\leq \frac{\Pr[D = \{e\}] \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}]}{\prod_{e'\in\mathcal{N}-e} (1 - \delta)} \\
&= (1 - \delta)^{1-|\mathcal{N}|} \cdot \Pr[D = \{e\}] \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}] \\
&< (1 - \delta)^{-n} \cdot \Pr[D = \{e\}] \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}] \ .
\end{aligned}$$

And the term, $(1 - \delta)^n$ can be lower bounded as following.

$$(1 - \delta)^n = (1 - \delta)^{n\delta/\delta} \geq [e^{-1}(1 - \delta)]^{n\delta} \geq e^{-2n\delta} \geq 1 - 2n\delta \ .$$

Also, for every $e \in \mathcal{N} - \mathcal{N}^+$:

$$\begin{aligned}
(x'_e - x_e) \cdot \partial_e F(x) = |x'_e - x_e| \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}] &= \frac{\Pr[D = \{e\}] \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}]}{\prod_{e'\in\mathcal{N}-e} (1 - |x'_{e'} - x_{e'}|)} \\
&\leq \Pr[D = \{e\}] \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}] \ .
\end{aligned}$$

Combining everything, and recalling that $f(S) \leq W \leq n \cdot f(OPT)$ for every $S \subseteq \mathcal{N}$, we get:

$$\begin{aligned}
\sum_{e\in\mathcal{N}} (x'_e - x_e) \cdot \partial_e F(x) - 2n^3\delta^2 f(OPT) &\leq \sum_{e\in\mathcal{N}} (x'_e - x_e) \cdot \partial_e F(x) - \delta n(2n\delta)W \\
&\leq \sum_{e\in\mathcal{N}} [(x'_e - x_e) \cdot \partial_e F(x) - 2n\delta \cdot |(x'_e - x_e) \cdot \partial_e F(x)|] \\
&\leq \sum_{e\in\mathcal{N}} \Pr[D = \{e\}] \cdot \mathbb{E}[F(x') - F(x) | D = \{e\}] \ . \qquad \square
\end{aligned}$$

Lemma E.2. $\mathbb{E}[F(x') - F(x_t) \mid D = \varnothing] = 0$

Proof. $D = \varnothing$ implies $R(x') = R(x)$. $\square$

Lemma E.3. $\Pr[|D| \geq 2] \cdot \mathbb{E}[F(x') - F(x) \mid |D| \geq 2] \geq -O(n^3\delta^2) f(OPT)$.

Proof. Let us bound the probability that $|D| \geq 2$. Since every element enters into D with probability at most δ:

$$\begin{aligned}
\Pr[|D| \geq 2] &\leq 1 - (1 - \delta)^n - n\delta \cdot (1 - \delta)^{n-1} \leq 1 - (1 + n\delta) \cdot (1 - \delta)^n \\
&\leq 1 - (1 + n\delta) \cdot e^{-n\delta} \cdot (1 - n\delta^2) \leq 1 - (1 + n\delta)(1 - n\delta)(1 - n\delta^2) \leq 2n^2\delta^2 \ .
\end{aligned}$$

Therefore, we get:

$$\Pr[|D| \geq 2] \cdot \mathbb{E}[F(x') - F(x) \mid |D| \geq 2] \geq \Pr[|D| \geq 2] \cdot (-W) \geq -2n^3\delta^2 f(OPT) \ . \qquad \square$$

Lemma 3.3 now follows immediately from the above lemmata and the law of total probability.

F Framework Extension Applications

Theorem 1.4 implies improved approximation ratios for many problems considered in [12]. Table 2 summarizes some of these improvement. A comprehensive list is deferred for a full version of this paper.

Proof of Theorem 1.4. Consider the case of a non-negative and submodular f. Apply Theorem 1.1 with stopping time $T = b$ (recall the $0 \leq b \leq 1$). Then we obtain a fractional solution $x \in b\mathcal{P}$ whose value satisfies: $F(x) \geq (be^{-b} - o(1)) \cdot f(OPT)$. The rest of the proof is exactly as in Theorem 1.8 of [12], thus details are omitted.

For the case of a normalized, monotone and submodular f, apply Theorem 1.2 with stopping time $T = b$ (recall as before that $0 \leq b \leq 1$). Then we obtain a fractional solution $x \in b\mathcal{P}$ whose value satisfies: $F(x) \geq (1 - e^{-b} - o(1)) \cdot f(OPT)$. Again, the rest of the proof is exactly as in Theorem 1.8 of [12]. □

Table 2: A few examples of improved approximation ratios due to Theorem 1.4. All previous results in this table are due to [12]. The notation of ε denotes an arbitrarily small positive constant.

Problem	**This Paper**	**Previous Result**
k Matchoid (linear)	$\frac{2}{e}\frac{1}{k+1}$	$\approx 0.6/k$
k Matchoid (monotone)	$\frac{2}{e}\frac{1}{k+1} - o(1)^*$	$\approx 0.38/k$
k Matchoid (non-monotone)		$\approx 0.19/k$
k Matroid Intersection *and* $O(1)$ Knapsacks (linear)	$\left(\frac{2}{e} - \varepsilon\right)\frac{1}{k+1}$	$\approx 0.6/k$
k Matroid Intersection *and* $O(1)$ Knapsacks (monotone)	$\left(\frac{2}{e} - \varepsilon - o(1)\right)\frac{1}{k+1}^*$	$\approx 0.38/k$
k Matroid Intersection *and* $O(1)$ Knapsacks (non-monotone)		$\approx 0.19/k$

* The $o(1)$ term is with respect to $\min\{n, k\}$, *i.e.*, it vanishes when both n and k are large. The exact function hiding behind the $o(1)$ term depends on the type of the objective function.

G Contention Resolution Schemes

In this appendix we provide improved contention resolution schemes for several matching, schedule and packing problems. These schemes imply improved approximation ratios for these problems using the framework of [12]. Moreover, Theorem 1.4 provides an additional improvement to these approximation ratios. Table 3 summarizes the approximation ratios proved in this appendix. The proofs of some of the lemmata in this appendix are deferred to a full version of this paper.

G.1 The Submodular Independent Set in Interval Graphs Problem

The first problem we consider is the Submodular Independent Set in Interval Graphs problem since many (constrained) submodular maximization problems can be reduced to it. In this problem we are given a set $\mathcal{N}$ of intervals, and a non-negative submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$. A solution $S \subseteq \mathcal{N}$ is feasible if no two intervals in S intersect, *i.e.*, the constraint family $\mathcal{I}$ contains all independent sets of the interval graph defined by $\mathcal{N}$. The goal is to find a feasible solution S which maximizes $f(S)$.

We note that Submodular Independent Set in Interval Graphs is a special case of the *unsplittable flow* problem where the graph is a path and all demands and capacities equal 1. For the unsplittable flow problem with general demands and capacities, [12] presents a $(b, 1 - \rho b)$-balanced contention resolution scheme for some constant ρ. This gives a $(1 - e^{-b})(1 - \rho b)$-approximation for

Table 3: Results proved in Appendix G. Objective functions: NMS - normalized monotone and submodular, NS - non-negative submodular, and L - linear.

Problem	**Objective function**	**This Paper**	**Previous Result**
Submodular Independent Set in Interval Graphs	NMS	$1/4$	$O(1)^*$ [12]
	NS	$1/(2e)$	
Submodular k-Independent Set in Interval Graphs	NMS	$1-e^{-1}-o(1)^{\triangle}$	
	NS	$e^{-1}-o(1)^{\triangle}$	
Submodular Job Interval Selection (single machine)	NMS	$1/4$	—
Submodular Job Interval Selection (k unrelated machines)	NMS	$1/4$	—
Submodular Job Interval Selection (k identical machines)	NMS	$1-e^{-1}-o(1)^{\triangle}$	—
	L	$1-o(1)^{\triangle}$	$1-e^{-1}-o(1)^{\triangle}$ [6]
Submodular Multiple Knapsack	NMS	$1/4^{\square}$	—
Submodular Multiple Knapsack (identical knapsack sizes)	NMS	$1-e^{-1}-o(1)^{\triangle,\square}$	—
Submodular Broadcast Scheduling	NMS	$1/4$	—
Submodular Matching Scheduling (edge degree $\leq k$)	NMS	$\frac{k}{e(k+1)^2}$	—

* The exact constant was not calculated by [12], but it is inferior to our corresponding results.
$^{\triangle}$ The $o(1)$ term is with respect to $\min\{n,k\}$, *i.e.*, it diminishes when both n and k are large.
$^{\square}$ Requires pseudo polynomial time. It is possible to get a polynomial time algorithm at the cost of some loss in the approximation ratio. For details, see the theorems below.

the monotone case and a $(be^{-b}-o(1)(1-\rho b)$-approximation for the non-monotone case.[8] We show that Submodular Independent Set in Interval Graphs admits an improved monotone (b,e^{-b})-balanced contention resolution scheme for every $b\in(0,1]$, and thus, has better approximation ratios than the ones known for unsplittable flow. We get $(1-e^{-b})e^{-b}$ and $\left(be^{-2b}-o(1)\right)$ approximation for the monotone and non-monotone cases, respectively.

Following is the algorithm for Submodular Independent Set in Interval Graphs induced by our contention resolution scheme. We give the complete algorithm, instead of the contention resolution scheme alone, because we believe the algorithm is more natural looking this way.

The first step of the algorithm is the relaxation solving and re-normalization steps of the contention resolution framework. The relaxation we use is the natural relaxations of the problem. The second step of the algorithm corresponds to the sampling step of the framework. We note that the probability of choosing an element into R in Line 2 is not as in [12]. However, since $1-e^{-x}\leq x$, one can think of it as the sampling step of the framework, followed by a second sampling step at the begining of the contention resultion scheme. Exact details are deferred to a full version of this paper.

Lemma G.1. *For every $b\in(0,1]$, Algorithm 3 is a monotone (b,e^{-b})-balanced contention resolution scheme for Submodular Independent Set in Interval Graphs.*

[8] In both cases, the approximation ratios presented are the ones achieved using our measured continuous greedy algorithm, and Theorem 1.4. These ratios are somewhat better than the original ones given by [12].

Algorithm 3: CR Scheme for Submodular Independent Set in Interval Graphs($\mathcal{N}, f, b$)

```
// Computing Fractional Solution
```
1 Use the *measured continuous greedy* algorithm with stopping time $T = b$ to obtain x.

```
// Sampling
```
2 Sample R where each interval $e \in \mathcal{N}$ is chosen independently with probability $1 - e^{-x_e}$.

```
// Diluting
```
3 For every $e \in \mathcal{N}$, mark interval e for deletion if there is a different interval $e' \in R$ that intersects the starting point of e.

```
// Output
```
4 Remove all marked intervals from R, and let S be the set of remaining intervals.

5 Output S.

Proof. First, we prove that for every interval $e \in \mathcal{N}$:

$$\Pr[e \in S] \geq \left(1 - e^{-x_e}\right) e^{-b} \ .$$

Let $\mathcal{L}_e$ be the set of intervals that intersect the starting point of e, excluding e itself. Since $x \in b\mathcal{P}$ (due to Theorems 1.1 and 1.2): $\sum_{e' \in \mathcal{L}_e} x_{e'} \leq b - x_e$. Therefore,

$$\begin{aligned}\Pr[e \in S] &= \left(1 - e^{-x_e}\right) \cdot \prod_{e' \in \mathcal{L}_e} e^{-x_{e'}} = \left(1 - e^{-x_e}\right) \cdot e^{-\sum_{e' \in \mathcal{L}_e} x_{e'}} \\ &\geq \left(1 - e^{-x_e}\right) \cdot e^{-(b - x_e)} \geq \left(1 - e^{-x_e}\right) \cdot e^{-b} \ .\end{aligned}$$

Since $e \in S$ implies $e \in R$, we get:

$$\Pr[e \in S \mid e \in R] = \frac{\Pr[e \in S \wedge e \in R]}{\Pr[e \in R]} = \frac{\Pr[e \in S]}{\Pr[e \in R]} \geq \frac{\left(1 - e^{-x_e}\right) e^{-b}}{1 - e^{-x_e}} = e^{-b} \ .$$

The monotonicity of contention resolution scheme is clear from the description of Algorithm 3. □

Corollary G.2. *Submodular Independent Set in Interval Graphs has a $1/4$-approximation if f is normalized, monotone and submodular, and a $(1/(2e) - o(1))$-approximation if f is non-negative and submodular.*

Proof. For the former case apply Theorems 1.3 and 1.4 along with Lemma G.1 with $b = \ln 2$ to obtain an approximation of $1/4$. For the latter case apply Theorem 1.1 and 1.4 along with Lemma G.1 with $b = 1/2$ to obtain an approximation of $(1/(2e) - o(1))$. □

We consider also a useful variation of Submodular Independent Set in Interval Graphs in which a valid solution might contain up to k intervals covering each time point on the line (as opposed to just a single interval). We denote this problem as the Submodular k-Independent Set in Interval Graphs problem. Algorithm 3, and the contention resolution scheme implying it, can still be applied with a few minor changes.

The $o(1)$ term in the next lemma is with respect to $\min\{k, n\}$, hence, it diminishes when both k and n are large.

Lemma G.3. *The above algorithm with $b = 1 - O\left(\sqrt{\log k / k}\right)$ gives a $(1 - 1/e - o(1))$-approximation for Submodular k-Independent Set in Interval Graphs for normalized monotone submodular f, and a $(1/e - o(1))$-approximation for non-negative submodular f.*

Algorithm 4: CR Scheme for Submodular k-Independent Set in Interval Graphs$(\mathcal{N}, f, k, b)$

```
// Computing Fractional Solution
```
1 Use the *measured continuous greedy* algorithm with stopping time $T = b$ to obtain x.

```
// Sampling
```
2 Sample R where each interval $e \in \mathcal{N}$ is chosen independently with probability x_e.

```
// Diluting
```
3 For every $e \in \mathcal{N}$, mark interval e for deletion if there are at least k other intervals $e' \in R$ that intersect the starting point of e.

```
// Output
```
4 Remove all marked intervals from R, and let S be the set of remaining intervals.

5 Output S.

G.2 The Submodular Job Interval Selection Problem

The Submodular Job Interval Selection problem is defined as follows. We are given a set $\mathcal{N}$ of n jobs, and each job $e \in \mathcal{N}$ is associated with a set $\mathcal{J}_e$ of possible intervals. Additionally, a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ over the jobs is given (note that f is not defined over the intervals, but over the jobs). A feasible schedule is a subset $S \subseteq \cup_{e \in \mathcal{N}} \mathcal{J}_e$ of intervals, such that no two intervals in S intersect. A job e is *scheduled* in S if S contains an interval from $\mathcal{J}_e$. The goal is to find a feasible schedule S that maximizes the value of f over the set of jobs scheduled in S. We show that Submodular Job Interval Selection can be reduced via an approximation preserving reduction to Submodular Independent Set in Interval Graphs. This reduction works only for normalized *monotone* submodular objective functions, hence, we do not consider general non-negative submodular objectives.

Lemma G.4. *There is an approximation preserving reduction from Submodular Job Interval Selection to Submodular Independent Set in Interval Graphs in case f is normalized, monotone and submodular.*

We consider three variants of Submodular Job Interval Selection. First, we consider the case where there are k *identical* machines. Second, we consider the case where there are k *unrelated* machines. Third and last, we consider the case where the objective function is a linear function, and there are k identical machines.

G.2.1 k Identical Machines

An instance of the k identical machines variant is an instance of Submodular Job Interval Selection with the following difference. A schedule S is a k-tuple $(S_1, S_2, \ldots, S_k)$, where every S_i is a set of intervals. S is a *feasible* schedule if no two intervals of the same set S_i intersect. As before, the goal is to maximize the value of f over the set of jobs scheduled in S.

Lemma G.5. *There is an approximation preserving reduction from Submodular Job Interval Selection with k identical machines to Submodular k-Independent Set in Interval Graphs in case f is normalized, monotone and submodular.*

G.2.2 k Unrelated Machines

An instance of the k unrelated machines variant is an instance of Submodular Job Interval Selection with the following difference. A job $e \in \mathcal{N}$ is associated with k sets of intervals: $\mathcal{J}_{e,1}, \mathcal{J}_{e,2}, \ldots, \mathcal{J}_{e,k}$, where set $\mathcal{J}_{e,i}$ is the collection of allowed intervals of job e on the i^{th} machine. A schedule S is a k-tuple $(S_1, S_2, \ldots, S_k)$, where every S_i is a set of intervals allowed for the i^{th} machine. Again, S is a *feasible* schedule if no two intervals of the same set S_i intersect.

Lemma G.6. *There is an approximation preserving reduction from* Submodular Job Interval Selection *with k unrelated machines to* Submodular Job Interval Selection *in the case that f is normalized, monotone and submodular.*

G.2.3 k Identical Machines with Linear Objective

An instance of this variant is an instance of the k identical machines variant, with a linear objective function. Since the objective is linear, one can use for this variant a linear programming formulation instead of the multilinear extension formulation. The rounding algorithm is that same as in Algorithm 4, with the following modification. In the sampling step, for every job e, the algorithm selects at most one interval $I \in \mathcal{J}_e$, *i.e.*, the choice of which interval of $\mathcal{J}_e$ to select is done using dependent rounding. The analysis of this variant is different because it does not go through Submodular k-Independent Set in Interval Graphs, and thus, it does not follow from [3, 12].

The $o(1)$ term in the next lemma is with respect to $\min\{k, n\}$, hence, it diminishes when both k and n are large.

Lemma G.7. *There is a polynomial time algorithm that achieves $(1 - o(1))$-approximation for the k identical machines variant with a linear objective variant.*

Observe that this is a linear problem for which we improve the best known approximation ratio. The previously best approximation ratio for this problem approached $1 - e^{-1}$ for large k values [4].

G.3 The Submodular Multiple Knapsack Problem

The Submodular Multiple Knapsack problem is defined as following. We are given a collection $\mathcal{N}$ of n elements and k knapsacks, where the size of the i^{th} knapsack is $B_i \in \mathbb{N}$. We note that the number of knapsacks, k, might not be a constant. Each element $e \in \mathcal{N}$ has a given size $s_e \in \mathbb{N}$. In addition, we are also given a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ defined over the elements. A feasible packing S is a k-tuple $(S_1, S_2, \ldots, S_k)$ such that the total size (*i.e.*, the sum of the sizes) of the elements in each set S_i is at most B_i. An element e is *packed* by S if there exists an S_i such that $e \in S_i$. The goal is to find a feasible packing S that maximizes the value of f over the set of elements packed by S. Note that Submodular Multiple Knapsack differs from the usual k-knapsack constraint, discussed earlier in this work, in the sense that here we are asked to pack each element to up to one of several possible knapsack, while the problem of k-knapsack constraints ask to pack each element either to no knapsack or to all of them at the same time.

We show that Submodular Multiple Knapsack can be reduced via an approximation preserving reduction to Submodular Independent Set in Interval Graphs (again, the reduction works only for normalized, monotone and submodular objective functions). However, the time complexity of this reduction depends on the sizes of the knapsacks. In order to get an approximation that runs in polynomial time regardless of the sizes of the elements and knapsacks, additional techniques has to be applied. The $o(1)$ term in Lemma G.9 is with regard to $\min\{n, k\}$, *i.e.*, it diminishes when both n and k are large.

Lemma G.8. *There is an approximation preserving reduction from* Submodular Multiple Knapsack *to* Submodular Independent Set in Interval Graphs *(for normalized, monotone and submodular objectives). The time complexity of this reduction is pseudo-polynomial.*

Lemma G.9. *There is a polynomial time $(1/13 - o(1))$-approximation algorithm for* Submodular Multiple Knapsack *with a normalized, monotone and submodular objective function.*

We also consider a variant of Submodular Multiple Knapsack where all knapsacks have equal size B. For this case, better approximations are possible, as shown by the following two lemmata.

Lemma G.10. *There exists an approximation preserving reduction from* Submodular Multiple Knapsack *with identical knapsack sizes, to* Submodular k-Independent Set in Interval Graphs *(for the case of normalized, monotone and submodular objective functions). The time complexity of this reduction is pseudo polynomial.*

Lemma G.11. *There is a polynomial time $((e-1)/(4e-1)-o(1))$-approximation algorithm for Submodular Multiple Knapsack with identical knapsack sizes and a normalized, monotone and submodular objective function.*

Note that Lemma G.10 implies an approximation ratio of about $1-e^{-1}$ in pseudo polynomial time for Submodular Multiple Knapsack with many identical knapsacks. This is the same approximation ratio achieved by [46] for a single knapsack, and is known to be tight [8].

G.4 The Submodular Broadcast Scheduling Problem

The Submodular Broadcast Scheduling problem is defined as follows. We are given a set $\mathcal{K}$ of n pages, and a set $\mathcal{N}$ of requests. Each page $P \in \mathcal{K}$ is associated with a set $\mathcal{J}_P$ of intervals, and each request $e \in \mathcal{N}$ is associated with a page P_e and a subset $\mathcal{J}_e \subseteq \mathcal{J}_{P_e}$ of intervals of P_e. Additionally, we are given a normalized monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$. A feasible schedule is a set S of intervals, such that no two intervals intersect. A request e is *fulfilled* by schedule S if S contains an interval from $\mathcal{J}_e$. The goal is to find a feasible schedule S maximizing the value of f over the set of requests fulfilled by S.

One can assume without loss of generality that there is only one page $\bar{P}$ in any instance of Submodular Broadcast Scheduling. The reduction goes as following. Replace all pages with a new page $\bar{P}$. The set of intervals of $\bar{P}$ is $\mathcal{J}_{\bar{P}} = \cup_{P \in \mathcal{K}} \mathcal{J}_P$. All requests $e \in \mathcal{N}$ are now associated with $\bar{P}$, but keep the same intervals set $\mathcal{J}_e \subseteq \mathcal{J}_{P_e} \subseteq \mathcal{J}_{\bar{P}}$. Neither the definition of feasible schedules, nor the objective of the problem, are changed by the introduction of $\bar{P}$. The following lemma is proved using this reduction.

Lemma G.12. *There exists an approximation preserving reduction from Submodular Broadcast Scheduling to Submodular Independent Set in Interval Graphs (for the case of a normalized monotone and submodular f).*

Lemma G.12 imply 4-approximation for Submodular Broadcast Scheduling. We would like to emphasis that this is the best approximation ratio known also for the linear variant of Submodular Broadcast Scheduling [5].

G.5 The Submodular Matching Scheduling Problem

The Submodular Matching Scheduling problem is defined as follows. We are given a multigraph $G = (V, E)$ of edge degree at most k and a set $\mathcal{J}_e$ of intervals *tuples* for every edge $e \in E$. Given an intervals tuple $J \in \mathcal{J}_e$ for some edge e, there is one interval in J for every node $v \in e$. For an intervals tuple J, let $E(J)$ be the edge with which the tuple J is associated, and let $J(u)$ denote the interval in J associated with u (assuming $u \in E(J)$). A feasible matching schedule $S \subseteq \cup_{e \in E} \mathcal{J}_e$ is a set of intervals tuples such that for every two intervals tuples $J_1, J_2 \in S$, if $E(J_1)$ and $E(J_2)$ share a common vertex u, then $J_1(u)$ and $J_2(u)$ do not intersect. An edge e is *covered* by a matching schedule if $\mathcal{J}_e \cap S \neq \emptyset$. The objective is to find a matching schedule S maximizing a normalized monotone submodular function $f : 2^E \to \mathbb{R}^+$ over the set of edges that are covered by S.

Informally, a matching schedule is a matching in which every edge is also assigned a tuple of intervals, one for each end point, and we allow multiple edges to hit the same node as long as their associated intervals do not intersect. This problem has several interesting special cases:

1. The multigraph is a graph, *i.e.*, $k = 2$, and every intervals tubple contains two identical intervals. In this case each edge e has a given list of allowed intervals $\mathcal{J}_e$. A feasible schedule chooses a subset E' of edges, and assigns an interval from $\mathcal{J}_e$ to every edge $e \in E'$ in such a way that the intervals assigned to edges intersecting a common vertex are disjoint (*i.e.*, they do not intersect). This problem models a situation where nodes wish to transmit a set of messages among themselves, however, a node can participate at every point in time in one

transmission only. The goal is to maximize a normalized monotone submodular function of the transmitted messages.

2. The input is a graph in which every node $v \in V$ has a budget $B(v) \in \mathbb{N}$ and every edge $e \in E$ has a weight w_e. A feasible schedule is a subset of edges such that for every vertex v, the total weight of chosen edges hitting v does not exceed $B(v)$. The goal is to maximize a normalized monotone submodular function of the chosen edges.
   This problem has an approximation preserving pseudo polynomial time reduction to Submodular Matching Scheduling. The reduction can be made polynomial, at the cost of some loss in the approximation ratio which depends on k (details are deferred to a full version of this paper).

Lemma G.13. *One can assume without loss of generality that for every edge $e \in E$, $|\mathcal{J}_e| = 1$.*

Proof. Replace every edge e with $|\mathcal{J}_e|$ parallel edges: $e_1, e_2, \ldots, e_{|\mathcal{J}_e|}$, and assign a unique interval pair from $\mathcal{J}_e$ to every edge e_i. The new objective function $\bar{f}$ is defined as follows: $\bar{f}(S) = f(\{e \in E | \exists_i J_{e_i} \subseteq S\})$. Clearly $\bar{f}$ is also a normalized monotone submodular function. Observe that any feasible matching schedule before the reduction is also a feasible matching schedule after the reduction, and vice versa. Moreover, it is easy to see that the reduction preserves the value of the objective function because f is not effected by the size of $\mathcal{J}_e \cap S$ as long as this quantity is positive. □

After the reduction presented by Lemma G.13, the problem Submodular Matching Scheduling can viewed as the intersection of n Submodular Independent Set in Interval Graphs, one for each node. Hence, we can apply the contention resolution scheme of Submodular Independent Set in Interval Graphs for each node separately, and get a scheme for Submodular Matching Scheduling. By Lemma 1.5 of [12], this scheme is a (b, e^{-bk})-monotone balanced contention resuoltion scheme for Submodular Matching Scheduling.

Corollary G.14. *There is a $ke^{-1}(k+1)^{-2}$ approximation algorithm for Submodular Matching Scheduling.*

Proof. Using the contention resolution scheme described above, Theorem 1.4 and $b = \ln(1 + 1/k)$, we get an algorithm for Submodular Matching Scheduling with an approximation ratio of:

$$(1 - e^{-b}) \cdot e^{-bk} = \left(1 - \frac{k}{k+1}\right) \cdot \left(\frac{k}{k+1}\right)^k = \frac{1}{k+1} \cdot \left(1 - \frac{1}{k+1}\right)^k \geq \frac{k}{e(k+1)^2} \ . \qquad \square$$