Charles University
Faculty of Mathematics and Physics

Doctoral Thesis

Submodularity in Combinatorial Optimization

Jan Vondrák

Department of Applied Mathematics

Malostranské nám. 25
Prague, Czech Republic

2007

Preface

This a continually updated version of my second PhD thesis, submitted to the Department of Applied Mathematics at Charles University. It serves mainly for myself as a compendium of my research on submodular optimization and related problems, at Microsoft Research and Princeton University between 2005-07.

Princeton, November 8, 2007

Acknowledgement

This thesis would not have seen the light of day without the support and perseverance of Martin Loebl, and significant assistance of several others. I would like to thank first and foremost Uriel Feige for introducing me to this subject and being a wonderful guide. I am also grateful to Chandra Chekuri for enlightening conversations and bringing my attention to one of the topics presented here. Let me not forget Gruia Calinescu, Mohammad Mahdian, Vahab Mirrokni, Benny Sudakov and others who have been very helpful in their discussions. Most of this work has been published together with some of the co-authors mentioned above; the copyright belongs to the respective periodicals.

Contents

Chapter 1

Introduction

This thesis deals with several problems either explicitly or implictly involving the notion of *submodularity*. Submodularity has attained an important role in theoretical computer science, due to its beneficial properties in the design of optimization algorithms. On the other hand, submodularity is not an artifical mathematical concept. It appears naturally in many applications, either as a structural property of combinatorial problems or as a natural assumption on certain valuation functions. In economics, submodularity has been known for a long time, although under different names (*diminishing returns*, *decreasing marginal values*, etc.). Recently, submodularity re-appeared in the context of combinatorial auctions. It seems like a fortunate phenomenon that this property imposes a structure which allows much stronger algorithmic results than we would be able to achieve without it.

Before proceeding to formal definitions, we discuss briefly the nature of the problems that we study here. Typically, they involve submodular functions $f : 2^X \to \mathbb{R}$ defined on a finite set of *items* X. The natural interpretation of $f(S)$ is the *value* of a subset of items S. As we mentioned, the submodularity property has an economic interpretation meaning that additional items have less and less value, as the set we possess grows. However, the function $f(S)$ can also have a more abstract meaning, derived from a problem not obviously involving submodularity.

We begin by studying the problem of maximizing a single submodular function without any constraints. This problem already captures various applications. More importantly, our investigation of this problem reveals structural properties of submodular functions which turn out to be very useful in further applications. We study the basic problem of submodular maximization in Chapter 2. We proceed to variants of submodular maximization under various constraints. The most flexible and interesting type of constraint seems to be a *matroid constraint*. We study the problem of maximizing a submodular function subject to a matroid constraint in Chapter 3. This framework captures a number of special cases of interest, which we examine in more detail. Two specific problems that we focus on are:

- **The Submodular Welfare Problem**: the problem of allocating items to multiple players, each of which values subsets of items according to a submodular "utility function". This problem appears as a fundamental building block in the framework of combinatorial auctions. We give more details in Chapters 4 and 5.
- **The General Assignment Problem**: the problem of assigning items to bins of limited capacity, where both "values" and "sizes" of items can depend on the bin where the item is placed. This problem is a generalization of many assignment/scheduling problems. For details, we refer the reader to Chapter 6.

We study these problems from the point of view of *approximation algorithms*. Since all the problems mentioned above are NP-hard, we settle for a less ambitious goal than solving these problems optimally. We seek a solution with an α-approximation guarantee, meaning that the value of our solution is at least $\alpha \cdot OPT$ where $\alpha < 1$ is the approximation factor and OPT is the value of the optimal solution. We are also looking for *hardness results*, either information-theoretic or complexity-theoretic. Such results indicate that a certain approximation algorithm is either downright impossible, or it would have unlikely consequences in complexity theory. In either case, we obtain limits on the positive results that one can aspire to achieve.

1.1 Definitions

We begin with the definition of submodularity.

Definition 1.1 (submodularity). *A function $f : 2^X \to \mathbb{R}$ is submodular if for any $S, T \subseteq X$,*

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T).$$

An alternative definition is in terms of marginal values. We do not prove the equivalence here; we refer the reader to [51].

Definition 1.2 (marginal values). *The marginal value of an item j added to a set S is defined as $f_S(j) = f(S \cup \{j\}) - f(S)$. Submodularity can be alternatively defined by $f_S(j) \geq f_T(j)$ for all $S \subseteq T, j \notin T$.*

Next, we define two additional properties that we will consider.

Definition 1.3 (monotonicity). *A function $f : 2^X \to \mathbb{R}$ is monotone if for any $S \subseteq T \subseteq X$,*

$$f(S) \leq f(T).$$

Definition 1.4 (symmetry). *A function $f : 2^X \to \mathbb{R}$ is symmetric if for any $S \subseteq X$,*

$$f(S) = f(X \setminus S).$$

Note that a function which is both monotone and symmetric would have to be constant. Hence, it make sense to consider monotonicity and symmetricity only separately.

Examples.

- An example of a monotone submodular function is a *coverage-type* function. For a collection of finite sets $\{A_j\}_{j \in X}$, we define

  $$f(S) = \left| \bigcup_{j \in S} A_j \right|.$$

  Then $f(S)$ is monotone and submodular. Finding the maximum of $f(S)$ over sets of size k is the *Max k-cover* problem.

- An example of a symmetric submodular function is a *cut-type* function. For a graph G, let $\delta(S)$ denote the number of edges with exactly one endpoint in S. Then $\delta(S)$ is symmetric and submodular. The problem of maximizing $\delta(S)$ is the *Max Cut* problem.

- An example of a submodular function that is neither symmetric nor monotone is a *directed cut-type* function. For a directed graph D, let $\delta(S)$ denote the number of arcs pointing from S to $\bar{S}$. Then δ is submodular. The problem of maximizing $\delta(S)$ in directed graphs is the *Max Di-Cut* problem.

- Another example of a non-monotone and non-symmetric submodular function can be obtained from any monotone submodular function $f(S)$ (such as a coverage-type function) by subtracting a *linear function* $w(S) = \sum_{j \in S} w_j$. Then $g(S) = f(S) - w(S)$ is again submodular but not necessarily monotone. In fact, it can be verified that every submodular function can be written in this form.

Another important concept that we use is that of a *matroid.*

Definition 1.5. *A matroid $\mathcal{M}$ is a pair $(X, \mathcal{I})$ where X is a ground set of elements and $\mathcal{I}$ is a collection of subsets of X (that we call "independent"), satisfying two axioms:*

1. $B \in \mathcal{I}, A \subset B \Rightarrow A \in \mathcal{I}$.
2. $A, B \in \mathcal{I}, |A| < |B| \Rightarrow \exists x \in B \setminus A; A \cup \{x\} \in \mathcal{I}$.

There are many examples of matroids. The matroids of particular interest here will be two very simple special cases:

- A *uniform matroid* is such that independent sets are all sets of size at most k for some $k \geq 1$.
- A *partition matroid* is such that X is partitioned into ℓ sets $X_1, X_2, \ldots, X_\ell$ with associated integers $k_1, k_2, \ldots, k_\ell$, and a set $I \subseteq X$ is independent iff $|I \cap X_i| \leq k_i$ for each i.

Matroids give rise to another example of a submodular function, the *rank function of a matroid.*

Definition 1.6. *For a matroid $\mathcal{M} = (X, \mathcal{I})$, the associated rank function is defined as*

$$r(S) = \max\{|I| \mid I \subseteq S, I \in \mathcal{I}\}.$$

It is known that the rank function of any matroid is monotone and submodular.

Next, we define the notion of a *matroid polytope* which is useful in applying linear programming to optimization over matroids.

Definition 1.7. *For a matroid $\mathcal{M} = (X, \mathcal{I})$, the matroid polytope $P(\mathcal{M})$ is the convex hull of $\{\chi_I \mid I \in \mathcal{I}\}$ where each χ_I is the characteristic vector of an independent set I.*

The matroid polytope has a nice description in terms of linear inequalities, discovered by Edmonds [15]:

$$P(\mathcal{M}) = \{x \in \mathbb{R}_+^X \mid \forall S \subseteq X; \sum_{j \in S} x_j \leq r(S)\}$$

where $r(S)$ is the rank function of $\mathcal{M}$. The number of inequalities here is exponential, but the submodular structure of $r(S)$ allows us to implement a separation oracle and hence optimize over $P(\mathcal{M})$ in polynomial time. (See [51] for more details.)

Representation of submodular functions. Our goal here is to study problems whose input involves one or more submodular functions. This raises the issue of how such a function should be represented. If we were to list the values for all possible sets, this would occupy space exponentially large in $|X|$. We would prefer to avoid this, and there seem to be two approaches to get around this obstacle.

1. **Compact representation.** In some cases, a submodular function can be represented by a structure of polynomial size - e.g., by a collection of sets in the case of a coverage-type function, or by a graph in the case of a cut-type function. In such cases, it is natural to specify the function by giving the respective combinatorial structure on the input explicitly. However, an efficient encoding is not possible for a general submodular function, since it is known that the number of submodular functions is doubly exponential in $|X|$, even if the values are integers in $\{0, \ldots, |X|\}$.

2. **Oracle access.** The most general access to a function is through an oracle, or "black box", which answers queries about the function.

   - **Value oracle.** The most basic query is: *What is the value of $f(S)$?* An oracle answering such queries is called a *value oracle*.
   - **Demand oracle.** Sometimes, a more powerful oracle is considered, which can answer queries of the following type: *Given an assignment of prices to items $p : X \to \mathbb{R}$, what is $\max_{S \subseteq X}(f(S) - \sum_{j \in S} p_j)$?* Such an oracle is called a *demand oracle*.

We consider algorithms that run in polynomial time. In the case of oracle access, this includes the requirement that the number of queries be polynomially bounded. We will see that this already imposes limits on how good an approximation we can obtain.

In oracle models, we consider two types of algorithms.

1. **Nonadaptive algorithms.** These algorithms are allowed to issue a polynomial number of queries and then the answers can be processed by a polynomial-time computation. Randomization is allowed but queries must not depend on the answers of previous queries.

2. **Adaptive algorithms.** These algorithms are allowed to issue arbitrary queries in the process of a polynomial-time computation. Unless otherwise noted, this is the default choice for our algorithms.

1.2 The problems and our results

Here we summarize the problems studied in this thesis, and we describe our results in the context of previous work.

1.2.1 Nonnegative submodular maximization

Problem: *Given a submodular function $f : 2^X \to \mathbb{R}_+$ with value-oracle access, maximize $f(S)$ over all subsets $S \subseteq X$.*

This is perhaps the most basic maximization problem involving submodular functions. We remark that the analogous *submodular minimization* problem can be solved exactly in polynomial time [53, 21]. Unlike submodular minimization, it is known that submodular maximization is NP-hard; for instance, the NP-hard Max Cut problem is a special case [33]. Hence, we seek to design approximation algorithms for this problem.

We consider only the value oracle model here; a demand oracle would make the problem trivial. Our only additional assumption about the function is nonnegativity. Note that any function on a finite set can be made nonnegative by adding a sufficiently large positive constant. This does not change the optimal solution; however, it changes the problem in terms of approximation guarantees. By a shifting argument, verifying whether the maximum of a submodular function is greater than zero or not is NP-hard. Thus, no approximation algorithm can be found for the maximization problem without any restrictions, unless P=NP.

Previous work. The maximization problem for general submodular functions has been studied in the operations research community. Many efforts have been focused on designing heuristics for this problem, including data-correcting search methods [27, 28, 34], accelatered greedy algorithms [48], and polyhedral algorithms [40]. Prior to our work, to the best of our knowledge, no guaranteed approximation factor was known for the general problem of maximizing non-monotone submodular functions.

A renowned special case is the Max Cut problem, for which a greedy $\frac{1}{2}$-approximation was the best known algorithm for a long time. A breakthrough came in 1995 when this was improved to a 0.878-approximation by Goemans and Williamson [26]. Their approach introduced semidefinite programming to the computer science community and led to advances on many other optimization problems. For the Max Di-Cut problem, a 0.874-approximation algorithm was designed in [16, 39], also using semidefinite programming. The approximation factor for Max Cut has been proved optimal, assuming the

Unique Games Conjecture [35, 43]. It should be noted that the best known combinatorial algorithms for Max Cut and Max Di-Cut achieve only a $\frac{1}{2}$-approximation, which is trivial for Max Cut but not for Max Di-Cut [31]. All these algorithms allow nonnegative edge weights. With possibly negative edge weights, the Max Cut problem becomes much more difficult and no constant factor approximation is likely to exist [37].

A number of other special cases have been investigated as well, such as Max Cut in hypergraphs and Max SAT with no mixed clauses. For all these special cases, there are approximation algorithms significantly beating the factor of $\frac{1}{2}$. Tight results are known for Max Cut in k-uniform hypergraphs for any fixed $k \geq 4$ [32, 29]. Here, the optimal approximation factor $(1 - 2^{-k+1})$ is achieved by a random solution (and the same result holds for Max $(k-1)$-SAT with no mixed clauses [29, 30]). The lowest approximation factor $(\frac{7}{8})$ is achieved for $k = 4$; for $k < 4$, better than random solutions can be found by semidefinite programming.

Our results. We design several constant factor approximation algorithms for this problem. First, we show that simply choosing a random set gives an expected value of at least $\frac{1}{4}OPT$. In the special case of symmetric submodular functions, we show that a random set already gives at least $\frac{1}{2}OPT$. In the general case, we design improved algorithms using local search: a deterministic $\frac{1}{3}$-approximation, and a randomized $\frac{2}{5}$-approximation. We also design a nonadaptive randomized $\frac{1}{3}$-approximation. Perhaps the most noteworthy of our algorithms is the randomized $\frac{2}{5}$-approximation. It proceeds by searching for a local optimum of a function $\Phi(S)$ derived from $f(S)$ by taking $\Phi(S) = \mathbf{E}[f(R)]$ for a random set R sampled from a certain distribution based on S. We call this a *smooth local search* algorithm.

In the symmetric case, a $\frac{1}{2}$-approximation can be also achieved by a deterministic algorithm. The threshold of $\frac{1}{2}$ seems significant: If the $\frac{1}{2}$-approximation should be improved, it would have to be either with the help of semidefinite programming, or using an entirely new method (which would also be a new way to beat $\frac{1}{2}$ for Max Cut). However, we prove that improving $\frac{1}{2}$ is impossible in the value oracle model. We prove that a $(\frac{1}{2} + \epsilon)$-approximation for any fixed $\epsilon > 0$ would require exponentially many value queries, regardless of our computational power. This result is independent of the $P \neq NP$ hypothesis.

We also obtain NP-hardness results in the case where the submodular function is represented explicitly on the input, as a combination of elementary submodular functions of constant size. (All special cases such as Max Cut fall in this category.) We prove that a $(\frac{3}{4} + \epsilon)$-algorithm for maximizing compactly represented submodular functions (or a $(\frac{5}{6}+\epsilon)$-algorithm for symmetric submodular functions) for any fixed $\epsilon > 0$ would imply $P = NP$. This shows that although $\frac{1}{2}$ might not be the optimal answer for compactly represented functions, the problem is still more difficult than Max Cut or other special variants where approximations better than $\frac{3}{4}$ are known.

Summary. The following table summarizes our results.

Model	RS	NA	DET	RAND	VQ-hard	NP-hard
Symmetric	1/2	1/2	1/2	1/2	$1/2+\epsilon$	$5/6+\epsilon$
General	1/4	1/3	1/3	2/5	$1/2+\epsilon$	$3/4+\epsilon$

The types of algorithms considered here are: RS = random set, NA = non-adaptive (possibly randomized), DET = deterministic adaptive, and RAND = randomized adaptive. The hardness results are either in the value query model (VQ-hard) or in the compact representation model (NP-hard).

We present these results in Chapter 2. They also appeared in [23], a joint work with Uriel Feige and Vahab Mirrokni.

1.2.2 Submodular maximization under a matroid constraint

Problem: *Given a monotone submodular function $f : 2^X \to \mathbb{R}_+$ and a matroid on $\mathcal{M} = (X, \mathcal{I})$, maximize $f(S)$ over all independent sets $S \in \mathcal{I}$.*

Again, we work in the value oracle model here. We also assume that we have a membership oracle for $\mathcal{M}$ available. This problem has a long history, starting with the seminal paper of Nemhauser, Wolsey and Fisher in 1978 [45]. Even the very special case where f is a coverage-type function and $\mathcal{M}$ is a uniform matroid (where independent sets are exactly those of size at most k) is an interesting problem, the Max k-cover problem. A natural algorithm for this problem is the following.

Greedy Algorithm.

```
Set $S := \emptyset$.
While ($S$ is not a maximal independent set)
  { Compute $f_S(j) = f(S \cup \{j\}) - f(S)$ for all $j \notin S$
      such that $S \cup \{j\}$ is still independent.
   Out of these elements, pick one maximizing $f_S(j)$ and include it in $S$. }
Output $S$.
```

Note that when f is a linear function ($f(S) = \sum_{j \in S} w_j$), this algorithm is just the greedy algorithm for finding a maximum-weight independent set in a matroid, which returns the optimal solution. However, this is not the case for general submodular functions.

Previous work. The Greedy Algorithm provides a $(1-1/e)$-approximation [1] for the Max k-cover problem and more generally for submodular maximization subject to $|S| \leq k$ [45]. It was known already by Nemhauser and Wolsey that the factor of $1-1/e$ is optimal in the value oracle model, if only a polynomial number of queries is allowed [47]. Moreover, it was proved by Feige [17] that this approximation factor is best possible even for Max k-cover unless $P = NP$.

For an arbitrary matroid constraint, Nemhauser, Wolsey and Fisher proved that the Greedy Algorithm provides a $\frac{1}{2}$-approximation. It is easy to observe that this algorithm does not yield a factor better than $\frac{1}{2}$, and it was not known for a long time whether a better approximation than $\frac{1}{2}$ is possible for the general problem. Even for a special case of this problem, the Submodular Welfare problem that we discuss later, only a $\frac{1}{2}$-approximation was known in general (this result was rediscovered in the context of combinatorial auctions [38]). In certain special cases, e.g. when the objective function is of coverage type and the constraint is given by a partition matroid (Maximum Coverage with Group Budget Constraints [5]), a $(1 - 1/e)$-approximation has been developed [2]. It has been an open question whether $(1 - 1/e)$-approximation might be possible in general.

Our result. *There is a* $(1-1/e)$*-approximation for the problem* $\max\{f(S) : S \in \mathcal{I}\}$, *where* $f(S)$ *is any monotone submodular function given by a value oracle, and* $\mathcal{M} = (X, \mathcal{I})$ *is an arbitrary matroid given by a membership oracle.*

An intermediate step towards this goal was our work with Calinescu, Chekuri and Pál [4] where the same result is proved for a subclass of submodular functions that we call *weighted rank sums.* We define the class of weighted rank sums here: Let $(X, \mathcal{X})$ be a matroid and $\{w_j\}_{j \in X}$ a collection of weights. We define a *weighted rank function*,

$$r_w(S) = \max\{\sum_{j \in I} w_j : I \subseteq S \ \& \ I \in \mathcal{X}\}.$$

[1]The value of $1 - 1/e$ is roughly 0.632, i.e. larger than 1/2. This constant will appear frequently in this thesis.

This is known to be a monotone submodular function. Our class of *weighted rank sums* is the class of functions representable as $f(S) = \sum_i g_i(S)$ where each g_i is a weighted rank function. To be more precise, our algorithm requires that the number of weighted rank functions in the sum is polynomially bounded, and we are given a value oracle for each $g_i(S)$.

It is easy to see that the class of weighted rank sums is closed under taking positive linear combinations, i.e. it forms a cone. It is strictly contained in the cone of all monotone submodular functions, i.e. it is known that there exist monotone submodular functions that cannot be written as a weighted rank sum. Yet, the class of weighted rank sums seems to be a rather large subclass of submodular functions, capturing some interesting special cases such as coverage-type functions.

The structure of a weighted rank function allows us to write a linear program which can be solved in polynomial time. The fractional solution can then be rounded using the *pipage rounding* method, introduced by Ageev and Sviridenko [2]. This method, originally developed in a slightly different context, turns out to be very natural for rounding fractional solutions inside the matroid polytope (which was observed by Gruia Calinescu, Chandra Chekuri and Martin Pál [4]). However, for a general submodular function, it is unclear how to write a helpful linear program. We have explored two possibilities that we discuss later, but both linear programs turn out to be NP-hard to solve.

The algorithm which works for all monotone submodular functions is a "smooth greedy search" method which finds a good fractional solution without solving a linear program. It proceeds by building up a fractional solution by small modifications, based on marginal values of items with respect to this fractional solution. The last part of the algorithm uses again pipage rounding in order to convert the fractional solution into an integral one.

We present these results in Chapter 3.

1.2.3 The Submodular Welfare Problem

The following problem has appeared in the context of combinatorial auctions.

Problem: *Given n players with utility functions $w_i : 2^X \rightarrow \mathbb{R}_+$ which are assumed to be monotone and submodular, find a partition $X = S_1 \cup S_2 \cup \ldots \cup S_n$ in order to maximize $\sum_{i=1}^{n} w_i(S_i)$.*

This is an example of a combinatorial allocation problem, where m items are to be allocated to n players with different interests in different combinations of items, in order to maximize their total happiness. As we discussed before, we have to clarify the issue of accessing the players' utility functions. Unless the utility functions have a special form (such as coverage-type) which allows us to encode them efficiently, we have to rely on oracle access. We can consider either the *value oracle*, or the more powerful *demand oracle* (defined in Section 1.1). Whether we want to allow the use of a demand oracle depends on a particular setting, or on our point of view. From an economic standpoint, it seems natural to assume that given an assignment of prices, a player can decide which set of items is the most valuable for her. On the other hand, from a computational point of view, this decision problem is NP-hard for some very natural submodular utility functions (e.g. coverage-type functions). Thus we can either assume that players have sufficient knowledge of their utility functions (or the utility functions are simple enough) so that they are able to answer demand queries, or we can restrict ourselves to the value oracle model. In either case, the problem is non-trivial - it is NP-hard to $(1-\epsilon)$-approximate the Submodular Welfare Problem for some fixed $\epsilon > 0$, even in the demand oracle model [19].

We remark that in the context of combinatorial auctions, the utility functions are actually unknown and players are not assumed to be necessarily willing to reveal their true valuations. This leads to the design of *incentive-compatible mechanisms*, where players are not only queried about their valuations but also motivated to answer truthfully. We do not deal with this issue here and assume instead that players are willing to cooperate and give true answers to our queries.

Previous work. It is a result of folklore that without any assumptions on the utility functions, the problem is at least as hard as Set Packing (which corresponds to "single-minded bidders" who are interested in exactly one set each). Hence, no reasonable approximation can be expected in general. Research has focused on classes of utility function that allow better positive results, in particular submodular utility functions. Lehmann, Lehmann and Nisan [38] provide an approximation ratio of $\frac{1}{2}$ for the Submodular Welfare Problem, using a simple greedy algorithm using only value queries. A randomized version of this algorithm is shown in [14] to give a somewhat improved approximation ratio of $\frac{n}{2n-1}$. It is shown in [36] that if only value queries are allowed, then it is NP-hard to approximate Submodular Welfare within a ratio strictly better than $1 - 1/e$. It was unknown whether a $(1 - 1/e)$-approximation can be actually found with value queries.

Several earlier works [13, 14, 18] considered the following linear programming relaxation of the problem, usually referred to as the *Configuration LP*. Here, $x_{i,S}$ is intended to be an indicator variable that specifies whether player

i gets set S.

Configuration LP: Maximize $\sum_{i,S} x_{i,S} w_i(S)$ subject to:

- Item constraints: $\sum_{i,S|j \in S} x_{i,S} \leq 1$ for every item j.
- Player constraints: $\sum_S x_{i,S} \leq 1$ for every player i.
- Nonnegativity constraints: $x_{i,S} \geq 0$.

This linear program has an exponential number of variables but only a polynomial number of constraints. Using the fact that the separation oracle for the dual is exactly the demand oracle, this LP can be solved optimally in the demand oracle model. We refer the reader to [44, 13] for more details.

The integrality gap of this LP is known (up to low order terms) for classes of utility functions that are more general than submodular. For subadditive utility functions [2], it is $\frac{1}{2}$ [18], and for fractionally subadditive utility functions [3] it is $1-1/e$ [14]. A $\frac{1}{2}$-approximation algorithm for subadditive utilities and a $(1-1/e)$-approximation algorithm for fractionally subadditive functions were given in [18, 14]. For submodular utility functions, which are strictly contained in the classes we just mentioned, the positive results still apply and hence a $(1-1/e)$-approximation is possible in the demand oracle model. Still, it was not known whether this approximation factor can be improved for submodular utilities or whether the Configuration LP actually has an integrality gap of $1-1/e$ in this case. An example of integrality gap $\frac{7}{8}$ was given in [14].

It is proved in [19] that even in the demand oracle problem, the problem is NP-hard to approximate within $(1-\epsilon)$ for some fixed $\epsilon > 0$. This can be shown in several ways, using utility functions sufficiently simple so that players can answer demand queries efficiently. In one possible reduction, each player is interested only in a constant number of items, hence any "reasonable" kind of query can be answered in constant time, and yet the problem is non-trivial. This shows that the difficulty of the problem can lie in the coordination of the wishes of different players, rather than a complicated structure of the individual utility functions, and this difficulty cannot be circumvented by any reasonable oracle model.

[2] f is subadditive if $f(A \cup B) \leq f(A) + f(B)$ for any A, B.

[3] f is fractionally subadditive if $f(S) \leq \sum_i \alpha_i f(A_i)$ for any positive linear combination such that $\sum_{i:j \in A_i} \alpha_i \geq 1$ for each $j \in S$.

Connection to maximization under a matroid constraint. The Submodular Welfare Problem is in fact a special case of submodular maximization under a matroid constraint. The reduction is very natural and we briefly describe it here.

Let the set of players be P, the set of items Q, and for each $i \in P$, let the respective utility function be $w_i : 2^Q \to \mathbb{R}_+$. We define a new ground set $X = P \times Q$, with a function $f : 2^X \to \mathbb{R}_+$ defined as follows: Every set $S \subseteq X$ can be written uniquely as $S = \bigcup_{i \in P}(\{i\} \times S_i)$. Then let

$$f(S) = \sum_{i \in P} w_i(S_i).$$

The interpretation of $P \times Q$ is that we make $|P|$ copies of each item, one for each player. The function $f(S)$ represents the total utility of all players after allocating the items in S to the respective players (assuming that we actually have multiple copies of each item for different players).

However, in reality we can only allocate one copy of each item. Therefore, let us define a partition matroid $\mathcal{M} = (X, \mathcal{I})$ as follows:

$$\mathcal{I} = \{S \subseteq X \mid \forall j; |S \cap (P \times \{j\})| \leq 1\}.$$

Then, it is easy to see that the Submodular Welfare Problem is equivalent to the problem $\max_{S \in \mathcal{I}} f(S)$.

This has immediate corollaries (recall Section 1.2.2). For example, the $\frac{1}{2}$-approximation developed by Lehman, Lehman and Nisan [38] can be actually seen as a special case of the $\frac{1}{2}$-approximation by Nemhauser, Wolsey and Fisher [46]. (It is also mentioned in [46] that the greedy algorithm in case of a partition matroid can be implemented by processing the parts in an arbitrary order, and for each part, choosing the best element greedily. This corresponds to the on-line implementation of the greedy algorithm in [38].)

The question of improvement over 1/2 seems to be intimately related to the same question for submodular maximization subject to a matroid constraint. In fact, it was the special case of Submodular Welfare which led us to the idea of "smooth greedy search" that we mentioned in Section 1.2.2.

Our results.

The Submodular Welfare Problem can be $(1 - 1/e)$*-approximated in the value oracle model.*

This is a corollary of our work on matroid constraints, since Submodular Welfare is a special case of this framework (see above). Our algorithm is randomized and succeeds with high probability. It is simpler than the general algorithm for submodular maximization subject to a matroid constraint, because the relevant matroid here is just a partition matroid and hence we can replace the pipage rounding technique by straightforward randomized rounding.

Due to [36], $(1-1/e)$-approximation is optimal in the value oracle model unless $P = NP$. We provide an even stronger argument separating the approximation results achievable in the two oracle models.

For any fixed $\epsilon > 0$, a $(1-1/e+\epsilon)$-approximation for Submodular Welfare in the value oracle model would require an exponential number of value queries, regardless of our computational power.

This result (see Section 4.4) is very similar in spirit to our value query complexity result for submodular maximization that we mentioned at the end of Section 1.2.1, and also to the query complexity result for a cardinality constraint proved by Nemhauser and Wolsey [47]. Let us discuss our construction briefly here. It involves two instances whose optima differ by a factor arbitrarily close to $1-1/e$, but the two instances cannot be distinguished by a subexponential number of value queries. In addition, the result holds even when all players have the same utility function. We also show that in this case, we can achieve a $(1-1/e)$-approximation easily: by allocating each item independently to a random player. Thus, value queries are entirely fruitless when all players have the same the utility function. We present these results in Chapter 4.

On the other hand, we show the following in the demand oracle model.

There is some absolute constant $\epsilon > 0$ such that the Submodular Welfare Problem can be $(1-1/e+\epsilon)$-approximated in the demand oracle model.

Our algorithm is randomized and the approximation guarantee is in expectation. The value of ϵ that we obtain is small, roughly 10^{-5}. The significance of this result is that $1-1/e$ is not the optimal answer, and hence it is likely that further improvements are possible. Our algorithm uses the same Configuration LP that was used before; only our rounding procedure is more efficient. (In fact, it is significantly more complicated than the one needed to obtain $1-1/e$.) Our rounding is oblivious in the sense that its only input is a fractional LP solution, which is rounded randomly, without knowing anything about the actual utility functions of the players. We present this result in Chapter 5. It also appeared in [19], a joint work with Uriel Feige.

Another way to look at this result is that the integrality gap of the Configuration LP with submodular functions cannot be arbitrarily close to $1-1/e \simeq 0.632$. We do not determine the worst case integrality gap; we only improve the example of $7/8$ from [14] to an example with a gap of roughly 0.782. It remains an interesting question to investigate, what is the worst case integrality gap of this LP and whether this LP can achieve an optimal approximation result for this problem. The range of possible answers for the possible approximation factor is still between $(1-1/e+\epsilon, 1-\epsilon')$ for some very small $\epsilon, \epsilon' > 0$.

1.2.4 The Generalized Assignment Problem

Problem: *An instance of the Generalized Assignment Problem (GAP) consists of n bins and m items. Each item j has two nonnegative numbers for each bin i; a value v_{ij} and a size s_{ij}. We seek an assignment of items to bins such that the total size of items in each bin is at most 1, and the total value of all packed items is maximized.*

This problem has been considered in [52, 7] as a common framework for a number of scheduling/assignment problems. Note that GAP can be also seen as a combinatorial allocation problem (where bins correspond to players) with utility functions

$$w_i(S) = \max\{\sum_{j \in T} v_{ij} : T \subseteq S, \sum_{j \in T} s_{ij} \leq 1\}.$$

A $\frac{1}{2}$-approximation for this problem was announced in [7], by adapting an LP-based $\frac{1}{2}$-approximation for the minimization version of a similar packing problem [52]. In [22], this was improved to a $(1-1/e)$-approximation using an exponential large linear program very similar to the Configuration LP we discussed in Section 1.2.3. It is known that GAP is NP-hard to approximate within $1-\epsilon$ for some small $\epsilon > 0$ [7].

The linear program in [22] is shown as LP_1 in Fig 1.1. In LP_1, $\mathcal{F}_i$ denotes the collection of all feasible assignments for bin i, i.e. sets satisfying $\sum_{j \in S} s_{ij} \leq 1$. The variable $x_{i,S}$ represents bin i receiving a set of items S. Although this is an LP of exponential size, it is shown in [22] that it can be solved to an arbitrary precision in polynomial time. (The reason being that the separation oracle for the dual boils down to a knapsack problem, which can be solved almost precisely.) Then the fractional solution can be rounded to an integral one to obtain a $(1-1/e)$ approximation.

The authors in [22] also consider a more general problem, a "Separable Assignment Problem" (SAP). In SAP, each bin has an arbitrary down-monotone collection of "feasible sets" which can be legally packed in it. For this problem, a $(1-1/e)$-approximation is given in [22], assuming that we have an FPTAS for the single-bin packing subproblem (which is the knapsack problem in the case of GAP). Also, it is shown in [22] that the $1-1/e$ factor is optimal for SAP, even when the single-bin packing subproblem can be solved exactly. However, it was not known whether $1-1/e$ is the optimal approximation factor for GAP.

$$LP_1: \quad \max \sum_{j,S\in\mathcal{F}_i} y_{i,S} v_i(S); \qquad\qquad LP_2: \quad \max \sum_{i,S\in\mathcal{F}_i,j\in S} v_{ij} x_{i,j,S};$$

$$\forall j; \sum_{S\in\mathcal{F}_i} y_{i,S} \leq 1, \qquad\qquad \forall i,j,S; x_{i,j,S} \leq y_{i,S},$$

$$\forall i; \sum_{i,S\in\mathcal{F}_i: j\in S} y_{i,S} \leq 1, \qquad\qquad \forall j;\ \vec{x}_j \in P(\mathcal{X}_j),$$

$$\forall j,S;\ y_{i,S} \geq 0. \qquad\qquad \vec{y} \in P(\mathcal{M}).$$

Figure 1.1: Different LP relaxations for GAP.

Chandra Chekuri later observed that this $(1-1/e)$-approximation algorithm can be interpreted as a special case of submodular maximization subject to a matroid constraint [4]. A $(\frac{1}{2}-o(1))$-approximation can also be obtained using the greedy algorithm from [46]. (The greedy selection of an optimal element corresponds to a knapsack problem, which we can solve almost optimally.) The reduction is not as straightforward as in the case of Submodular Welfare, and in fact it blows up the size of the problem exponentially, so one has to be careful when using this reduction.

Reduction to submodular maximization under a matroid constraint. Consider a GAP instance with n bins and m items of values v_{ij}, and let $\mathcal{F}_i$ denote the collection of sets feasible for bin i. We define $X = \{(i,S) \mid 1 \leq i \leq n, S \in \mathcal{F}_i\}$ and a submodular function $f: 2^X \to \mathbb{R}^+$,

$$f(\mathcal{S}) = \sum_j \max_i \{v_{ij} : \exists (i,S) \in \mathcal{S}, j \in S\}.$$

We maximize this function subject to a matroid constraint $\mathcal{M}$, where $\mathcal{S} \in \mathcal{M}$ iff $\mathcal{S}$ contains at most one pair (i,S) for each i. Such a set $\mathcal{S}$ corresponds to an assignment of set S to bin i for each $(i,S) \in \mathcal{S}$. This is equivalent to GAP: although the bins can be assigned overlapping sets in this formulation, we only count the value of the most valuable assignment for each item. We can write $f(\mathcal{S}) = \sum_j g_j(\mathcal{S})$ where

$$g_j(\mathcal{S}) = \max_i \{v_{ij} : \exists (i,S) \in \mathcal{S}, j \in S\}$$

is a weighted rank function of a matroid $\mathcal{X}_j$ on X. In the matroid $\mathcal{X}_j$, an element $(i,S) \in X$ has weight v_{ij} if $j \in S$ and 0 otherwise. A set is independent in $\mathcal{X}_j$ iff its cardinality is at most 1. Therefore the problem falls under the umbrella of the framework we described in Section 1.2.2.

Let us write explicitly the LP arising from interpreting GAP as a submodular maximization problem. We have variables $y_{i,S}$ for each i and $S \in \mathcal{F}_i$. In addition, for each matroid $\mathcal{X}_j$, we define copies of these variables $x_{i,j,S}$. The resulting linear program is given as LP_2 in Fig 1.1. LP_2 has exponentially many variables and exponentially many constraints. However, observe that a feasible solution $y_{i,S}$ for LP_1 is also feasible for LP_2, when we set $x_{i,j,S} = y_{i,S}$ for $j \in S$ and 0 otherwise. This is because the constraint $\sum_{i,S:j \in S} y_{i,S} \leq 1$ in LP_1 implies $x_j \in P(\mathcal{X}_j)$, and the constraint $\sum_S y_{i,S} \leq 1$ implies $\vec{y} \in P(\mathcal{M})$. Therefore, we can solve LP_1 using the techniques of [22] and then convert the result into a feasible solution of LP_2. Finally, we can apply the pipage rounding technique to obtain a $(1 - 1/e)$-approximation.

This is simply a reformulation of the algorithm from [22]. However, the flexibility of the matroid constraint allows for more complicated conditions than each bin choosing at most one set. For example, one can handle a variant of GAP where we are allowed to use only k bins for some k given on the input. More generally, we can have a laminar matroid constraint on the bins that we are allowed to use. Again, it is possible to achieve a $(1 - 1/e)$-approximation for this problem. For more details, see [4].

Our results. We return to the original LP_1, as considered in [22]. Similarly to the case of the Configuration LP, it was not known whether the integrality gap of this LP is indeed $1 - 1/e$. We prove in Chapter 6 that this is not the case.

There is an absolute constant $\epsilon > 0$ such that any fractional solution of LP_1 can be rounded to an integral solution while losing at most a factor of $(1-1/e+\epsilon)$. Thus, the Generalized Assignment Problem can be $(1-1/e+\epsilon)$-approximated in polynomial time.

In this case, there is no issue of oracle models, since the item sizes and values can be specified explicitly on the input. Again, we use the technique of [22] to solve LP_1 (to an arbitrary precision). We achieve our improvement by applying a more sophisticated rounding technique to the fractional solution. The value of ϵ that we achieve here is even much smaller than in the case of Submodular Welfare, roughly on the order of 10^{-120}. It is reasonable to expect that it is possible to achieve an improvement more significant that this. The worst integrality gap that we found for LP_1 is $\frac{4}{5}$.

This result also appeared in [19], a joint work with Uriel Feige.

1.3 Concluding remarks

In this thesis, we present a number of approximation and hardness results on problems involving submodular functions. It seems that the factor of $1 - 1/e$ plays a very special role here, reappearing in almost every problem that we study. In the context of submodular maximization, $1 - 1/e$ is often seen as a natural threshold beyond which any improvement is impossible, and this intuition harks back to the remarkable inapproximability results of Feige [17]. We show that although this intuition often proves to be true, in two particular cases of interest (Submodular Welfare in the demand oracle model and the Generalized Assignment problems), the threshold of $1 - 1/e$ can be in fact exceeded. It is intriguing that we were able to achieve only such a miniscule improvement over $1 - 1/e$, while on the other hand the known inapproximability results are very close to 1. We have no conjecture about what the optimal approximation factors for these two problems might be.

Chapter 2

Nonnegative Submodular Maximization

Problem: *Given a submodular function $f : 2^X \to \mathbb{R}_+$ with value-oracle access, maximize $f(S)$ over all subsets $S \subseteq X$.*

We develop a number of approximation algorithms for this problem. First we consider nonadaptive algorithms, as defined in Section 1.1.

2.1 Nonadaptive algorithms

Recall that Max Cut and Max Di-Cut are special cases of submodular maximization. It is known that simply choosing a random cut is a good choice for Max Cut, achieving an approximation factor of 1/2. Prior to the semidefinite programming approach of Goemans and Williamson, this was the best known approximation algorithm for Max Cut. Similarly, a random cut achieves a $\frac{1}{4}$-approximation for Max Di-Cut. We show that submodularity is the underlying reason for these factors, by presenting the same results in the case of general submodular functions.

2.1.1 A random set

The following "algorithm" does not use any queries at all. It simply returns a random set. We show that this is already a reasonable approximation algorithm for any nonnegative submodular function.

The Random Set Algorithm: RS.

- Return $R = X(1/2)$, a random subset of X where each element appears independently with probability 1/2.

Theorem 2.1. *Let $f : 2^X \to \mathbb{R}_+$ be a submodular function, $OPT = \max_{S \subseteq X} f(S)$ and let R denote a uniformly random subset $R = X(1/2)$. Then $\mathbf{E}[f(R)] \geq \frac{1}{4}OPT$. In addition, if f is symmetric ($f(S) = f(X \setminus S)$ for every $S \subseteq X$), then $\mathbf{E}[f(R)] \geq \frac{1}{2}OPT$.*

Before proving this result, we show a useful probabilistic property of submodular functions (extending the considerations of [18, 19]). This property will be essential in the analysis of our more involved randomized algorithms as well.

Lemma 2.2. *Let $g : 2^X \to \mathbb{R}$ be submodular. Denote by $A(p)$ a random subset of A where each element appears with probability p. Then*

$$\mathbf{E}[g(A(p))] \geq (1-p)\ g(\emptyset) + p\ g(A).$$

Proof. By induction on the size of A: For $A = \emptyset$, the lemma is trivial. So assume $A = A' \cup \{x\}, x \notin A'$. We can also write $A'(p) = A(p) \cap A'$; then

$$\begin{aligned} \mathbf{E}[g(A(p))] &= \mathbf{E}[g(A'(p))] + \mathbf{E}[g(A(p)) - g(A(p) \cap A')] \\ &\geq \mathbf{E}[g(A'(p))] + \mathbf{E}[g(A' \cup A(p)) - g(A')] \end{aligned}$$

using submodularity on A' and $A(p)$. The set $A' \cup A(p)$ is either equal to A (when $x \in A(p)$, which happens with probability p) or otherwise it's equal to A'. Therefore we get

$$\mathbf{E}[g(A(p))] \geq \mathbf{E}[g(A'(p)] + p(g(A) - g(A'))$$

and using the inductive hypothesis, $\mathbf{E}[g(A'(p))] \geq (1-p)\ g(\emptyset) + p\ g(A')$, we get the statement of the lemma. □

By a double application of Lemma 2.2, we obtain the following.

Lemma 2.3. *Let $f : 2^X \to \mathbb{R}$ be submodular, $A, B \subseteq X$ two (not necessarily disjoint) sets and $A(p), B(q)$ their independently sampled subsets, where each element of A appears in $A(p)$ with probability p and each element of B appears in $B(q)$ with probability q. Then*

$$\mathbf{E}[f(A(p) \cup B(q))] \geq (1-p)(1-q)\ f(\emptyset) + p(1-q)\ f(A) + (1-p)q\ f(B) + pq\ f(A \cup B).$$

Proof. Condition on $A(p) = A'$ and define $g(T) = f(A' \cup T)$. This is a submodular function as well and Lemma 2.2 implies $\mathbf{E}[g(B(q))] \geq (1-q)\ f(A') + q\ f(A' \cup B)$. Also, $\mathbf{E}[g(B(q))] = \mathbf{E}[f(A(p) \cup B(q)) \mid A(p) = A']$, and by unconditioning: $\mathbf{E}[f(A(p) \cup B(q))] \geq \mathbf{E}[(1-q)\ f(A(p)) + q\ f(A(p) \cup B)]$. Finally, we apply Lemma 2.2 once again: $\mathbf{E}[f(A(p))] \geq (1-p)\ f(\emptyset) + p\ f(A)$, and by applying the same to the submodular function $h(S) = f(S \cup B)$, $\mathbf{E}[f(A(p) \cup B)] \geq (1-p)\ f(B) + p\ f(A \cup B)$. This implies the claim. □

This lemma gives immediately the performance of Algorithm RS.

Proof. Denote the optimal set by S and its complement by $\bar{S}$. We can write $R = S(1/2) \cup \bar{S}(1/2)$. Using Lemma 2.3, we get

$$\mathbf{E}[f(R)] \geq \frac{1}{4}f(\emptyset) + \frac{1}{4}f(S) + \frac{1}{4}f(\bar{S}) + \frac{1}{4}f(X).$$

Every term is nonnegative and $f(S) = OPT$, so we get $\mathbf{E}[f(R)] \geq \frac{1}{4}OPT$. In addition, if f is symmetric, we also have $f(\bar{S}) = OPT$ and then $\mathbf{E}[f(R)] \geq \frac{1}{2}OPT$. $\square$

A set of value very close to the expectation can be found, if we sample independently a polynomial number of random sets and return the maximum.

We will see that for symmetric functions, the factor of 1/2 cannot be improved by any (even adaptive) algorithm, using a polynomial number of value queries. (See Section 2.3.) So the only remaining gap is in the general case, where a random set gives only $\frac{1}{4}$-approximation. We will also show that the factor of 1/4 is optimal for nonadaptive algorithms, assuming that the algorithm returns one of the queried sets. However, it is possible to design a $\frac{1}{3}$-approximation algorithm which queries a polynomial number of sets nonadaptively and then returns a possibly different set after a polynomial-time computation. We present this algorithm next.

2.1.2 A nonadaptive randomized algorithm

The intuition behind the algorithm comes from the Max Di-Cut problem: When does a random cut achieve only 1/4 of the optimum? This is if and only if the optimum contains all the directed edges of the graph, i.e. the vertices can be partitioned into $V = A \cup B$ so that all edges of the graph are directed from A to B. However, in this case it's easy to find the optimal solution, by a local test on the in-degree and out-degree of each vertex. In the language of submodular function maximization, this means that elements can be easily partitioned into those whose inclusion in S always increases the value of $f(S)$, and those which always decrease $f(S)$. Our generalization of this local test is the following.

Definition 2.4. *Let $R = X(1/2)$ denote a uniformly random subset of X. For each element x, define*

$$\omega(x) = \mathbf{E}[f(R \cup \{x\}) - f(R \setminus \{x\})].$$

Note that these values can be estimated by random sampling, up to an error polynomially small relative to $\max_{R,x} |f(R \cup \{x\}) - f(R \setminus \{x\})| \leq OPT$. This is sufficient for our purposes; in the following, we assume that we have estimates $\tilde{\omega}(x)$ such that $|\omega(x) - \tilde{\omega}(x)| \leq OPT/n^2$.

The Non-Adaptive Algorithm: NA.

- Use random sampling to find $\tilde{\omega}(x)$ for each $x \in X$.
- Independently, sample a random set $R = X(1/2)$.
- With prob. 8/9, return R.
- With prob. 1/9, return

$$A = \{x \in X : \tilde{\omega}(x) > 0\}.$$

Theorem 2.5. *For any nonnegative submodular function, Algorithm NA achieves expected value at least* $(1/3 - o(1))\ OPT$.

Proof. Let $A = \{x \in X : \tilde{\omega}(x) > 0\}$ and $B = X \setminus A = \{x \in X : \tilde{\omega}(x) \leq 0\}$. Therefore we have $\omega(x) \geq -OPT/n^2$ for any $x \in A$ and $\omega(x) \leq OPT/n^2$ for any $x \in B$. We shall keep in mind that (A, B) is a partition of all the elements, and so we have $(A \cap T) \cup (B \cap T) = T$ for any set T, etc.

Denote by C the optimal set, $f(C) = OPT$. Let $f(A) = \alpha$, $f(B \cap C) = \beta$ and $f(B \cup C) = \gamma$. By submodularity, we have

$$\alpha + \beta = f(A) + f(B \cap C) \geq f(\emptyset) + f(A \cup (B \cap C)) \geq f(A \cup C)$$

and

$$\alpha + \beta + \gamma \geq f(A \cup C) + f(B \cup C) \geq f(X) + f(C) \geq OPT.$$

Therefore, either α, the value of A, is at least $OPT/3$, or else one of β and γ is at least $OPT/3$; we prove that then $\mathbf{E}[f(R)] \geq OPT/3$ as well.

Let's start with $\beta = f(B \cap C)$. Instead of $\mathbf{E}[f(R)]$, we show that it's enough to estimate $\mathbf{E}[f(R \cup (B \cap C))]$. Recall that for any $x \in B$, we have $\omega(x) = \mathbf{E}[f(R \cup \{x\}) - f(R \setminus \{x\})] \leq OPT/n^2$. Consequently, we also have $\mathbf{E}[f(R \cup \{x\}) - f(R)] = \frac{1}{2}\omega(x) \leq OPT/(2n^2)$. Let's order the elements of $B \cap C = \{b_1, \ldots, b_\ell\}$ and write

$$f(R \cup (B \cap C)) = f(R) + \sum_{j=1}^{\ell} (f(R \cup \{b_1, \ldots, b_j\}) - f(R \cup \{b_1, \ldots, b_{j-1}\})).$$

By the property of decreasing marginal values, we get

$$f(R \cup (B \cap C)) \leq f(R) + \sum_{j=1}^{\ell} (f(R \cup \{b_j\}) - f(R)) = f(R) + \sum_{x \in B \cap C} (f(R \cup \{x\}) - f(R))$$

and therefore

$$\begin{aligned} \mathbf{E}[f(R \cup (B \cap C))] &\leq \mathbf{E}[f(R)] + \sum_{x \in B \cap C} \mathbf{E}[f(R \cup \{x\}) - f(R)] \\ &\leq \mathbf{E}[f(R)] + |B \cap C| \frac{OPT}{2n^2} \leq \mathbf{E}[f(R)] + \frac{OPT}{2n}. \end{aligned}$$

So it's enough to lower-bound $\mathbf{E}[f(R \cup (B \cap C))]$. We do this by defining a new submodular function, $g(R) = f(R \cup (B \cap C))$, and applying Lemma 2.3 to $\mathbf{E}[g(R)] = \mathbf{E}[g(C(1/2) \cup \bar{C}(1/2))]$. The lemma implies that

$$\begin{aligned} \mathbf{E}[f(R \cup (B \cap C))] &\geq \frac{1}{4} g(\emptyset) + \frac{1}{4} g(C) + \frac{1}{4} g(\bar{C}) + \frac{1}{4} g(X) \\ &\geq \frac{1}{4} g(\emptyset) + \frac{1}{4} g(C) = \frac{1}{4} f(B \cap C) + \frac{1}{4} f(C) \\ &= \frac{\beta}{4} + \frac{OPT}{4}. \end{aligned}$$

Note that $\beta \geq OPT/3$ implies $\mathbf{E}[f(R \cup (B \cap C))] \geq OPT/3$. Symmetrically, we show a similar analysis for $\mathbf{E}[f(R \cap (B \cup C))]$. Now we use the fact that for any $x \in A$, $\mathbf{E}[f(R) - f(R \setminus \{x\})] = \frac{1}{2}\omega(x) \geq -OPT/(2n^2)$. Let $A \setminus C = \{a_1, a_2, \ldots, a_k\}$ and write

$$\begin{aligned} f(R) &= f(R \setminus (A \setminus C)) + \sum_{j=1}^{k} (f(R \setminus \{a_{j+1}, \ldots, a_k\}) - f(R \setminus \{a_j, \ldots, a_k\})) \\ &\geq f(R \setminus (A \setminus C)) + \sum_{j=1}^{k} (f(R) - f(R \setminus \{a_j\}) \end{aligned}$$

using the condition of decreasing marginal values. Note that $R \setminus (A \setminus C) = R \cap (B \cup C)$. By taking the expectation,

$$\begin{aligned} \mathbf{E}[f(R)] &\geq \mathbf{E}[f(R \cap (B \cup C))] + \sum_{j=1}^{k} \mathbf{E}[f(R) - f(R \setminus \{a_j\})] \\ &= \mathbf{E}[f(R \cap (B \cup C))] - |A \setminus C| \frac{OPT}{2n^2} \geq \mathbf{E}[f(R \cap (B \cup C))] - \frac{OPT}{2n}. \end{aligned}$$

Again, we estimate $\mathbf{E}[f(R \cap (B \cup C))] = \mathbf{E}[f(C(1/2) \cup (B \setminus C)(1/2))]$ using Lemma 2.3. We get

$$\begin{aligned}\mathbf{E}[f(R \cap (B \cup C))] &\geq \frac{1}{4}f(\emptyset) + \frac{1}{4}f(C) + \frac{1}{4}f(B \setminus C) + \frac{1}{4}f(B \cup C)\\ &\geq \frac{OPT}{4} + \frac{\gamma}{4}.\end{aligned}$$

Now we combine our estimates for $\mathbf{E}[f(R)]$:

$$\mathbf{E}[f(R)] + \frac{OPT}{2n} \geq \frac{1}{2}\mathbf{E}[f(R \cup (B \cap C))] + \frac{1}{2}\mathbf{E}[f(R \cap (B \cup C))] \geq \frac{OPT}{4} + \frac{\beta}{8} + \frac{\gamma}{8}.$$

Finally, the expected value obtained by the algorithm is

$$\frac{8}{9}\mathbf{E}[f(R)] + \frac{1}{9}f(A) \geq \frac{2}{9}OPT - \frac{4}{9n}OPT + \frac{\beta}{9} + \frac{\gamma}{9} + \frac{\alpha}{9} \geq \left(\frac{1}{3} - \frac{4}{9n}\right)OPT$$

since $\alpha + \beta + \gamma \geq OPT$. □

2.2 Adaptive algorithms

2.2.1 A deterministic local search algorithm

Our deterministic algorithm is based on a simple local search technique. We try to increase the value of our solution S by either including a new element in S or discarding one of the elements of S. We call S a *local optimum* if no such operation increases the value of S. Local optima have the following property which was first observed in [8, 28].

Lemma 2.6. *Given a submodular function f, if S is a local optimum of f, and I and J are two sets such that $I \subseteq S \subseteq J$, then $f(I) \leq f(S)$ and $f(J) \leq f(S)$.*

This property turns out to be very useful in proving that a local optimum is a good approximation to the global optimum. However, it is known that finding a local optimum for the Max Cut problem is PLS-complete [50]. Therefore, we relax our local search and find an *approximate local optimal solution.*

Local Search Algorithm: LS.

1. Let $S := \{v\}$ where $f(\{v\})$ is the maximum over all singletons $v \in X$.
2. If there exists an element $a \in X \setminus S$ such that $f(S \cup \{a\}) > (1 + \frac{\epsilon}{n^2})f(S)$, then let $S := S \cup \{a\}$, and go back to Step 2.

3. If there exists an element $a \in S$ such that $f(S\backslash\{a\}) > (1+\frac{\epsilon}{n^2})f(S)$, then let $S := S\backslash\{a\}$, and go back to Step 2.

4. Return the maximum of $f(S)$ and $f(X\backslash S)$.

It is easy to see that if the algorithm terminates, the set S is a $(1+\frac{\epsilon}{n^2})$-approximate local optimum, in the following sense.

Definition 2.7. *Given $f : 2^X \to \mathbb{R}$, a set S is called a* $(1+\alpha)$-approximate local optimum, *if $(1+\alpha)f(S) \geq f(S \setminus \{v\})$ for any $v \in S$, and $(1+\alpha)f(S) \geq f(S \cup \{v\})$ for any $v \notin S$.*

We prove the following analogue of Lemma 2.6.

Lemma 2.8. *If S is an $(1+\alpha)$-approximate local optimum for a submodular function f, then for any subsets such that $I \subseteq S \subseteq J$, $f(I) \leq (1+n\alpha)f(S)$ and $f(J) \leq (1+n\alpha)f(S)$.*

Proof. Let $I = T_1 \subseteq T_2 \subseteq \ldots \subseteq T_k = S$ be a chain of sets where $T_i\backslash T_{i-1} = \{a_i\}$. For each $2 \leq i \leq k$, we know that $f(T_i) - f(T_{i-1}) \geq f(S) - f(S \setminus \{a_i\}) \geq -\alpha f(S)$ using the submodularity and approximate local optimality of S. Summing up these inequalities, we get $f(S) - f(I) \geq -k\alpha f(S)$. Thus $f(I) \leq (1+k\alpha)f(S) \leq (1+n\alpha)f(S)$. This completes the proof for set I. The proof for set J is very similar. □

Theorem 2.9. *Algorithm LS is a $\left(\frac{1}{3} - \frac{\epsilon}{n}\right)$-approximation algorithm for maximizing nonnegative submodular functions, and a $\left(\frac{1}{2} - \frac{\epsilon}{n}\right)$-approximation algorithm for maximizing nonnegative symmetric submodular functions. The algorithm uses at most $O(\frac{1}{\epsilon}n^3 \log n)$ oracle calls.*

Proof. Consider an optimal solution C and let $\alpha = \frac{\epsilon}{n^2}$. If the algorithm terminates, the set S obtained at the end is a $(1+\alpha)$-approximate local optimum. By Lemma 2.8, $f(S \cap C) \leq (1+n\alpha)f(S)$ and $f(S \cup C) \leq (1+n\alpha)f(S)$. Using submodularity, $f(S \cup C) + f(X\backslash S) \geq f(C\backslash S) + f(X) \geq f(C\backslash S)$, and $f(S \cap C) + f(C\backslash S) \geq f(C) + f(\emptyset) \geq f(C)$. Putting these inequalities together, we get

$$2(1+n\alpha)f(S)+f(X\backslash S) \geq f(S\cap C)+f(S\cup C)+f(X\backslash S) \geq f(S\cap C)+f(C\backslash S) \geq f(C).$$

For $\alpha = \frac{\epsilon}{n^2}$, this implies that either $f(S) \geq (\frac{1}{3} - o(1))OPT$ or $f(X\backslash S) \geq (\frac{1}{3} - o(1))OPT$.

For symmetric submodular functions, we get

$$2(1+n\alpha)f(S) \geq f(S \cap C) + f(S \cup \bar{C}) = f(S \cap C) + f(\bar{S} \cap C) \geq f(C)$$

and hence $f(S)$ is a $(\frac{1}{2} - o(1))$-approximation.

To bound the running time of the algorithm, let v be the element with the maximum value $f(\{v\})$ over all elements of X. It is simple to see that $OPT \leq nf(\{v\})$. Since after each iteration, the value of the function increases by a factor of at least $(1 + \frac{\epsilon}{n^2})$, if the number of iterations of the algorithm is k, then $(1 + \frac{\epsilon}{n^2})^k \leq n$. Therefore, $k = O(\frac{1}{\epsilon} n^2 \log n)$ and the number of queries is $O(\frac{1}{\epsilon} n^3 \log n)$. □

2.2.2 A smooth local search algorithm

Next, we present a randomized algorithm which improves the approximation ratio of 1/3. The main idea behind this algorithm is to find a "smoothed" local optimum, where elements are sampled randomly but with different probabilities, based on some underlying set A. The general approach of local search, based on a function derived from the one we are interested in, has been referred to as "non-oblivious local search" in the literature [3].

Definition 2.10. *We say that a set is sampled with bias δ based on A, if elements in A are sampled independently with probability $p = (1 + \delta)/2$ and elements outside of A are sampled independently with probability $q = (1 - \delta)/2$. We denote this random set by $\mathcal{R}(A, \delta)$.*

The Smooth Local Search algorithm: SLS.

1. Choose $\delta \in [0, 1]$ and start with $A = \emptyset$. Let $n = |X|$ denote the total number of elements. In the following, use an estimate for OPT, for example from Algorithm LS.

2. For each element x, estimate $\omega_{A,\delta}(x) = \mathbf{E}[f(\mathcal{R}(A, \delta) \cup \{x\})] - \mathbf{E}[f(\mathcal{R}(A, \delta) \setminus \{x\})]$, within an error of $\frac{1}{n^2} OPT$. Call this estimate $\tilde{\omega}_{A,\delta}(x)$.

3. If there is an element $x \in X \setminus A$ such that $\tilde{\omega}_{A,\delta}(x) > \frac{2}{n^2} OPT$, include x in A and go to Step 2.

4. If there is $x \in A$ such that $\tilde{\omega}_{A,\delta}(x) < -\frac{2}{n^2} OPT$, remove x from A and go to Step 2.

5. Choose $\delta' \in [-1, 1]$ and return a random set from the distribution $\mathcal{R}(A, \delta')$.

In effect, we find an approximate local optimum of a derived function $\Phi(A) = \mathbf{E}[f(\mathcal{R}(A, \delta))]$. Then we return a set sampled according to $\mathcal{R}(A, \delta')$; possibly for $\delta' \neq \delta$. One can run Algorithm SLS with $\delta = \delta'$ and prove that the best approximation factor for such parameters is achieved by setting $\delta = \delta' = \frac{\sqrt{5}-1}{2}$, the golden ratio. Then, we get an approximation factor of $\frac{3-\sqrt{5}}{2} - o(1) \simeq 0.38$. Interestingly, one can improve this approximation factor to 0.4 by choosing two parameter pairs (δ, δ') and taking the maximum of the two solutions.

Theorem 2.11. *Algorithm SLS runs in polynomial time. If we run SLS for two choices of parameters, $(\delta = \frac{1}{3}, \delta' = \frac{1}{3})$ and $(\delta = \frac{1}{3}, \delta' = -1)$, the better of the two solutions has expected value at least $(\frac{2}{5} - o(1))OPT$.*

Proof. Let $\Phi(A) = \mathbf{E}[f(\mathcal{R}(A, \delta))]$. Recall that in $\mathcal{R}(A, \delta)$, elements from A are sampled with probability $p = (1 + \delta)/2$, while elements from B are sampled with probability $q = (1 - \delta)/2$. Consider Step 3 where an element x is added to A. Also, let $B' = X \setminus (A \cup \{x\})$. The reason why x is added to A is that $\tilde{\omega}_{A,\delta}(x) > \frac{2}{n^2}OPT$; i.e. $\omega_{A,\delta}(x) > \frac{1}{n^2}OPT$. During this step, $\Phi(A)$ increases by

$$\begin{aligned}
\Phi(A \cup \{x\}) - \Phi(A) &= \mathbf{E}[f((A \cup \{x\})(p) \cup B'(q)] - \mathbf{E}[f(A(p) \cup (B' \cup \{x\})(q))] \\
&= (p - q)\ \mathbf{E}[f(A(p) \cup B'(q) \cup \{x\}) - f(A(p) \cup B'(q))] \\
&= \delta\ \mathbf{E}[f(\mathcal{R}(A, \delta) \cup \{x\}) - f(\mathcal{R}(A, \delta) \setminus \{x\})] \\
&= \delta\ \omega_{A,\delta}(x) > \frac{\delta}{n^2}OPT.
\end{aligned}$$

Similarly, executing Step 4 increases $\Phi(A)$ by at least $\frac{\delta}{n^2}OPT$. Since the value of $\Phi(A)$ is always between 0 and OPT, the algorithm cannot iterate more than n^2/δ times and thus it runs in polynomial time. Also, note that finding a local maximum of $\Phi(A)$ is equivalent to finding a set A such that its elements $x \in A$ have $\omega_{A,\delta}(x) \geq 0$, while elements $x \notin A$ have $\omega_{A,\delta}(x) \leq 0$. Here, we find a set satisfying this up to a certain error.

From now on, let A be the set at the end of the algorithm and $B = X \setminus A$. We also use $R = A(p) \cup B(q)$ to denote a random set from the distribution $\mathcal{R}(A, \delta)$. We denote by C the optimal solution, while our algorithm returns either R (for $\delta' = \delta$) or B (for $\delta' = -1$). When the algorithm terminates, we have $\omega_{A,\delta}(x) \geq -\frac{3}{n^2}OPT$ for any $x \in A$, and $\omega_{A,\delta}(x) \leq \frac{3}{n^2}OPT$ for any $x \in B$. Consequently, for any $x \in B$ we have $\mathbf{E}[f(R \cup \{x\})] - f(R)] = \Pr[x \notin R]\mathbf{E}[f(R \cup \{x\}) - f(R \setminus \{x\})] = \frac{2}{3}\omega_{A,\delta}(x) \leq \frac{2}{n^2}OPT$, using $\Pr[x \notin R] = p = 2/3$. Let's order the elements of $B \cap C = \{b_1, \ldots, b_\ell\}$ and write

$$f(R \cup (B \cap C)) = f(R) + \sum_{j=1}^{\ell}(f(R \cup \{b_1, \ldots, b_j\}) - f(R \cup \{b_1, \ldots, b_{j-1}\})).$$

By the property of decreasing marginal values, we get $f(R \cup \{b_1, \ldots, b_j\}) - f(R \cup \{b_1, \ldots, b_{j-1}\}) \leq f(R \cup \{b_j\}) - f(R)$ and hence

$$\begin{aligned} \mathbf{E}[f(R \cup (B \cap C))] &\leq \mathbf{E}[f(R)] + \sum_{x \in B \cap C} \mathbf{E}[f(R \cup \{x\}) - f(R)] \\ &\leq \mathbf{E}[f(R)] + |B \cap C| \frac{2}{n^2} OPT \leq \mathbf{E}[f(R)] + \frac{2}{n} OPT. \end{aligned}$$

Similarly, we can obtain $\mathbf{E}[f(R \cap (B \cup C))] \leq \mathbf{E}[f(R)] + \frac{2}{n} OPT$. This means that instead of R, we can analyze $R \cup (B \cap C)$ and $R \cap (B \cup C)$. In order to estimate $\mathbf{E}[f(R \cup (B \cap C))]$ and $\mathbf{E}[f(R \cap (B \cup C))]$, we use a further extension of Lemma 2.3 which can be proved by another iteration of the same proof:

$$(*) \quad \mathbf{E}[f(A_1(p_1) \cup A_2(p_2) \cup A_3(p_3))] \geq \sum_{I \subseteq \{1,2,3\}} \prod_{i \in I} p_i \prod_{i \notin I} (1 - p_i) \; f\left(\bigcup_{i \in I} A_i \right).$$

First, we deal with $R \cap (B \cup C) = (A \cap C)(p) \cup (B \cap C)(q) \cup (B \setminus C)(q)$. We plug in $\delta = 1/3$, i.e. $p = 2/3$ and $q = 1/3$. Then (*) yields

$$\begin{aligned} \mathbf{E}[f(R \cap (B \cup C))] &\geq \frac{8}{27} f(A \cap C) + \frac{2}{27} f(B \cup C) \\ &+ \frac{2}{27} f(B \cap C) + \frac{4}{27} f(C) + \frac{4}{27} f(F) + \frac{1}{27} f(B) \end{aligned}$$

where we denote $F = (A \cap C) \cup (B \setminus C)$ and we discarded the terms $f(\emptyset) \geq 0$ and $f(B \setminus C) \geq 0$. Similarly, we estimate $\mathbf{E}[f(R \cup (B \cap C))]$, applying (*) to a submodular function $h(R) = f(R \cup (B \cap C))$ and writing $\mathbf{E}[f(R \cup (B \cap C))] = \mathbf{E}[h(R)] = \mathbf{E}[h((A \cap C)(p) \cup (A \setminus C)(p) \cup B(q))]$:

$$\begin{aligned} \mathbf{E}[f(R \cup (B \cap C))] &\geq \frac{8}{27} f(A \cup C) + \frac{2}{27} f(B \cup C) \\ &+ \frac{2}{27} f(B \cap C) + \frac{4}{27} f(C) + \frac{4}{27} f(\bar{F}) + \frac{1}{27} f(B). \end{aligned}$$

Here, $\bar{F} = (A \setminus C) \cup (B \cap C)$. We use $\mathbf{E}[f(R)] + \frac{2}{n} OPT \geq \frac{1}{2}(\mathbf{E}[f(R \cap (B \cup C))] + \mathbf{E}[f(R \cup (B \cap C))])$ and combine the two estimates.

$$\begin{aligned} \mathbf{E}[f(R)] + \frac{2}{n} OPT &\geq \frac{4}{27} f(A \cap C) + \frac{4}{27} f(A \cup C) + \frac{2}{27} f(B \cap C) + \frac{2}{27} f(B \cup C) \\ &+ \frac{4}{27} f(C) + \frac{2}{27} f(F) + \frac{2}{27} f(\bar{F}) + \frac{1}{27} f(B). \end{aligned}$$

Now we add $\frac{3}{27} f(B)$ on both sides and apply submodularity: $f(B) + f(F) \geq f(B \cup C) + f(B \setminus C) \geq f(B \cup C)$ and $f(B) + f(\bar{F}) \geq f(B \cup (A \setminus C)) + f(B \cap C) \geq$

$f(B \cap C)$. This leads to

$$\begin{aligned}\mathbf{E}[f(R)] + \frac{1}{9}f(B) + \frac{2}{n}OPT &\geq \frac{4}{27}f(A \cap C) + \frac{4}{27}f(A \cup C) \\ &+ \frac{4}{27}f(B \cap C) + \frac{4}{27}f(B \cup C) + \frac{4}{27}f(C)\end{aligned}$$

and once again using submodularity, $f(A \cap C) + f(B \cap C) \geq f(C)$ and $f(A \cup C) + f(B \cup C) \geq f(C)$, we get

$$\mathbf{E}[f(R)] + \frac{1}{9}f(B) + \frac{2}{n}OPT \geq \frac{12}{27}f(C) = \frac{4}{9}OPT.$$

To conclude, either $\mathbf{E}[f(R)]$ or $f(B)$ must be at least $(\frac{2}{5} - \frac{2}{n})OPT$, otherwise we get a contradiction. □

2.2.3 Discussion of possible improvements

The analysis of the SLS algorithm is tight, as can be seen from the following example.

Example 2.12. *Consider $X = \{a, b, c, d\}$ and a directed graph on these vertices, consisting of 3 edges (a, b), (b, c) and (c, d). Let's define a function $f : 2^X \to \mathbb{R}_+$ by defining $f(S)$ as the number of edges going from S to $X \setminus S$. This is a directed cut type function, which is known to be submodular.*

Observe that $A = \{a, b\}$ could be the set found by our algorithm. In fact, this is a local optimum of $\mathbf{E}[f(\mathcal{R}(A, \delta))]$ for any value of δ, since we have $\mathbf{E}[f(\mathcal{R}(\{a, b\}, \delta))] = 2pq + p^2 = 1 - q^2$ (where $p = (1 + \delta)/2$ and $q = (1 - \delta)/2$). It can be verified that the value of $\mathbf{E}[f(\mathcal{R}(A, \delta))]$ for any A obtained by switching one element is at most $1 - q^2$.

Algorithm SLS returns either $\mathcal{R}(A, 1/3)$ of expected value $1 - (1/3)^2$, with probability $9/10$, or B of value 0, with probability $1/10$. Thus the expected value returned by SLS is 0.8, while the optimum is $f(\{a, c\}) = 2$.

Of course, this example can be circumvented if we define SLS to take the maximum of $\mathbf{E}[f(\mathcal{R}(A, \delta))]$ and $f(B)$ rather than a weighted average. However, it's not clear how to use this in the analysis.

The SLS algorithm can be presented more generally as follows.

Algorithm SLS^***.**

1. Find an approximate local maximum of $\mathbf{E}[f(\mathcal{R}(A, \delta))]$, for some $\delta \in [0, 1]$.

2. Return a random set from the distribution $\mathcal{R}(A, \delta')$ for some $\delta' \in [-1, 1]$.

The algorithm SLS uses a fixed value of $\delta = 2/3$ and two possible values of $\delta' = 2/3$ or $\delta' = -1$. More generally, we can search for the best values of δ and δ' for a given instance. Then the algorithm SLS^* might have a chance to improve the approximation factor of 2/5. Considering Section 2.3.3, the best we can hope for would be a $\frac{1}{2}$-approximation in the general case. This might be possible with SLS^*; however, we show that it is not enough to fix a value of δ in advance, as we do in our algorithms above.

Example 2.13. *Similarly to Example 2.12, consider a directed graph on $X = \{a, b, c, d\}$, with 6 edges: (a, b) of weight p, (b, a) of weight q, (b, c) of weight 1, (c, b) of weight 1, (c, d) of weight p and (d, c) of weight q.*

It can be verified that $A = \{a, b\}$ is a local optimum for $\mathbf{E}[f(\mathcal{R}(A, \delta))]$ such that $p = (1 + \delta)/2$ and $q = (1 - \delta)/2$. Moreover, for any (possibly different) probabilities $p' = (1 + \delta')/2$ and $q' = (1 - \delta')/2$, we have

$$\begin{aligned}\mathbf{E}[f(\mathcal{R}(\{a, b\}, \delta'))] &= p'q'(w(a, b) + w(b, a) + w(c, d) + w(d, c)) \\ &\quad + p'^2 w(b, c) + q'^2 w(c, b) = 2p'q' + p'^2 + q'^2 = 1.\end{aligned}$$

Thus Algorithm SLS^* with a fixed choice of δ and a flexible choice of δ' returns a set of expected value 1. The optimum solution here is $f(\{a, c\}) = 1 + 2p$, so this yields an approximation factor $1/(1 + 2p)$. Since $p \geq 1/2$, the only value which gives a $\frac{1}{2}$-approximation is $\delta = 0$ and $p = q = 1/2$. For this case, we design a different counterexample.

Example 2.14. *Consider a directed graph on $X = \{a, b, c, d\}$ with 5 edges: (a, b) of weight $\frac{3}{2}p$, (b, a) of weight $\frac{3}{2}q$, (b, c) of weight 1, (c, d) of weight $\frac{3}{2}p$ and (d, c) of weight $\frac{3}{2}q$.*

It can be seen that $A = \{a, b\}$ is a local optimum for $p \leq 3/4$. For a given p', q', Algorithm SLS^* returns

$$\begin{aligned}\mathbf{E}[f(\mathcal{R}(\{a, b\}, \delta'))] &= p'q'(w(a, b) + w(b, a) + w(c, d) + w(d, c)) + p'^2 w(b, c) \\ &= 3p'q' + p'^2\end{aligned}$$

which is maximized for $p' = 3/4, q' = 1/4$. Then, the algorithm achieves 9/8 while the optimum is $f(\{b, d\}) = 1 + 3q = 5/2$ for $p = q = 1/2$. Thus, for small δ and p, q close to 1/2, the approximation factor is also bounded away from 1/2.

We showed that in order to get close to a $\frac{1}{2}$-approximation, we would need to run SLS^* with different values of δ and take the best solution. The analysis of such an algorithm remains elusive.

2.3 Inapproximability Results

In this section, we give hardness results for submodular maximization. Our results are of two flavors. First, we consider submodular functions that have a succint representation on the input, in the form of a sum of "building blocks" of constant size. Note that all the special cases such as Max Cut are of this type. For algorithms in this model, we prove complexity-theoretic inapproximability results. The strongest one is that in the general case, a $(3/4 + \epsilon)$-approximation for any fixed $\epsilon > 0$ would imply $P = NP$.

In the value oracle model, we show a much tighter result. Namely, any algorithm achieving a $(1/2+\epsilon)$-approximation for a fixed $\epsilon > 0$ would require an exponential number of queries to the value oracle. This holds even in the case of symmetric submodular functions, i.e. our 1/2-approximation algorithm is optimal in this model.

2.3.1 NP-hardness results

Our reductions are based on Håstad's 3-bit and 4-bit PCP verifiers [32]. Some inapproximability results can be obtained immediately from [32], by considering the known special cases of submodular maximization. The strongest result along these lines is that the Max Cut problem in 4-uniform hypergraphs is NP-hard to approximate within a factor better than 7/8. Therefore, we get the same hardness result for submodular maximization.

We obtain stronger hardness results by reductions from systems of parity equations. The parity function is not submodular, but we can obtain hardness results by a careful construction of a "submodular gadget" for each equation.

Theorem 2.15. *There is no polynomial-time* $(5/6 + \epsilon)$*-approximation algorithm to maximize a nonnegative symmetric submodular function, unless* $P = NP$.

Proof. Consider an instance of Max E4-Lin-2, a system of m parity equations, each on 4 boolean variables. Let's define two elements for each variable, T_i and F_i, corresponding to variable x_i being either true or false. For each equation e on variables (x_i, x_j, x_k, x_ℓ), we define a function $g_e(S)$. (This is our "submodular gadget".) Let $S' = S \cap \{T_i, F_i, T_j, F_j, T_k, F_k, T_\ell, F_\ell\}$. We say that S' is *valid quadruple*, if it defines a boolean assignment, i.e. contains exactly one element from each pair $\{T_i, F_i\}$. The function value is determined by S', as follows:

- If $|S'| < 4$, let $g_e(S) = |S'|$. If $|S'| > 4$, let $g_e(S) = 8 - |S'|$.

- If S' is a valid quadruple satisfying e, let $g_e(S) = 4$ (a *true quadruple*).
- If S' is a valid quadruple not satisfying e, let $g_e(S) = 8/3$ (a *false quadruple*).
- If $|S'| = 4$ but S' is not a valid quadruple, let $g_e(S) = 10/3$ (an *invalid quadruple*).

It can be verified that this is a submodular function, using the structure of the parity constraint. We define $f(S) = \sum_{e \in \mathcal{E}} g_e(S)$ by taking a sum over all equations. This is again a nonnegative submodular function. Observe that for each equation, it is more profitable to choose an invalid assignment than a valid assignment which does not satisfy the equation. Nevertheless, we claim that WLOG the maximum is obtained by selecting exactly one of T_i, F_i for each variable: Consider a set S and call a variable *undecided*, if S contains both or neither of T_i, F_i. For each equation with an undecided variable, we get value at most 10/3. Now, modify S by randomly selecting exactly one of T_i, F_i for each undecided variable. The new set S' induces a valid assignment to all variables. For equations which had a valid assignment already in S, the value does not change. Each equation which had an undecided varible is satisfied by S' with probability 1/2. Therefore, the expected value for each such equation is $(8/3 + 4)/2 = 10/3$, at least as before, and $\mathbf{E}[f(S')] \geq f(S)$. Hence there must exist a set S' such that $f(S') \geq f(S)$ and S' induces a valid assignment. Consequently, we have $OPT = \max f(S) = (8/3)m + (4/3)\#SAT$ where $\#SAT$ is the maximum number of satisfiable equations. Since it is NP-hard to distinguish whether $\#SAT \geq (1-\epsilon)m$ or $\#SAT \leq (1/2+\epsilon)m$, it is also NP-hard to distinguish between $OPT \geq (4-\epsilon)m$ and $OPT \leq (10/3+\epsilon)m$. $\square$

In the case of general nonnegative submodular functions, we improve the hardness threshold to 3/4. This hardness result is slightly more involved. It requires certain properties of Håstad's 3-bit verifier, implying that Max E3-Lin-2 is NP-hard to approximate even for linear systems of a special structure. We formalize this in the following lemma, which is needed to prove Theorem 2.17.

Lemma 2.16. *Fix any $\epsilon > 0$ and consider systems of weighted linear equations (of total weight 1) over boolean variables, partitioned into $\mathcal{X}$ and $\mathcal{Y}$, so that each equation contains 1 variable $x_i \in \mathcal{X}$ and 2 variables $y_j, y_k \in \mathcal{Y}$. Define a matrix $P \in [0,1]^{\mathcal{Y} \times \mathcal{Y}}$ where P_{jk} is the weight of all equations where the first variable from $\mathcal{Y}$ is y_j and the second variable is y_k. Then it's NP-hard to decide whether there is a solution satisfying equations of weight at least*

$1 - \epsilon$ *or whether any solution satisfies equations of weight at most* $1/2 + \epsilon$, *even in the special case where* P *is positive semidefinite.*

Proof. We show that the system of equations arising from Håstad's 3-bit verifier (see [32], pages 24-25) in fact satisfies the properties that we need. In his notation, the equations are generated by choosing $f \in \mathcal{F}_U$ and $g_1, g_2 \in \mathcal{F}_W$ where U, W, $U \subset W$, are randomly chosen and $\mathcal{F}_U, \mathcal{F}_W$ are the spaces of all ± 1 functions on $\{-1,+1\}^U$ and $\{-1,+1\}^W$, respectively. The equation corresponds to a 3-bit test on f, g_1, g_2 and its weight is the probability that the verifier performs this particular test. One variable is associated with $f \in \mathcal{F}_U$, indexing a bit in the Long Code of the first prover, and two variables are associated with $g_1, g_2 \in \mathcal{F}_W$, indexing bits in the Long Code of the second prover. This defines a natural partition of variables into $\mathcal{X}$ and $\mathcal{Y}$.

The actual variables appearing in the equations are determined by the folding convention; for the second prover, let's denote them by y_j, y_k where $j = \phi(g_1)$ and $k = \phi(g_2)$. The particular convention will not matter to us, as long as it is the same for both g_1 and g_2 (which is the case in [32]). Let P_{jk} be the probability that the selected variables corresponding to the second prover are y_j and y_k. Let $P^{U,W}_{jk}$ be the same probability, conditioned on a particular choice of U, W. Since P is a positive linear combination of $P^{U,W}$, it suffices to prove that each $P^{U,W}$ is positive semidefinite. The way that g_1, g_2 are generated (for given U, W) is that $g_1 : \{-1,+1\}^W \to \{-1,+1\}$ is uniformly random and $g_2(y) = g_1(y) f(y|_U) \mu(y)$, where $f : \{-1,+1\}^U \to \{-1,+1\}$ uniformly random and $\mu : \{-1,+1\}^W \to \{-1,+1\}$ is a "random noise", where $\mu(x) = 1$ with probability $1 - \epsilon$ and -1 with probability ϵ. The value of ϵ will be very small, certainly $\epsilon < 1/2$.

Both g_1 and g_2 are distributed uniformly (but not independently) from $\mathcal{F}_W$. The probability of sampling (g_1, g_2) is the same as the probability of sampling (g_2, g_1), hence $P^{U,W}$ is a symmetric matrix. It remains to prove positive semidefiniteness. Let's choose an arbitrary function $A : \mathcal{Y} \to \mathbb{R}$ and analyze

$$\sum_{jk} P^{U,W}_{jk} A(j)A(k) = \mathbf{E}_{g_1,g_2}[A(\phi(g_1))A(\phi(g_2))] = \mathbf{E}_{g_1,f,\mu}[A(\phi(g_1))A(\phi(g_1 f \mu))]$$

where g_1, f, μ are sampled as described above. If we prove that this quantity is always nonnegative, then $P^{U,W}$ is positive semidefinite. Let $B : \mathcal{F}_W \to \mathbb{R}$, $B = A \circ \phi$; i.e., we want to prove $\mathbf{E}[B(g_1)B(g_1 f \mu)] \geq 0$. We can expand B using its Fourier transform,

$$B(g) = \sum_{\alpha \subseteq \{-1,+1\}^W} \hat{B}(\alpha) \chi_\alpha(g).$$

Here, $\chi_\alpha(g) = \prod_{x \in \alpha} g(x)$ are the Fourier basis functions. We obtain

$$\mathbf{E}[B(g_1)B(g_1 f \mu)] = \sum_{\alpha,\beta \subseteq \{-1,+1\}^W} \mathbf{E}[\hat{B}(\alpha)\chi_\alpha(g_1)\hat{B}(\beta)\chi_\beta(g_1 f \mu)]$$

$$= \sum_{\alpha,\beta \subseteq \{-1,+1\}^W} \hat{B}(\alpha)\hat{B}(\beta) \prod_{x \in \alpha \Delta \beta} \mathbf{E}_{g_1}[g_1(x)] \mathbf{E}_f[\prod_{y \in \beta} f(y|_U)] \prod_{z \in \beta} \mathbf{E}_\mu[\mu(z)].$$

The terms for $\alpha \neq \beta$ are zero, since then $\mathbf{E}_{g_1}[g_1(x)] = 0$ for each $x \in \alpha \Delta \beta$. Therefore,

$$\mathbf{E}[B(g_1)B(g_1 f \mu)] = \sum_{\beta \subseteq \{-1,+1\}^W} \hat{B}^2(\beta)\ \mathbf{E}_f[\prod_{y \in \beta} f(y|_U)] \prod_{z \in \beta} \mathbf{E}_\mu[\mu(z)].$$

Now all the factors are nonnegative, since $\mathbf{E}_\mu[\mu(z)] = 1 - 2\epsilon > 0$ for every z and $\mathbf{E}_f[\prod_{y \in \beta} f(y|_U)] = 1$ or 0, depending on whether every string in $\{-1,+1\}^U$ is the projection of an even number of strings in β (in which case the product is 1) or not (in which case the expectation gives 0 by symmetry). To conclude,

$$\sum_{j,k} P^{U,W}_{jk} A(j)A(k) = \mathbf{E}[B(g_1)B(g_1 f \mu)] \geq 0$$

for any $A : \mathcal{Y} \to \mathbb{R}$, which means that each $P^{U,W}$ and consequently P is positive semidefinite. □

Theorem 2.17. *There is no polynomial-time* $(3/4 + \epsilon)$*-approximation algorithm to maximize a nonnegative submodular function, representable as a sum of functions on a constant number of elements, unless* $P = NP$.

Proof. We use a reduction from a system of linear equations as in Lemma 2.16. For each variable $x_i \in \mathcal{X}$, we have two elements T_i, F_i and for each variable $y_j \in \mathcal{Y}$, we have two elements $\tilde{T}_j, \tilde{F}_j$. Denote the set of equations by $\mathcal{E}$. Each equation e contains one variable from $\mathcal{X}$ and two variables from $\mathcal{Y}$. For each $e \in \mathcal{E}$, we define a submodular function $g_e(S)$ tailored to this structure. Assume that $S \subseteq \{T_i, F_i, \tilde{T}_j, \tilde{F}_j, \tilde{T}_k, \tilde{F}_k\}$, the elements corresponding to this equation; g_e does not depend on other than these 6 elements. We say that S is a valid triple, if it contains exactly one of each $\{T_i, F_i\}$.

- The value of each singleton T_i, F_i corresponding to a variable in $\mathcal{X}$ is 1.

- The value of each singleton $\tilde{T}_j, \tilde{F}_j$ corresponding to a variable in $\mathcal{Y}$ is 1/2.

- For $|S| < 3$, $g_e(S)$ is the sum of its singletons, except $g_e(\{T_i, F_i\}) = 1$ (*a weak pair*).
- For $|S| > 3$, $g_e(S) = g_e(\bar{S})$.
- If S is a valid triple satisfying e, let $g_e(S) = 2$ (*true triple*).
- If S is a valid triple not satisfying e, let $g_e(S) = 1$ (*false triple*).
- If S is an invalid triple containing exactly one of $\{T_i, F_i\}$ then $g_e(S) = 2$ (*type I*).
- If S is an invalid triple containing both/neither of $\{T_i, F_i\}$ then $g_e(S) = 3/2$ (*type II*).

The analysis of this gadget is more involved. We first emphasize the important points. A true triple gives value 2, while a false triple gives value 1. For invalid assignments of value 3/2, we can argue as before that a random valid assignment achieves expected value 3/2 as well, so we might as well choose a valid assignment. However, in this gadget we also have invalid triples of value 2 (type I). (We cannot avoid this due to submodularity.) Still, we prove that the optimum is attained for a valid boolean assignment. The main argument is, roughly, that if there are many invalid triples of type I, there must be also many equations where we get value only 1 (a weak pair). For this, we use the positive semidefinite property from Lemma 2.16.

Verifying that $g_e(S)$ is submodular is somewhat tedious. We have to check marginal values for two types of elements. First, consider T_i (or equivalently F_i), associated with a variable in $\mathcal{X}$. The marginal value $g_A(T_i)$ is equal to 1 if A does not contain F_i and contains at most two elements for variables in $\mathcal{Y}$, except when these elements would form a false triple with T_i; then $g_A(T_i) = 0$. Also, $g_A(T_i) = 0$ if A does not contain F_i and contains at least three elements for $\mathcal{Y}$, or if A contains F_i and at most two elements for $\mathcal{Y}$, except when these elements would form a true triple with T_i; then $g_A(T_i) = -1$. Finally, $g_A(T_i) = -1$ if A contains F_i and at least three elements for $\mathcal{Y}$. Hence, the marginal value depends in a monotone way on the subset A.

For an element like $\tilde{T}_j$, associated with a variable in $\mathcal{Y}$, we have $g_A(\tilde{T}_j) = 1/2$ if A does not contain any element for $\mathcal{X}$ and contains at most two elements for $\mathcal{Y}$, or A contains one element for $\mathcal{X}$ and at most one element for $\mathcal{Y}$ (but not forming a false triple with $\tilde{T}_j$), or A is a false triple, or A contains both elements for $\mathcal{X}$ and no other element for $\mathcal{Y}$. In all other cases, $g_A(\tilde{T}_j) = -1/2$. Here, the only way submodularity could be violated is that

$g_A(\tilde{T}_j) < f_B(\tilde{T}_j)$ for $A \subset B$ where A forms a false triple with $\tilde{T}_j$ and B is a false triple - but then $B = A \cup \{\tilde{T}_j\}$, contradiction.

We define $f(S) = \sum_{e \in \mathcal{E}} w(e) g_e(S)$ where $w(e)$ is the weight of equation e. We claim that $\max f(S) = 1 + \max w_{SAT}$, where w_{SAT} is the weight of satisfied equations. First, for a given boolean assignment, the corresponding set S selecting T_i or F_i for each variable achieves value $f(S) = w_{SAT} \cdot 2 + (1 - w_{SAT}) \cdot 1 = 1 + w_{SAT}$. The non-trivial part is proving that the optimum $f(S)$ is attained for a set inducing a valid boolean assignment.

Consider any set S and define $V : \mathcal{E} \to \{-1, 0, +1\}$ where $V(e) = +1$ if S induces a satisfying assignment to equation e, $V(e) = -1$ if S induces a non-satisfying assignment to e and $V(e) = 0$ if S induces an invalid assignment to e. Also, define $A : \mathcal{Y} \to \{-1, 0, +1\}$, where $A(y_j) = |S \cap \{\tilde{T}_j, \tilde{F}_j\}| - 1$, i.e. $A(y_j) = 0$ if S induces a valid assignment to y_j, and $A(y_j) = \pm 1$ if S contains both/neither of $\tilde{T}_j, \tilde{F}_j$. Observe that for an equation e whose $\mathcal{Y}$-variables are y_j, y_k, only one of $V(e)$ and $A(y_j)A(y_k)$ can be nonzero. The gadget $g_e(S)$ is designed in such a way that

$$g_e(S) \le \frac{1}{2}(3 - A(y_j)A(y_k) + V(e)).$$

This can be checked case by case: for valid assignments, $A(y_j)A(y_k) = 0$ and we get value 2 or 1 depending on $V(e) = \pm 1$. For invalid assignments, $V(e) = 0$; if at least one of the variables y_j, y_k has a valid assignment, then $A(y_j)A(y_k) = 0$ and we can get at most $3/2$ (an invalid triple of type II). If both y_j, y_k are invalid and $A(y_j)A(y_k) = 1$, then we can get only 1 (a weak pair or its complement) and if $A(y_j)A(y_k) = -1$, we can get 2 (an invalid triple of type I). The total value is

$$f(S) = \sum_{e \in \mathcal{E}} w(e) g_e(S) \le \sum_{j,k} \sum_{e=(x_i, y_j, y_k)} w(e) \cdot \frac{1}{2}(3 - A(y_j)A(y_k) + V(e)).$$

Now we use the positive semidefinite property of our linear system, which means that

$$\sum_{j,k} \sum_{e=(x, y_j, y_k)} w(e) A(y_j) A(y_k) = \sum_{j,k} P_{jk} A(y_j) A(y_k) \ge 0$$

for any function A. Hence, $f(S) \le \frac{1}{2} \sum_{e \in \mathcal{E}} w(e)(3 + V(e))$. Let's modify S into a valid boolean assignment by choosing randomly one of T_i, F_i for all variables such that S contains both/neither of T_i, F_i. Denote the new set by S' and the equations containing any randomly chosen variable by $\mathcal{R}$. We satisfy each equation in $\mathcal{R}$ with probability $1/2$, which gives expected value 3/2 for each such equation, while the value for other equations remains unchanged.

$$\mathbf{E}[f(S')] = \frac{3}{2}\sum_{e\in\mathcal{R}} w(e) + \frac{1}{2}\sum_{e\in\mathcal{E}\setminus\mathcal{R}} w(e)(3+V(e)) = \frac{1}{2}\sum_{e\in\mathcal{E}} w(e)(3+V(e)) \geq f(S).$$

This means that there is a set S' of optimal value, inducing a valid boolean assignment. □

2.3.2 Smooth submodular functions

In this section, we introduce a formalism which is very convenient for constructing examples of submodular functions; in particular, for proving our query complexity bound on maximizing symmetric submodular functions. Instead of constructing explicit discrete functions, we want to define continuous function with certain analytic properties.

Assume that a ground set X is partitioned into $(X_1, X_2, \ldots, X_n)$ where each X_i has a large number of elements. The function f that we construct depends only on the fractions of $X_1, \ldots, X_n$ that S contains: $x_i = |S \cap X_i|/|X_i|$. We think of x_i as real variables in $[0,1]$ and define our function as $\tilde{f}(x_1, \ldots, x_n)$. This induces a discrete function on X. The following lemma provides a link between the analytic and discrete properties that we are interested in.

Lemma 2.18. *Let $X = X_1 \cup X_2 \cup \ldots \cup X_n$ as above and let $\tilde{f} : [0,1]^n \to \mathbb{R}$ be a function with continuous first partial derivatives, and second partial derivatives almost everywhere. Define a discrete function $f : X \to \mathbb{R}$ so that*

$$f(S) = \tilde{f}\left(\frac{|S \cap X_1|}{|X_1|}, \ldots, \frac{|S \cap X_n|}{|X_n|}\right).$$

Then,

1. *If $\tilde{f}(x_1, \ldots, x_n) = \tilde{f}(1 - x_1, \ldots, 1 - x_n)$ everywhere, then the function f is symmetric.*
2. *If $\frac{\partial \tilde{f}}{\partial x_i} \geq 0$ everywhere for each i, then the function f is monotone.*
3. *If $\frac{\partial^2 \tilde{f}}{\partial x_i \partial x_j} \leq 0$ almost everywhere [1] for any i, j, then the function f is submodular.*

[1] To be more precise, on any axis-parallel line there are only finitely many points where $\frac{\partial^2 \tilde{f}}{\partial x_i \partial x_j}$ might not be defined.

Proof. The first property is clear: If $\tilde{f}(\vec{x}) = \tilde{f}(\vec{y})$ where $y_i = 1 - x_i$, then $f(S) = f(X \setminus S)$.

For monotonicity, it's sufficient to observe that if $\frac{\partial f}{\partial x_i} \geq 0$, then $\tilde{f}$ is nondecreasing in each coordinate. Hence, adding elements cannot decrease the value of f.

For the submodularity condition, fix an element in $a \in X_i$ and consider a set S parametrized by $x_i = |S \cap X_i|/|X_i|$. The marginal value of a added to S is equal to

$$\begin{aligned} f_S(a) &= \tilde{f}(x_1, \ldots, x_i + 1, \ldots, x_n) - \tilde{f}(x_1, \ldots, x_i, \ldots, x_n) \\ &= \int_0^1 \frac{\partial \tilde{f}}{\partial x_i}(x_1, \ldots, x_i + t, \ldots, x_n) dt. \end{aligned}$$

We want to prove that $f_S(a)$ cannot increase by adding elements to S, i.e. by increasing any coordinate x_j. Because $\frac{\partial \tilde{f}}{\partial x_i}$ is continuous and its derivative along x_j, $\frac{\partial^2 \tilde{f}}{\partial x_i \partial x_j}$, is non-positive except at finitely many points, $\frac{\partial \tilde{f}}{\partial x_i}$ is nonincreasing in x_j. By shifting the entire integral to a higher value of x_j, the marginal value cannot increase. $\square$

2.3.3 Query complexity results

Finally, we prove that our $\frac{1}{2}$-approximation for symmetric submodular functions is optimal in the value oracle model. First, we present a similar result for the "random set" model, which illustrates some of the ideas needed for the more general result.

Proposition 2.19. *For any $\delta > 0$, there is $\epsilon > 0$ such that for any (random) sequence of queries $\mathcal{Q} \subseteq 2^X$, $|\mathcal{Q}| \leq 2^{\epsilon n}$, there is a nonnegative submodular function f such that (with high probability) for all queries $Q \in \mathcal{Q}$,*

$$f(Q) \leq \left(\frac{1}{4} + \delta\right) OPT.$$

Proof. Let $\epsilon = \delta^2/32$ and fix a sequence $\mathcal{Q} \subseteq 2^X$ of $2^{\epsilon n}$ queries. We prove the existence of f by the probabilistic method. Consider functions corresponding to cuts in a complete bipartite directed graph on (C, D), $f_C(S) = |S \cap C| \cdot |\bar{S} \cap D|$. We choose a uniformly random $C \subseteq X$ and $D = X \setminus C$. The idea is that for any query, a typical C bisects both Q and its complement, which means that $f_C(Q)$ is roughly $\frac{1}{4}OPT$. We call a query $Q \in \mathcal{Q}$ "successful", if $f_C(Q) > (\frac{1}{4} + \delta)OPT$. Our goal is to prove that with high probability, C avoids any successful query.

We use Chernoff's bound: For any set $A \subseteq X$ of size a,

$$\Pr[|A \cap C| > \frac{1}{2}(1+\delta)|A|] = \Pr[|A \cap C| < \frac{1}{2}(1-\delta)|A|] < e^{-\delta^2 a/2}.$$

With probability at least $1 - 2e^{-2\delta^2 n}$, the size of C is in $[(\frac{1}{2}-\delta)n, (\frac{1}{2}+\delta)n]$, so we can assume this is the case. We have $OPT \geq (\frac{1}{4}-\delta^2)n^2 \geq \frac{1}{4}n^2/(1+\delta)$ (for small $\delta > 0$). No query can achieve $f_C(Q) > (\frac{1}{4}+\delta)OPT \geq \frac{1}{16}n^2$ unless $|Q| \in [\frac{1}{16}n, \frac{15}{16}n]$, so we can assume this is the case for all queries. By Chernoff's bound, $\Pr[|Q \cap C| > \frac{1}{2}(1+\delta)|Q|] < e^{-\delta^2 n/32}$ and $\Pr[|\bar{Q} \cap D| > \frac{1}{2}(1+\delta)|\bar{Q}|] < e^{-\delta^2 n/32}$. If neither of these events occurs, the query is not successful, since $f_C(Q) = |Q \cap C| \cdot |\bar{Q} \cap D| < \frac{1}{4}(1+\delta)^2|Q| \cdot |\bar{Q}| \leq \frac{1}{16}(1+\delta)^2 n^2 \leq \frac{1}{4}(1+\delta)^3 OPT \leq (\frac{1}{4}+\delta)OPT$.

For now, fix a sequence of queries. By the union bound, we get that the probability that any query is successful is at most $2^{\epsilon n} 2e^{-\delta^2 n/32} = 2(2/e)^{\epsilon n}$. Thus with high probability, there is no successful query for C. Even for a random sequence, the probabilistic bound still holds by averaging over all possible sequences of queries. We can fix any C for which the bound is valid, and then the claim of the lemma holds for the submodular function f_C. $\square$

This means that in the model where an algorithm only samples a sequence of polynomially many sets and returns the one of maximal value, we cannot improve our $\frac{1}{4}$-approximation (Section 2.1). Perhaps surprisingly, this example can be modified for the model of adaptive algorithms with value queries, to show that our $\frac{1}{2}$-approximation for symmetric submodular functions is optimal, even among adaptive algorithms!

Theorem 2.20. *For any $\epsilon > 0$, there are instances of nonnegative symmetric submodular maximization, such that there is no (adaptive, possibly randomized) algorithm using less than $e^{\epsilon^2 n/16}$ queries that always finds a solution of expected value at least $(1/2+\epsilon)OPT$.*

Proof. We construct a nonnegative symmetric submodular function on $[n] = C \cup D$ by defining a smooth function $f(x,y)$ of two variables, as in Section 2.3.2. By slight abuse of notation, we denote the correponding smooth and discrete functions by the same symbol. Our function will have the following properties.

- When $|x-y| \leq \epsilon$, the function has the form

$$f(x,y) = (x+y)(2-x-y)$$

which corresponds to the cut function of a complete graph. The value depends only on $x+y$, and the maximum attained in this range is $f(1/2,1/2) = 1$.

- When $|x - y| > \epsilon$, the function has the form

  $$f(x, y) = 2x(1 - y) + 2(1 - x)y - O(\epsilon),$$

  corresponding (up to an $O(\epsilon)$ term) to the cut function of a complete bipartite graph on (C, D) with edge weights doubled compared to the previous case. The maximum in this range is $f(0, 1) = f(1, 0) = 2 - O(\epsilon)$.

If we construct such a function, we can argue as follows. Consider any algorithm, for now deterministic. (For a randomized algorithm, let's condition on its random bits.) Let the partition (C, D) be random and unknown to the algorithm. The algorithm issues some queries Q to the value oracle. Call Q "unbalanced", if $x = |Q \cap C|/|C|$ differs from $y = |Q \cap D|/|D|$ by more than ϵ. For any query Q, the probability that Q is unbalanced is at most $e^{-\epsilon^2 n/8}$, by the Chernoff bound. Therefore, for any fixed sequence of $e^{\epsilon^2 n/16}$ queries, the probability that any query is unbalanced is still at most $e^{\epsilon^2 n/16} \cdot e^{-\epsilon^2 n/8} = e^{-\epsilon^2 n/16}$. As long as queries are balanced, the algorithm gets the same answer regardless of (C, D). Hence, it follows the same path of computation and issues the same queries. With probability at least $1 - e^{-\epsilon^2 n/16}$, all its queries will be balanced and it will never find out any information about the partition (C, D). For a randomized algorithm, we can now average over its random choices; still, with probability at least $1 - e^{-\epsilon^2 n/16}$ the algorithm will never query any unbalanced set.

Alternatively, consider a function $g(x, y)$ which is defined by $g(x, y) = (x + y)(1 - x - y)$ *everywhere.* This is a smooth function with nonpositive second partial derivatives, hence it induces a submodular function by Lemma 2.18. We proved that with high probability, the algorithm will never query a set where $f \neq g$ and hence cannot distinguish between the two instances. However, $\max f(x, y) = 2 - O(\epsilon)$, while $\max g(x, y) = 1$. This means that there is no $(1/2 + \epsilon)$-approximation algorithm with a subexponential number of queries, for any $\epsilon > 0$.

It remains to construct the function f and prove its submodularity. For this purpose, we use Lemma 2.18. In the range where $|x - y| \leq \epsilon$, we already defined $f(x, y) = (x + y)(2 - x - y)$. In the range where $|x - y| \geq \epsilon$, let us define

$$\begin{aligned} f(x, y) &= (x + y)(2 - x - y) + (|x - y| - \epsilon)^2 \\ &= 2x(1 - y) + 2(1 - x)y - 2\epsilon|x - y| + \epsilon^2. \end{aligned}$$

The ϵ-terms are chosen so that $f(x, y)$ is a smooth function on the boundary of the two regions. E.g., for $x - y = \epsilon$, we get $f(x, y) = (x + y)(2 - x - y)$

for both expressions. Moreover, the partial derivatives also extend smoothly. We have

- $\frac{\partial f}{\partial x} = 2 - 2x - 2y$ for $|x - y| \le \epsilon$,
- $\frac{\partial f}{\partial x} = 2 - 4y - 2\epsilon$ for $x - y > \epsilon$,
- $\frac{\partial f}{\partial x} = 2 - 4y + 2\epsilon$ for $x - y < -\epsilon$.

It's easy to see that the derivative values coincide for $|x - y| = \epsilon$; similarly for $\frac{\partial f}{\partial y}$. The second partial derivatives are clearly nonpositive in all ranges, except on the boundary $|x - y| = \epsilon$ where they are not defined. Due to Lemma 2.18, this defines a submodular function. $\square$

2.4 Concluding remarks

For nonadaptive algorithms, one remaining question is whether our algorithms can be derandomized and implemented by querying only a predetermined collection of polynomially many sets.

For adaptive algorithms, it would be nice to achieve a $\frac{1}{2}$-approximation in the general case (as was done for Max Di-Cut in [31]). We lean toward the opinion that this might be possible, which would settle the approximation status of submodular maximization in the value oracle model.

It is still conceivable that a better than $\frac{1}{2}$-approximation might be achieved in a model of computation requiring an explicit representation of $f(S)$. Our NP-hardness results in this case are still quite far away from $1/2$. Considering the known approximation results for Max Cut, such an improvement would most likely require semidefinite programming. Currently, we do not know how to implement such an approach.

Chapter 3

Submodular Maximization over a Matroid

Problem: *Given a monotone submodular function $f : 2^X \rightarrow \mathbb{R}_+$ and a matroid on $\mathcal{M} = (X, \mathcal{I})$, maximize $f(S)$ over all independent sets $S \in \mathcal{I}$.*

As we discussed in Section 1.2.2, this problem generalizes the classical Max k-cover problem, where we take a coverage-type function for f and a uniform matroid for $\mathcal{M}$. (See Section 1.1 for definitions.) In this special case, we know that the greedy algorithm provides a $(1 - 1/e)$-approximation and this is optimal unless $P = NP$ [17]. Furthermore, the greedy algorithm gives a $(1 - 1/e)$-approximation for any monotone submodular function and a uniform matroid [45]. For any monotone submodular function and an arbitrary matroid, the greedy algorithm provides a $\frac{1}{2}$-approximation [46].

On the other hand, a $(1 - 1/e)$-approximation was also known in certain cases where the matroid is not uniform. One example is the Maximum Coverage Problem with Group Budget Constraints [5, 2], where we have groups of sets and we want to choose one set from each group to maximize the size of their union. Another example is the Submodular Welfare Problem with coverage-type utilities [14]. Both of these problems can be seen as coverage maximization under a partition matroid constraint (for the reduction from Submodular Welfare, see Section 1.2.3). Hence, there is some indication that a $(1 - 1/e)$-approximation might be possible for any monotone submodular function and any matroid. We indeed achieve this goal in this chapter.

Theorem 3.1. *Let $f : 2^N \rightarrow \mathbb{R}_+$ be a monotone submodular function given by a value oracle, and $\mathcal{M} = (N, \mathcal{I})$ a matroid given by a membership oracle. Then there is a randomized polynomial time $(1 - 1/e - o(1))$-approximation to the problem $\max_{S \in \mathcal{I}} f(S)$.*

3.1 Preliminaries

Given a submodular function $f : N \rightarrow \mathbb{R}^+$ and $A \subset N$, the function f_A defined by $f_A(S) = f(S \cup A) - f(A)$ is also submodular. Further, if f is monotone, f_A is also monotone. For $i \in N$, we abbreviate $S \cup \{i\}$ by $S + i$. By $f_A(i)$, we denote the "marginal value" $f(A+i) - f(A)$. Submodularity is equivalent to $f_A(i)$ being non-increasing as a function of A for every fixed i.

Given a matroid $\mathcal{M} = (N, \mathcal{I})$, we denote by $r_{\mathcal{M}}$ the rank function of $\mathcal{M}$ where $r_{\mathcal{M}}(A) = \max\{|S| : S \subseteq A, S \in \mathcal{I}\}$. The rank function is monotone and submodular.

By $P(\mathcal{M})$, we denote the *matroid polytope* of $\mathcal{M}$:

$$P(\mathcal{M}) = \text{conv } \{\mathbf{1}_I \mid I \in \mathcal{I}\},$$

where $\mathbf{1}_I$ is the characteristic vector of I. For a vector $y \in \mathbb{R}^N$, we denote $y(S) = \sum_{j \in S} y_j$. Edmonds showed that an equivalent definition of the matroid polytope is

$$P(\mathcal{M}) = \{y \geq 0 \mid \forall S; y(S) \leq r_{\mathcal{M}}(S)\}.$$

Further, given a membership oracle for $\mathcal{M}$ (i.e., given $S \subseteq N$, the oracle answers whether $S \in \mathcal{I}$ or not), one can optimize linear functions over $P(\mathcal{M})$. (See [51] for more details.)

A base of $\mathcal{M}$ is a set $S \in \mathcal{I}$ such that $r_{\mathcal{M}}(S) = r_{\mathcal{M}}(N)$. The base polytope $B(\mathcal{M})$ of $\mathcal{M}$ is given by $\{y \in P(\mathcal{M}) \mid y(N) = r_{\mathcal{M}}(N)\}$. The extreme points of $B(\mathcal{M})$ are the characteristic vectors of the bases of $\mathcal{M}$. Given the problem $\max_{S \in \mathcal{I}} f(S)$, where $\mathcal{M} = (N, \mathcal{I})$ is a matroid, there always exists an optimum solution S^* where S^* is a base of $\mathcal{M}$. Note that this is false if f is not monotone. Thus, for monotone f, it is equivalent to consider the problem $\max_{S \in \mathcal{B}} f(S)$ where $\mathcal{B}$ is the set of bases of $\mathcal{M}$. See [51] for more details on matroids and polyhedral aspects.

3.2 Overview

Given a monotone submodular function $f : 2^N \rightarrow \mathbb{R}^+$ and a matroid $\mathcal{M} = (N, \mathcal{I})$, we wish to solve $\max_{S \in \mathcal{I}} f(S)$. Let $y_i \in \{0, 1\}$ be a variable that indicates whether $i \in S$. Then $\max_{S \in \mathcal{I}} f(S)$ can be written as the following problem: $\max\{f(y) : y \in P(\mathcal{M}), y \in \{0, 1\}^N\}$. As we observed in Section 3.1, this is equivalent to $\max\{f(y) : y \in B(\mathcal{M}), y \in \{0, 1\}^N\}$ where $B(\mathcal{M})$ is the base polytope of $\mathcal{M}$.

Our framework relies on the ability to solve (at least approximately) a relaxation of the problem in polynomial time. To obtain a relaxation, we define $F : [0,1]^N \to \mathbb{R}_+$ by

$$F(y) = \mathbf{E}[f(\hat{y})]$$

where $\hat{y}$ a randomly rounded version of y, with each coordinate $\hat{y}_j$ independently rounded to 1 with probability y_j and 0 otherwise. Our algorithm proceeds as follows.

1. We consider the (non-linear) problem
$$\max\{F(y) : y \in P(\mathcal{M})\}.$$
We show how to find $y^* \in P(\mathcal{M})$ such that $F(y^*) \geq (1 - 1/e) \cdot OPT$.

2. In the second stage, we use the *pipage rounding* technique to convert y^* into an integral solution S. Pipage rounding guarantees that at the end, we have
$$f(S) \geq F(y^*) \geq (1 - 1/e) \cdot OPT.$$

In keeping with the chronological order of how things evolved, we turn to the second stage of our framework first.

3.3 Pipage rounding

Ageev and Sviridenko [2] developed an elegant technique for rounding solutions of linear and non-linear programs that they called “pipage rounding”. Subsequently, Srinivasan [54] and Gandhi et al. [25] interpreted some applications of pipage rounding as a deterministic variant of dependent randomized rounding. In a typical scenario, randomly rounding a fractional solution of a linear program does not preserve the feasibility of constraints. Nevertheless, the techniques of [2, 54, 25] show that randomized rounding can be applied in a certain controlled way to guide a solution that respects a certain class of constraints. Gruia Calinescu, Chandra Chekuri and Martin Pál observed that this framework can be also adapted to submodular maximization over a matroid polytope, and this is the version of pipage rounding that we describe here.

In the following, we assume that we start with a point y^* in the basis polytope $B(\mathcal{M})$. The pipage rounding technique aims to convert y^* into an integral solution. Given $y \in [0,1]^n$ we say that i is fractional in y if $0 < y_i < 1$. For $y \in P(\mathcal{M})$, a set $A \subseteq N$ is *tight* if $y(A) = r_{\mathcal{M}}(A)$. The following well-known proposition follows easily from the submodularity of the rank function $r_{\mathcal{M}}$.

Proposition 3.2. *If A and B are two tight sets with respect to y then $A \cap B$ and $A \cup B$ are also tight with respect to y.*

Proof. Using $y(A) = r_{\mathcal{M}}(A)$, $y(B) = r_{\mathcal{M}}(B)$, $y(A \cap B) \leq r_{\mathcal{M}}(A \cap B)$, $y(A \cup B) \leq r_{\mathcal{M}}(A \cup B)$, the submodularity of $r_{\mathcal{M}}$ and the linearity of y, we get

$$\begin{aligned} y(A) + y(B) &= r_{\mathcal{M}}(A) + r_{\mathcal{M}}(B) \geq r_{\mathcal{M}}(A \cap B) + r_{\mathcal{M}}(A \cup B) \\ &\geq y(A \cap B) + y(A \cup B) = y(A) + y(B). \end{aligned}$$

Therefore, all inequalities above must be tight. □

We are interested in tight sets that contain a fractional variable. Observe that a tight set with a fractional variable has at least two fractional variables. Given a tight set A with fractional variables i, j, we let $y_{ij}(t)$ be the vector obtained by adding t to y_i, subtracting t from y_j and leaving the other values unchanged. We define a real-valued function F^y_{ij} where $F^y_{ij}(t) = F(y_{ij}(t))$. We want to argue that $F(y_{ij}(t))$ cannot decrease for both $t > 0$ and $t < 0$, and hence we can move in one of the two directions without losing on the value of $F(y)$. The following lemma gives us the property that we need.

Lemma 3.3. *For any submodular f, if $F(y) = \mathbf{E}[f(\hat{y})]$, then F^y_{ij} is convex for any $y \in [0,1]^N$ and $i, j \in N$.*

Proof. For $S \subseteq N \setminus \{i,j\}$ and $y \in [0,1]^N$, let

$$p_y(S) = \Pr[\hat{y}|_{N \setminus \{i,j\}} = \mathbf{1}_S] = \prod_{l \in S} y_l \prod_{l \in N \setminus \{i,j\} \setminus S} (1 - y_l).$$

Then

$$\begin{aligned} F(y) = \sum_{S \subseteq N \setminus \{i,j\}} p_y(S) \quad & ((1-y_i)(1-y_j) f(S) + (1-y_i) y_j f(S+j) \\ & + y_i(1-y_j) f(S+i) + y_i y_j f(S+i+j)). \end{aligned}$$

We have $F^y_{ij}(t) = F(y_{ij}(t))$. Let $x = y_{ij}(t)$, i.e. $x_i = y_i + t$, $x_j = y_j - t$ and $x_l = y_l$ for all $l \in N \setminus \{i,j\}$. Hence it follows that $p_x(S) = p_y(S)$ for $S \subseteq N \setminus \{i,j\}$. It can be seen that $F(y_{ij}(t)) = F(x) = c_2 t^2 + c_1 t + c_0$ where c_2, c_1, c_0 do not depend on t (they depend only on y and f). Thus to show that $F^y_{ij}(t)$ is convex in t, it is sufficient to prove that $c_2 \geq 0$. It is easy to check that

$$c_2 = \sum_{S \subseteq N \setminus \{i,j\}} p_y(S)(-f(S) + f(S+j) + f(S+i) - f(S+i+j)).$$

By submodularity, $f(S+i) + f(S+j) \geq f((S+i) \cap (S+j)) + f((S+i) \cup (S+j)) = f(S) + f(S+i+j)$ which proves that $c_2 \geq 0$. □

Therefore, we will be able to modify the fractional variables y_i, y_j by increasing one of them and decreasing the other, without decreasing the value of $F(y)$. We do this until we hit another constraint of the matroid polytope. The subroutine **HitWall** performs this task. It returns a new tight set A, which contains exactly one of i, j: either $i \in A$ if the new tight constraint is $y(A) = r(A)$, or $A = \{j\}$ if the new tight constraint is $y_j = 0$. In the first case, we can continue with a smaller tight set $A \cap T$ (due to Prop. 3.2). In the second case, we restart with $T = N$.

```
Subroutine HitWall($y$, $i$, $j$):
  Denote $\mathcal{A} = \{A \subseteq N : i \in A, j \notin A\}$;
  Find $\delta = \min_{A \in \mathcal{A}}(r(A) - y(A))$ and $A \in \mathcal{A}$ achieving this;
  If $y_j < \delta$ then $\{\delta \leftarrow y_j,\ A \leftarrow \{j\}\}$;
  $y_i \leftarrow y_i + \delta,\ y_j \leftarrow y_j - \delta$;
  Return $(y, A)$.

Algorithm PipageRound($y$):
  While ($y$ is not integral) do
    $T \leftarrow N$;
    While ($T$ contains fractional variables) do
      Pick $i, j \in T$ fractional;
      $(y^+, A^+) \leftarrow$ HitWall$(y, i, j)$;
      $(y^-, A^-) \leftarrow$ HitWall$(y, j, i)$;
      If $F(y^+) \geq F(y^-)$
         Then $\{y \leftarrow y^+, T \leftarrow T \cap A^+\}$;
         Else $\{y \leftarrow y^-, T \leftarrow T \cap A^-\}$;
    EndWhile
  EndWhile
  Output $y$.
```

Lemma 3.4. *Given $y \in B(\mathcal{M})$ and oracle access to $F(y)$,* **PipageRound***(y) outputs in polynomial time an integral solution $S \in \mathcal{I}$ of value $f(S) \geq F(y)$.*

Proof. The algorithm does not alter a variable y_i once $y_i \in \{0, 1\}$. An invariant maintained by the algorithm is that $y \in B(\mathcal{M})$ and T is a tight set. To verify that $y \in B(\mathcal{M})$, observe that $y(N) = r(N)$ remains unchanged throughout the algorithm; we need to check that $y(S) \leq r(S)$ remains satisfied for all sets S. Consider the subroutine HitWall. The only sets whose value $y(S)$ increases are those containing i and not containing j, i.e. $S \in \mathcal{A}$. We increase y_i by at most $\delta = \min_{S \in \mathcal{A}}(r(S) - y(S))$, therefore $y(S) \leq r(S)$

is still satisfied for all sets. We also make sure that we don't violate nonnegativity, by checking whether $y_j < \delta$. In case $y_j < \delta$, the procedure makes y_j zero and returns $A = \{j\}$.

Concerning the tightness of T, we initialize $T \leftarrow N$ which is tight because $y \in B(\mathcal{M})$. After calling HitWall, we obtain sets A^+, A^- which are tight for y^+ and y^-, respectively. The new set T is obtained by taking an intersection with one of these sets; in any case, we get a new tight set T at the end of the inner loop, due to Proposition 3.2.

Note that each of the sets A^+, A^- returned by HitWall contains exactly one of the elements i, j. Therefore, the size of T decreases after the execution of the inner loop. As long as we do not make any new variable integral, one of the fractional variables y_i, y_j is still in the new tight set T and so we can in fact find a pair of fractional variables in T. However, due to the decreasing size of T, we cannot repeat this more than $n-1$ times. At some point, we must make a new variable integral and then we restart the inner loop with $T = N$. The outer loop can also iterate at most n times, since the number of integral variables increases after each outer loop.

The non-trivial step in the algorithm is the minimization of $r(S) - y(S)$ over $\mathcal{A} = \{S \subseteq N : i \in S, j \notin S\}$. Since $r(S) - y(S)$ is a submodular function, minimization can be implemented in polynomial time [21].

Finally, we need to show that the value of $F(y)$ does not decrease throughout the algorithm. In each step, we choose the larger of $F(y^+) = F^y_{ij}(\epsilon^+)$ and $F(y^-) = F^y_{ij}(\epsilon^-)$ where $\epsilon^- < 0 < \epsilon^+$. If both values were smaller than $F(y)$, we would have $F^y_{ij}(\epsilon^-) < F^y_{ij}(0) > F^y_{ij}(\epsilon^+)$ but this would contradict Lemma 3.3 which says that F^y_{ij} is convex. $\square$

We remark that in fact we do not have access to exact values of $F(y)$. However, we can estimate $F(y)$ by random sampling. This can be done within an additive error $\pm OPT/poly(n)$ in polynomial time. Due to this, we may lose $o(OPT)$ overall which decreases the approximation factor by $o(1)$; we omit the details.

3.4 The smooth greedy algorithm

Now we return to the first stage, where we need to find a point $y \in B(\mathcal{M})$ with a good value of $F(y) = \mathbf{E}[f(\hat{y})]$. What follows is a conceptual description of our algorithm. It can be viewed as a continuous greedy algorithm, which strives to move in the direction of maximum gain, while staying in the matroid polytope.

A dynamic process.

- For each $y \in [0,1]^N$, define $I(y) \in \mathcal{I}$ to be an independent set maximizing $\sum_{j \in I} \frac{\partial F}{\partial y_j}(y)$.
- Let $y(t)$ be a trajectory starting at $y(0) = \vec{0}$ and satisfying the following differential equation:
$$\frac{dy}{dt} = \mathbf{1}_{I(y)}.$$
- We run this process for $t \in [0,1]$ and return $y(1)$.

We sketch an argument explaining why this leads to a $(1-1/e)$-approximation. First, it can be seen that $y(1)$ is a convex linear combination of independent sets,
$$y(1) = \int_0^1 \mathbf{1}_{I(y(t))} dt.$$
In fact, each $I(y(t))$ is WLOG a basis of $\mathcal{M}$, and hence $y(1) \in B(\mathcal{M})$. To analyze the value of $F(y(t))$, we first show the following bound.

Lemma 3.5. *Let $OPT = \max_{S \in \mathcal{I}} f(S)$. Consider any $y \in [0,1]^N$ and let R denote a random set corresponding to $\hat{y}$, with elements sampled independently according to y_j. Then*
$$OPT \leq F(y) + \max_{I \in \mathcal{I}} \sum_{j \in I} \mathbf{E}[f_R(j)] \leq F(y) + \max_{I \in \mathcal{I}} \sum_{j \in I} \frac{\partial F}{\partial y_j}.$$

Proof. Fix an optimal solution $O \in \mathcal{I}$. By submodularity, we have
$$OPT = f(O) \leq f(R) + \sum_{j \in O} f_R(j)$$
for any set R. By taking the expectation over a random R as above,
$$OPT \leq \mathbf{E}[f(R) + \sum_{j \in O} f_R(j)] = F(y) + \sum_{j \in O} \mathbf{E}[f_R(j)] \leq F(y) + \max_{I \in \mathcal{I}} \sum_{j \in I} \mathbf{E}[f_R(j)].$$
To prove the second inequality in the lemma, write explicitly
$$F(y) = \mathbf{E}[f(R)] = \sum_{R \subseteq N} f(R) \prod_{j \in R} y_j \prod_{j \notin R} (1 - y_j).$$
Note that $F(y)$ is linear in each y_j and therefore $\frac{\partial F}{\partial y_j} = F(y|y_j = 1) - F(y|y_j = 0) = \mathbf{E}[f(R+j)] - \mathbf{E}[f(R-j)]$. By monotonicity, $\frac{\partial F}{\partial y_j} \geq \mathbf{E}[f(R+j)] - \mathbf{E}[f(R)] = \mathbf{E}[f_R(j)]$. □

Now consider how $F(y(t))$ evolves as time progresses. We know that $\frac{dy}{dt} = \mathbf{1}_{I(y)}$ and $I(y)$ is an independent set maximizing $\sum_{j \in I} \frac{\partial F}{\partial y_j}$. By the chain rule, we get

$$\frac{dF}{dt} = \sum_j \frac{\partial F}{\partial y_j} \frac{dy_j}{dt} = \sum_{j \in I(y)} \frac{\partial F}{\partial y_j} = \max_{I \in \mathcal{I}} \sum_{j \in I} \frac{\partial F}{\partial y_j} \geq OPT - F(y(t))$$

using Lemma 3.5. This implies that $F(y(t))$ dominates the solution of the differential equation $\phi'(t) = OPT - \phi(t)$, which is $\phi(t) = (1 - e^{-t})OPT$. For $t = 1$, we get $F(y(1)) \geq \phi(1) = (1 - 1/e)OPT$.

Inspired by the continuous dynamic process above, we design a finite algorithm that mimicks this process. Recall that the process follows a path whose direction is given by replacing $F(y)$ locally by a *linear function* and optimizing it over $\mathcal{I}$. Luckily, we are able to find such a direction efficiently.

The Smooth Greedy Algorithm.

1. Let $\delta = 1/n^2$ where $n = |N|$ is the total number of elements. Start with $t = 0$ and $y(0) = \vec{0}$.

2. Let $R(t)$ contain each item j independently with probability $y_j(t)$. For each $j \in N$, estimate
$$\omega_j(t) = \mathbf{E}[f_{R(t)}(j)].$$
By repeated sampling, we can get an estimate within an additive error $\pm OPT/poly(n)$ in polynomial time w.h.p.

3. Let $I(t)$ be an independent set in $\mathcal{M}$, maximizing $\sum_{j \in I} \omega_j(t)$. This can be found by the greedy algorithm in linear time. Let
$$y(t + \delta) = y(t) + \delta \cdot \mathbf{1}_{I(t)}.$$

4. Set $t := t + \delta$; if $t < 1$, go back to Step 2. Otherwise, return $y(1)$.

Lemma 3.6. *The fractional solution y found by the Smooth Greedy Algorithm is contained in the basis polytope $B(\mathcal{M})$ and*

$$F(y) \geq \left(1 - \frac{1}{e} - o(1)\right) \cdot OPT$$

Proof. In each time step, y increases by $\delta \cdot \mathbf{1}_I$ where I is a maximum-weight independent set, WLOG a basis of $\mathcal{M}$. The final fractional solution $y(1)$ is a sum of $1/\delta$ such vectors, hence it is contained in $B(\mathcal{M})$.

Our goal is to estimate how much $F(y)$ increases during one step of the algorithm. Consider a random set $R(t)$ corresponding to $\hat{y}(t)$, and an independently random set $D(t)$ that contains each item j independently with probability $\Delta_j(t) = y_j(t+\delta) - y_j(t)$. I.e., $\Delta(t) = y(t+\delta) - y(t) = \delta \cdot \mathbf{1}_{I(t)}$ and $D(t)$ is a random subset of $I(t)$ where each element appears independently with probability δ. It can be seen easily that $F(y(t+\delta)) = \mathbf{E}[f(R(t+\delta))] \geq \mathbf{E}[f(R(t) \cup D(t))]$. This follows from monotonicity, because $R(t+\delta)$ contains items independently with probabilities $y_j(t) + \Delta_j(t)$, while $R(t) \cup D(t)$ contains items independently with (smaller) probabilities $1 - (1 - y_j(t))(1 - \Delta_j(t))$.

Now we are ready to estimate how much $F(y)$ gains at time t. Again, it is important that the probability that any item appears in $D(t)$ is very small, so we can focus on the contributions from singleton sets $D(t)$. From the discussion above, we obtain

$$\begin{aligned}
F(y(t+\delta)) - F(y(t)) &\geq \mathbf{E}[f(R(t) \cup D(t)) - f(R(t))] \\
&\geq \sum_j \Pr[D(t) = \{j\}]\ \mathbf{E}[f_{R(t)}(j)] \\
&= \sum_{j \in I(t)} \delta(1-\delta)^{|I(t)|-1} \mathbf{E}[f_{R(t)}(j)] \\
&\geq \delta(1-\delta)^{n-1} \sum_{j \in I(t)} \mathbf{E}[f_{R(t)}(j)].
\end{aligned}$$

Recall that $I(t)$ is an independent set maximizing $\sum_{j \in I} \mathbf{E}[f_{R(t)}(j)]$. Hence,

$$\begin{aligned}
F(y(t+\delta)) - F(y(t)) &\geq \delta(1-\delta)^{n-1} \max_{I \in \mathcal{I}} \sum_{j \in I} \mathbf{E}[f_{R(t)}(j)] \\
&\geq \delta(1-\delta)^{n-1}(OPT - F(y(t)))
\end{aligned}$$

using the first inequality in Lemma 3.5. From here, $OPT - F(y(t+\delta)) \leq (1 - \delta(1-\delta)^{n-1})(OPT - F(y(t)))$ and by induction, $OPT - F(y(k\delta)) \leq (1 - \delta(1-\delta)^{n-1})^k\ OPT$. With our choice of $\delta = 1/n^2$ and $k = n^2$, we get

$$OPT - F(y(1)) \leq \left(1 - \frac{1}{n^2}\left(1 - \frac{1}{n^2}\right)^{n-1}\right)^{n^2} OPT \leq e^{-(1-1/n)} OPT.$$

Therefore, $F(y(1)) \geq (1 - 1/e - o(1))OPT$. □

Summary. This completes the proof of Theorem 3.1. First we find a point $y^* \in B(\mathcal{M})$ using the smooth greedy algorithm, which satisfies $F(y^*) \geq (1 - 1/e - o(1))OPT$. Then, we convert y^* into an integral solution $S \in \mathcal{I}$ such that $f(S) \geq F(y^*) - o(OPT)$. Both stages rely on random sampling estimates which succeed with high probability.

3.5 Extensions of submodular functions

In this section, we consider some alternative ways of extending a monotone submodular function $f : 2^N \to \mathbb{R}^+$ to a continuous function $\tilde{f} : [0,1]^N \to \mathbb{R}^+$. Our initial hope was to find an extension such that $\tilde{f}(y)$ could be optimized over $P(\mathcal{M})$ in polynomial time and the value of the solution could be compared approximately with $F(y)$. These considerations helped us gain a lot of insight into the problem, and they also led to the LP-based algorithm for weighted rank sums which appeared in [4].

Extension f^+: Our first candidate for $\tilde{f}$ is an extension similar to the objective function of the "Configuration LP" [22, 18, 19].

- $f^+(y) = \max\left\{\sum_{S \subseteq N} \alpha_S f(S) : \sum_S \alpha_S \leq 1, \alpha_S \geq 0 \ \& \ \forall j; \sum_{S:j\in S} \alpha_S \leq y_j\right\}.$

Extension f^*: Another candidate is a function appearing in [45] and subsequently [46, 56, 57], where it is used indirectly in the analysis of the greedy algorithm for submodular function maximization:

- $f^*(y) = \min\left\{f(S) + \sum_{j\in N} f_S(j) y_j : S \subseteq N\right\}.$

Unfortunately, as we show later (in Section 3.7), it is NP-hard to optimize $f^+(y)$ and $f^*(y)$ over matroid polytopes. Still, $f^+(y)$ and $f^*(y)$ provide interesting information about the problem.

It is known and easy to see that for $y \in \{0,1\}^N$, both f^+ and f^* functions coincide with f and thus they are indeed extensions of f. For any $y \in [0,1]^N$, we first show the following.

Lemma 3.7. *For any monotone submodular f, $F(y) \leq f^+(y) \leq f^*(y)$.*

Proof. To see the first inequality, let $\alpha_S = \prod_{i\in S} y_i \prod_{i\notin S}(1 - y_i)$ be the probability that we obtain $\hat{y} = \chi_S$ by independent rounding of y. Since $\sum_{S:j\in S} \alpha_S = \Pr[\hat{y}_j = 1] = y_j$, this is a feasible solution for $f^+(y)$ and therefore $f^+(y) \geq \sum_S \alpha_S f(S) = \mathbf{E}[f(\hat{y})] = F(y)$.

For the second inequality, consider any feasible vector α_S and any set $T \subseteq N$:

$$\sum_S \alpha_S f(S) \leq \sum_S \alpha_S \left(f(T) + \sum_{j \in S} f_T(j) \right) \leq f(T) + \sum_{j \in N} y_j f_T(j)$$

using submodularity and the properties of α_S. By taking the maximum on the left and the minimum on the right, we obtain $f^+(y) \leq f^*(y)$. □

It is tempting to conjecture that $f^+(y)$ and $f^*(y)$ are in fact equal, due to some duality relationship. However, this is not the case: both inequalities in Lemma 3.7 can be sharp and both gaps can be close to $1 - 1/e$. For the first inequality, consider the submodular function $f(S) = \min\{|S|, 1\}$ and $y_j = 1/n$ for all j; then $F(y) = 1 - (1 - 1/n)^n$ and $f^+(y) = 1$. For the second inequality, choose a large but fixed k, $f(S) = 1 - (1 - |S|/n)^k$ and $y_j = 1/k$ for all j. The reader can verify that $f^+(y) = 1 - (1 - 1/k)^k$, while $f^*(y) \geq 1 - k/n \to 1$ as $n \to \infty$. We prove that $1 - 1/e$ is the worst possible gap for both inequalities. Moreover, even the gap between $F(y)$ and $f^*(y)$ is bounded by $1 - 1/e$.

Lemma 3.8. *For any monotone submodular f, $F(y) \geq \left(1 - \frac{1}{e}\right) f^*(y)$.*

Proof. For each element $j \in N$, set up an independent Poisson clock $\mathcal{C}_j$ of rate y_j, i.e. a device which sends signals at random times, in any infinitesimal time interval of size dt independently with probability $y_j dt$. We define a random process which starts with an empty set $S(0) = \emptyset$ at time $t = 0$. At any time when the clock $\mathcal{C}_j$ sends a signal, we include element j in S, which increases its value by $f_S(j)$. (If j is already in S, nothing happens; the marginal value $f_S(j)$ is zero in this case.) Denote by $S(t)$ the random set we have at time t. By the definition of a Poisson clock, $S(1)$ contains element j independently with probability $1 - e^{-y_j} \leq y_j$. Since such a set can be obtained as a subset of the random set defined by $\hat{y}$, we have $\mathbf{E}[f(S(1))] \leq F(y)$ by monotonicity. We show that $\mathbf{E}[f(S(1))] \geq (1 - 1/e) f^*(y)$ which will prove the claim.

Let $t \in [0, 1]$. Condition on $S(t) = S$ and consider how $f(S(t))$ changes in an infinitesimal interval $[t, t + dt]$. The probability that we include element j is $y_j dt$. Since dt is very small, the events for different elements j are effectively disjoint. Thus the expected increase of $f(S(t))$ is (up to $O(dt^2)$ terms)

$$\mathbf{E}[f(S(t + dt)) - f(S(t)) \mid S(t) = S] = \sum_{j \in N} f_S(j) y_j dt \geq (f^*(y) - f(S)) dt$$

using the definition of $f^*(y)$. We divide by dt and take the expectation over S:

$$\frac{1}{dt} \mathbf{E}[f(S(t + dt)) - f(S(t))] \geq f^*(y) - \mathbf{E}[f(S(t))].$$

We define $\phi(t) = \mathbf{E}[f(S(t))]$, i.e. $\frac{d\phi}{dt} \geq f^*(y) - \phi(t)$. We solve this differential inequality by considering $\psi(t) = e^t\phi(t)$ and $\frac{d\psi}{dt} = e^t(\frac{d\phi}{dt} + \phi(t)) \geq e^t f^*(y)$. Since $\psi(0) = \phi(0) = 0$, this implies

$$\psi(x) = \int_0^x \frac{d\psi}{dt} dt \geq \int_0^x e^t f^*(y) dt = (e^x - 1) f^*(y)$$

for any $x \geq 0$. We conclude that $\mathbf{E}[f(S(t))] = \phi(t) = e^{-t}\psi(t) \geq (1-e^{-t})f^*(y)$ and $F(y) \geq \mathbf{E}[f(S(1))] \geq (1 - 1/e)f^*(y)$. $\square$

We remark that we did not actually use submodularity in the proof of Lemma 3.8! Formally, it can be stated for all monotone functions f. However, $f^*(y)$ is not a proper extension of f when f is not submodular (e.g., $f^*(y)$ is identically zero if $f(S) = 0$ for $|S| \leq 1$). So the statement of Lemma 3.8 is not very meaningful in this generality.

To summarize what we have proved so far, we have two relaxations of our problem:

- $\max\{f^+(y) : y \in P(\mathcal{M})\}$
- $\max\{f^*(y) : y \in P(\mathcal{M})\}$

Our framework together with Lemma 3.7 and Lemma 3.8 implies that both of these relaxations have integrality gap at most $1-1/e$ (and this is tight, see Section 3.8). Unfortunately, there exist submodular functions f for which it is NP-hard to solve either of these relaxations. (See Theorem 3.10) We show how to use the framework efficiently in a restricted case of interest which is described in the following section.

3.6 Algorithm for weighted rank sums

We can use the extension $f^+(y)$ to achieve an LP-based $(1-1/e)$-approximation for any submodular function f that can be expressed as a weighted rank sum. Here we describe this class of functions in detail.

Weighted rank functions of matroids: Given a matroid $(N, \mathcal{X})$ and a weight function $w : N \to \mathbb{R}^+$, we define a *weighted rank function* $g : 2^N \to \mathbb{R}^+$,

$$g(S) = \max\{\sum_{j \in I} w_j : I \subseteq S \ \&\ I \in \mathcal{X}\}.$$

It is well known that such a function is monotone and submodular. A simple special case is when $\mathcal{X} = \{I \mid |I| = 1\}$. Then $g(S)$ returns simply the maximum-weight element of S; this will be useful in our application to GAP.

Weighted rank sums: We consider functions $f : 2^N \to \mathbb{R}^+$ of the form $f(S) = \sum_{i=1}^m g_i(S)$ where each g_i is a weighted rank function for matroid $(N, \mathcal{X}_i)$ with weights w_{ij}. Again, $f(S)$ is monotone and submodular.

The functions that can be generated in this way form a fairly rich subclass of monotone submodular functions. In particular, they generalize submodular functions arising from coverage systems. Coverage-type submodular functions can be obtained by considering a simple uniform matroid $(N, \mathcal{X})$ with $\mathcal{X} = \{I \subseteq N \mid |I| \leq 1\}$. For a collection of sets $\{A_j\}_{j\in N}$ on a ground set $[m]$, we can define m collections of weights on N, where $w_{ij} = 1$ if A_j contains element i, and 0 otherwise. Then the weighted rank function $g_i(S) = \max\{w_{ij} : j \in S\}$ is simply an indicator of whether $\bigcup_{j\in S} A_j$ covers element i. The sum of the rank functions $g_i(S)$ gives exactly the size of this union $f(S) = \sum_{i=1}^m g_i(S) = \left|\bigcup_{j\in S} A_j\right|$. Generalization to the weighted case is straightforward.

LP formulation for weighted rank sums: For a submodular function given as $f(S) = \sum_{i=1}^m g_i(S)$ where $g_i(S) = \max\{w_i(I) : I \subseteq S, I \in \mathcal{X}_i\}$, consider an extension $g_i^+(y)$ for each g_i, as defined in Section 3.5:

$$g_i^+(y) = \max\{\sum_{S\subseteq N} \alpha_S g_i(S) : \sum_S \alpha_S \leq 1, \alpha_S \geq 0 \ \&\ \forall j; \sum_{S:j\in S} \alpha_S \leq y_j\}.$$

Here, we can assume without loss of generality that α_S is nonzero only for $S \in \mathcal{X}_i$ (otherwise replace each S by a subset $I \subseteq S, I \in \mathcal{X}_i$, such that $g_i(S) = w_i(I)$). Therefore, g_i^+ can be written as

$$g_i^+(y) = \max\{\sum_{I\in\mathcal{X}_i} \alpha_I \sum_{j\in I} w_{ij} : \sum_{I\in\mathcal{X}_i} \alpha_I \leq 1, \alpha_I \geq 0 \ \&\ \forall j; \sum_{I\in\mathcal{X}_i: j\in I} \alpha_I \leq y_j\}.$$

We can set $x_{ij} = \sum_{I\in\mathcal{X}_i:j\in I} \alpha_I$ and observe that a vector $x_i = (x_{ij})_{j\in N}$ can be obtained in this way if and only if it is a convex linear combination of independent sets; i.e., if it is in the matroid polytope $P(\mathcal{X}_i)$. The objective function becomes $\sum_{j\in N} w_{ij} \sum_{I\in\mathcal{X}_i:j\in I} \alpha_I = \sum_{j\in N} w_{ij} x_{ij}$ and so we can write equivalenly:

$$g_i^+(y) = \max\{\sum_{j\in N} w_{ij} x_{ij} : x_i \in P(\mathcal{X}_i) \ \&\ \forall j; x_{ij} \leq y_j\}.$$

We sum up these functions to obtain an extension of f, $\tilde{f}(y) = \sum_{i=1}^m g_i^+(y)$. This leads to the following LP formulation for the problem $\max\{\tilde{f}(y) : y \in$

$P(\mathcal{M})\}$:

$$\begin{aligned} \max \sum_{i=1}^{m} \sum_{j \in N} w_{ij} x_{ij}; \\ \forall i, j; x_{ij} \leq y_j, \\ \forall i; x_i \in P(\mathcal{X}_i), \\ y \in P(\mathcal{M}). \end{aligned}$$

We can solve the LP using the ellipsoid method, since a separation oracle can be efficiently implemented for each matroid polytope, and therefore also for this LP. To obtain a $(1-1/e)$-approximation via the above LP using the pipage rounding framework from Section 3.3, it is sufficient to prove the following lemma.

Lemma 3.9. *For any weighted rank sum f, $F(y) \geq (1-1/e)\tilde{f}(y)$.*

Proof. By Lemma 3.8, $F(y) \geq (1-1/e)f^*(y)$ and hence it suffices to prove that $f^*(y) \geq \tilde{f}(y)$. By Lemma 3.7, $g_i^+(y) \leq g_i^*(y)$ where $g_i^*(y) = \min_{S_i}(g_i(S_i) + \sum_j y_j g_{i,S_i}(j))$. (Here, $g_{i,S_i}(j) = g_i(S_i + j) - g_i(S_i)$.) Consequently,

$$\begin{aligned} \tilde{f}(y) &= \sum_{i=1}^{m} g_i^+(y) \leq \sum_{i=1}^{m} \min_{S_i}(g_i(S_i) + \sum_{j \in N} y_j g_{i,S_i}(j)) \\ &\leq \min_S \sum_{i=1}^{m} (g_i(S) + \sum_{j \in N} y_j g_{i,S}(j)) = \min_S (f(S) + \sum_{j \in N} y_j f_S(j)) = f^*(y). \end{aligned}$$

□

3.7 NP-hardness results for our relaxations

In this section, we show negative results concerning the possibility using a linear programming approach based on the relaxations $\max\{f^+(y) : y \in P(\mathcal{M})\}$ or $\max\{f^*(y) : y \in P(\mathcal{M})\}$.

Theorem 3.10. *There is $\delta > 0$ such that for a given matroid $\mathcal{M}$ it is NP-hard to find any point $z \in P(\mathcal{M})$ such that $f^+(z) \geq (1-\delta)\max\{f^+(y) : y \in P(\mathcal{M})\}$. Similarly, it is NP-hard to find any point $z \in P(\mathcal{M})$ such that $f^*(z) \geq (1-\delta)\max\{f^*(y) : y \in P(\mathcal{M})\}$. These results hold even for coverage-type submodular functions and partition matroids.*

We present the proof in two parts.

3.7.1 Optimizing $f^+(y)$

Proposition 3.11. *There is $\delta > 0$ such that for a given monotone submodular function f and matroid $\mathcal{M}$, it is NP-hard to find any point $y^* \in P(\mathcal{M})$ such that*

$$f^+(y^*) \geq (1-\delta)\max\{f^+(y) : y \in P(\mathcal{M})\}.$$

Proof. We use the following hardness result for the max-k-cover problem [17, 19]:

For any $\epsilon > 0$, given a collection of sets $\{S_i \subseteq U \mid i \in N\}$ where $|S_i| = s$ and $|U| = ks$, it is NP-hard to distinguish the case where U can be covered by k disjoint sets S_i and the case where any collection of k sets covers at most $(1-1/e+\epsilon)|U|$ elements.

Given such a collection of sets, we define a coverage function $f(A) = |\bigcup_{i \in A} S_i|$. A natural matroid here is the uniform matroid $\mathcal{M} = \{I \subseteq N \mid |I| \leq k\}$, i.e. $P(\mathcal{M}) = \{y \in [0,1]^N \mid \sum y_i \leq k\}$.

Note that if there are k sets indexed by $A, |A| = k$, such that $\bigcup_{i\in A} S_i = U$, then $\mathbf{1}_A \in P(\mathcal{M})$ and $f^+(\mathbf{1}_A) = f(A) = |U|$. On the other hand, if there are no such k sets, then $\max\{f^+(y) : y \in P(\mathcal{M})\} < |U|$, because $f^+(y)$ for any $y \in P(\mathcal{M})$ is a convex linear combination $\sum \alpha_A f(A)$ where $\sum \alpha_A |A| \leq k$, and therefore all sets in the linear combination cannot satisfy $f(A) = |U|$. This proves that it is NP-hard to compute $\max\{f^+(y) : y \in P(\mathcal{M})\}$. However, we want to prove NP-hardness even for the problem of finding a (near-)optimal point y, without computing its value.

We extend the set system by k additional sets $T_1, \ldots, T_k$, each containing $(1-\frac{1}{4e})s$ elements disjoint from everything else. The index set becomes $N' = N \cup K$, $|K| = k$. Our goal is to show that in case of a YES instance (k sets covering U), the optimum is attained for a vector y supported by N. In case of a NO instance, it is more profitable to use the sets $T_1, \ldots, T_k$ and therefore the optimal point is $y = \mathbf{1}_K$. By careful analysis, we get hardness even for finding an approximately optimal point.

First, consider a YES instance, where $f(A) = |U| = ks$ for $A \subseteq N, |A| = k$. Then $f^+(\mathbf{1}_A) = ks$ is clearly an optimum over $P(\mathcal{M})$, because $f(B) \leq |B|s$ for any $B \subseteq N'$ and hence $f^+(y) \leq \sum_{i \in N'} y_i s \leq ks$ for $y \in P(\mathcal{M})$. Moreover, any vector y such that $\sum_{i \in K} y_i > 4e\delta k$ has $f^+(y) < (1-\delta)ks$, because each element $i \in K$ can contribute only $(1-\frac{1}{4e})s$ rather than s. Therefore, any $(1-\delta)$-approximate optimum satisfies $\sum_{i\in K} y_i \leq 4e\delta k$.

Next, consider a NO instance, i.e. for any $A \subseteq N, |A| \leq k$, we have $f(A) \leq (1-1/e+\epsilon)ks$. By submodularity, we get $f(A) \leq (1-1/e+\epsilon)|A|s$ for $|A| > k$, since otherwise we could find a subset $A' \subseteq A$ of size k and value $f(A') > (1-1/e+\epsilon)ks$. We also have $f(A) \leq |A|s$ for any $|A| < (1-1/e+\epsilon)k$.

By combining these 3 bounds in the ranges of $|A|$ between $[(1-1/e+\epsilon)k, k]$, above k and below $(1-1/e+\epsilon)k$, we obtain

$$f(A) \leq (1-1/e+\epsilon)(|A| + (1/e-\epsilon)k)s$$

for all $A \subseteq N$. Since this bound is linear in $|A|$, it implies a similar bound for $f^+(y), y \in [0,1]^N$: Assume $\sum_{i\in N} y_i = a$. Then $f^+(y)$ is a convex linear combination $\sum_{A\subseteq N} \alpha_A f(A)$ where $\sum_{A\subseteq N} \alpha_A |A| = a$ and $\sum_{A\subseteq N} \alpha_A = 1$. Consequently,

$$f^+(y) \leq (1-1/e+\epsilon)(a + (1/e-\epsilon)k)s.$$

Now assume $y \in [0,1]^{N\cup K}$ where $\sum_{i\in N} y_i = a$ and $\sum_{i\in K} y_i = k-a$. Since each element of K contributes $(1-\frac{1}{4e})s$, independently of other elements, we have

$$\begin{aligned} f^+(y) &\leq (1-\frac{1}{e}+\epsilon)(a+(\frac{1}{e}-\epsilon)k)s + (k-a)(1-\frac{1}{4e})s \\ &\leq (1+\frac{3}{4e}-\frac{1}{e^2})ks - (\frac{3}{4e}-\epsilon)as. \end{aligned}$$

If $a \geq (1-4e\delta)k$, we get $f^+(y) \leq (1-\frac{1}{e^2}+O(\epsilon+\delta))ks$, while $f^+(\mathbf{1}_K) = (1-\frac{1}{4e})ks$ is a substantially larger value. Therefore, any near-optimal point $y \in P(\mathcal{M})$ must satisfy $\sum_{i\in K} y_i = k-a > 4e\delta k$.

If we could find any $(1-\delta)$-approximate optimum $y \in P(\mathcal{M})$, we could test whether $\sum_{i\in K} y_i$ is above or below $4e\delta k$ and hence distinguish YES and NO instances. □

3.7.2 Computing $f^*(y)$

In this section, we show that it is NP-hard to compute $f^*(y)$.

Proposition 3.12. *Let $G=(V,E)$ be a graph. For $S \subseteq V$, let $c(S)$ denote the number of edges incident with S, and let $f(S) = |S| + c(S)$. Then there is $\delta > 0$ such that it is NP-hard to $(1-\delta)$-approximate $f^*(y)$ for a given graph G and $y = (3/4,\ldots,3/4)$.*

Note that $c(S)$ is a coverage-type submodular function and hence $f(S)$ also, by adding one distinct element to each set in the system.

Proof. Let's analyze the extension $f^*(y) = \min_S f(S) + \sum_j y_j f_S(j)$ for this particular submodular function. Here, $f_S(j) = 1 + c_S(j)$ where $c_S(j)$ is the number of edges incident with j but not with S. For $y = (3/4, \dots, 3/4)$, we get

$$f^*(y) = \min_S |S| + c(S) + \frac{3}{4} \sum_{j \in V \setminus S} (1 + c_S(j)) = \min_S |S| + c(S) + \frac{3}{4}|V \setminus S| + \frac{3}{2}|E(\bar{S})|.$$

Here, $E(\bar{S})$ denotes the edges disjoint from S, which get a contribution of 3/4 from each endpoint. Using $c(S) + |E(\bar{S})| = |E|$ and joining the terms, we get

$$f^*(y) = \min_S \frac{3}{4}|V| + \frac{3}{2}|E| + \frac{1}{4}|S| - \frac{1}{2}c(S).$$

Note that if S doesn't cover all edges, we can decrease this expression by including another vertex in S, covering a new edge. The minimum can be attained only when all edges are covered by S and $c(S) = |E|$. Therefore, $f^*(y) = \frac{3}{4}|V| + |E| + \frac{1}{4}k^*$ where k^* is the minimum vertex cover of G. Since this number is NP-hard to compute, so is $f^*(y)$. In fact, vertex cover is APX-hard, even for instances where $|E| = O(|V|)$, and hence $f^*(y)$ is also APX-hard. □

3.7.3 Optimizing $f^*(y)$

Proposition 3.13. *There is $\delta > 0$ such that for a given monotone submodular function f and matroid $\mathcal{M}$, it is NP-hard to find any point $y^* \in P(\mathcal{M})$ such that*

$$f^*(y^*) \geq (1 - \delta) \max\{f^*(y) : y \in P(\mathcal{M})\}.$$

Proof. We use the following hardness result on Vertex Cover in r-uniform hypergraphs [12]:

For any $\epsilon > 0$ and $r > 2/\epsilon$ constant, it is NP-hard for a given r-uniform hypergraph H on n vertices to distinguish whether the minimum vertex cover in H has size at most ϵn or at least $(1 - \epsilon)n$.

Here, the reduction is somewhat less intuitive. We denote the vertices of H by V, edges by E, and we define our ground set to be $N = V' \cup V''$, a union of two disjoint copies of V. In other words, we produce two clones j', j'' for each vertex $j \in V$. Our submodular function is the following:

$$f(S) = |S \cap V'| + 2|S \cap V''| + c(S)$$

where $c(S)$ is the number of edges $e \in E$ such that S contains some clone of a vertex in e. In other words, $c(S)$ is the vertex-cover function for a derived hypergraph where each vertex is replaced by two clones.

We prove that it is NP-hard to optimize $f^*(y)$ over $P(\mathcal{M})$ for the following matroid:

$$\mathcal{M} = \{I \subseteq V' \cup V'' \mid \forall j \in V; |I \cap \{j', j''\}| \leq 1\},$$

i.e. we allow at most one clone of each vertex to be chosen. The matroid polytope $P(\mathcal{M})$ is defined by the constraints $y_{j'} + y_{j''} \leq 1$ for each vertex j. Since the maximum of a monotone function over $P(\mathcal{M})$ must be attained on the basis face of $P(\mathcal{M})$, we can assume that $y_{j'} + y_{j''} = 1$ for any candidate for an optimum.

Observe that maximizing $f(S)$ over independent sets $S \in \mathcal{M}$ yields $f(V'') = 2n + |E|$, which does not depend on the structure of H in any way. However, $f^*(y)$ for fractional vectors $y \in P(\mathcal{M})$ will depend on the minimum vertex cover of H in a way similar to the previous section. Then the value can be possibly larger than $2n + |E|$. The intuition here is that if the minimum vertex cover is very small, the gain for a fractional vector is not significant and it is more profitable to take $y = \mathbf{1}_{V''}$ where $f^*(y) = 2n + |E|$. On the other hand, if the minimum vertex cover is very large, the gain for a fractional solution outweighs the loss incurred by the fact that elements of V' contribute 1 rather than 2 to $f(S)$. Then we can achieve a value close to $\frac{7}{3}n + |E|$, for $y_{j'} = 1/3, y_{j''} = 2/3$. (The reader can verify that this is the exact optimum when the minimum vertex cover is $|V| = n$.) Note that the optimal point does not necessarily indicate what the minimum vertex cover is, but testing how close the optimal point is to being integral can help us decide about the size of the minimum vertex cover.

Let's write $f^*(y)$ in the following form:

$$\begin{aligned} f^*(y) &= \min_S f(S) + \sum_{i \in N} y_i f_S(i) \\ &= \min_S |S \cap V'| + 2|S \cap V''| + c(S) + y(V' \setminus S) + 2y(V'' \setminus S) + \sum_{e: e \cap S = \emptyset} y(e). \end{aligned}$$

Again, we want to argue that the minimum is attained for a vertex cover. We assume that $y_{j'} + y_{j''} = 1$ for each pair of clones, i.e. we get $y(e) = r > 4$ for each edge. Therefore, covering a new edge by including a new vertex in S can only decrease the subject of minimization and

$$f^*(y) = \min_{\text{vertex cover } S} |S \cap V'| + 2|S \cap V''| + y(V' \setminus S) + 2y(V'' \setminus S) + |E|.$$

Consider a YES instance, where H has a vertex cover of size ϵn. We can set $y_{j'} = 0, y_{j''} = 1$ for each $j \in V$ and we get $f^*(y) = f(V'') = 2n + |E|$. We do not claim that this is exactly the optimum but we claim that any approximately optimal point must be close to $\mathbf{1}_{V''}$. Suppose that for at least

$n/2$ vertices, we have $y_{j'} \geq 1/6$ and $y_{j''} \leq 5/6$. Then choosing $S \subseteq V'$ to be a vertex cover of size at most ϵn shows that

$$\begin{aligned} f^*(y) &= |S \cap V'| + 2|S \cap V''| + y(V' \setminus S) + 2y(V'' \setminus S) + |E| \\ &\leq \epsilon n + y(V') + 2y(V'') + |E| \leq \epsilon n + \frac{n}{2}\left(\frac{1}{6} + 2 \cdot \frac{5}{6}\right) + \frac{n}{2}(0 + 2 \cdot 1) + |E| \\ &\leq \left(2 - \frac{1}{12} + \epsilon\right) n + |E|. \end{aligned}$$

For sufficiently small ϵ, this is smaller than the optimum. Thus for any near-optimal point $y \in P(\mathcal{M})$, more than $n/2$ vertices must satisfy $y_{j'} < 1/6, y_{j''} > 5/6$.

On the other hand, consider a NO instance, where any vertex cover has size at least $(1-\epsilon)n$. Here, let's choose $y^*_{j'} = 1/3, y^*_{j''} = 2/3$ for each $j \in V$. This gives

$$f^*(y^*) = |S \cap V'| + 2|S \cap V''| + \frac{1}{3}|V' \setminus S| + \frac{4}{3}|V'' \setminus S| + |E| = \frac{2}{3}|S| + \frac{1}{3}|V'| + \frac{4}{3}|V''| + |E|$$

for some vertex cover S. Since $|S| \geq (1-\epsilon)n$, we have

$$f^*(y^*) \geq \frac{2}{3}(1-\epsilon)n + \frac{1}{3}n + \frac{4}{3}n + |E| = \left(\frac{7}{3} - \frac{2}{3}\epsilon\right) n + |E|.$$

We claim that any near-optimal point y must be close y^*. Suppose on the contrary that there are at least $n/2$ vertices for which $y_{j'} \leq 1/6$ and $y_{j''} \geq 5/6$. Let

$$S = \{j' \in V' \mid y_{j'} \geq 1/3\} \cup \{j'' \in V'' \mid y_{j'} < 1/3\}.$$

This is a vertex cover, containing exactly one clone of each vertex. Note that we have $y_{j'} < 1/3$ for $j' \in V' \setminus S$, and at least $n/2$ of these coordinates in fact satisfy $y_{j'} \leq 1/6$. For $j'' \in V'' \setminus S$, we have $y_{j''} \leq 2/3$. Therefore

$$\begin{aligned} f^*(y) &\leq |S \cap V'| + 2|S \cap V''| + y(V' \setminus S) + 2y(V'' \setminus S) + |E| \\ &\leq |S \cap V'| + 2|S \cap V''| + \left(\frac{1}{3}|V' \setminus S| - \frac{n}{2} \cdot \frac{1}{6}\right) + \frac{4}{3}|V'' \setminus S| + |E| \\ &= |S| + |V''| + \frac{1}{3}|(V' \cup V'') \setminus S| - \frac{1}{12}n + |E| = \frac{7}{3}n - \frac{1}{12}n + |E| \end{aligned}$$

which is smaller than $f^*(y^*)$. Therefore, for any near-optimal vector, more than $n/2$ vertices must satisfy $y_{j'} > 1/6, y_{j''} < 5/6$. If we could find such a near-optimal vector y, we could distinguish YES and NO instances by testing how many vertices satisfy this condition. We note that the hard instances of H in [12] have constant (albeit large) degrees, therefore $|E| = O(n)$ and the hardness gap is a constant factor. $\square$

3.8 Integrality gaps of our relaxations

In this section, we show that the integrality gap for both of our relaxations, $\max\{f^+(y) : y \in P(\mathcal{M})\}$ and $\max\{f^*(y) : y \in P(\mathcal{M})\}$ can be arbitrarily close to $1 - 1/e$, even for coverage-type submodular functions. This might seem unsurprising, since we have a $(1 - 1/e + \epsilon)$-inapproximability result for the discrete optimization problem already. However, let's not forget that these relaxations themselves are NP-hard to solve in the value oracle model. Thus it is plausible that by having some means to solve them (e.g., a demand oracle), we might achieve an approximation better than $1 - 1/e$. But this is not the case.

Proposition 3.14. *The relaxation* $\max\{f^+(y) : y \in P(\mathcal{M})\}$ *can have an integrality gap arbitrarily close to* $1-1/e$*, even for a coverage-type submodular function and a partition matroid.*

Since we know $f^+(y) \le f^*(y)$, we get at least the same gap for $\max\{f^*(y) : y \in P(\mathcal{M})\}$ as well.

Proof. Let $X = X_1 \cup \ldots \cup X_n$ and let $\mathcal{M} = (X, \mathcal{I})$ be a partition matroid where $I \in \mathcal{I}$ iff $|I \cap X_i| \le 1$ for each i. We also define a submodular function $f : X \to \mathbb{R}$ in the following way: Let T be a random set containing independently one random element from each X_i, and let

$$f(S) = \Pr_T[S \cap T \ne \emptyset].$$

This is a coverage-type submodular function, since probability is measure on a probability space and the event $[S \cap T \ne \emptyset]$ can be written as a union of events $\bigcup_{j \in S}[j \in T]$.

For any independent set $I \in \mathcal{I}$, we have

$$f(I) = \Pr_T[I \cap T \ne \emptyset] \le 1 - (1 - 1/n)^n$$

where equality is achieved when I contains exactly one element from each X_i.

On the other hand, consider a fractional vector $y \in P(\mathcal{M})$ where each element $j \in X$ gets value $y_j = 1/n$. This is a convex combination of independent sets, hence a feasible vector in $P(\mathcal{M})$. What is the value of $f^+(y)$? By the definition of $f^+(y)$,

$$f^+(y) = \max\left\{ \sum_{S \subseteq X} \alpha_S f(S) : \sum_S \alpha_S \le 1, \alpha_S \ge 0 \;\&\; \forall j; \sum_{S : j \in S} \alpha_S \le y_j \right\}.$$

We choose $\alpha_{X_i} = 1/n$ for each $i = 1, 2, \ldots, n$. Then $\sum_{S:j \in S} \alpha_S = 1/n$ because each element appears in exactly one set X_i. Also,

$$f(X_i) = \Pr_T[T \cap X_i] = 1$$

since T always contains some element of X_i. Hence, $f^+(y) \geq 1$ and the integrality gap is at least $1 - (1 - 1/n)^n$. $\square$

Chapter 4

The Submodular Welfare Problem

Problem: *Given n players with utility functions $w_i : 2^X \to \mathbb{R}_+$ which are assumed to be monotone and submodular, find a partition $X = S_1 \cup S_2 \cup \ldots \cup S_n$ in order to maximize $\sum_{i=1}^{n} w_i(S_i)$.*

This problem is sometimes referred to as *combinatorial auctions*, in which case the players are referred to as *bidders*. However, in this thesis we do not consider issues of auction design such as players not being necessarily truthful. We regard this as an optimization problem where we have access to the true utility functions that the players hold. We discussed in Section 1.1 how this access can be implemented. We consider two models: (a) the value oracle model, and (b) the demand oracle model. We leave the (b) option to the next chapter, and focus here on the value oracle model.

We already mentioned in Section 1.2.3 that the Submodular Welfare Problem is in fact a special case of submodular maximization under a matroid constraint. We review the reduction here.

Reduction from Submodular Welfare to Submodular Maximization over a Matroid. Let the set of players be P, the set of items Q, and for each $i \in P$, let the respective utility function be $w_i : 2^Q \to \mathbb{R}_+$. We define a new ground set $X = P \times Q$, with a function $f : 2^X \to \mathbb{R}_+$ defined as follows: Every set $S \subseteq X$ can be written uniquely as $S = \bigcup_{i \in P}(\{i\} \times S_i)$. Then let

$$f(S) = \sum_{i \in P} w_i(S_i).$$

The interpretation of $P \times Q$ is that we make $|P|$ copies of each item, one for each player. The function $f(S)$ represents the total utility of all players after allocating the items in S to the respective players (assuming that we actually have multiple copies of each item for different players).

However, in reality we can only allocate one copy of each item. Therefore, let us define a partition matroid $\mathcal{M} = (X, \mathcal{I})$ as follows:

$$\mathcal{I} = \{S \subseteq X \mid \forall j; |S \cap (P \times \{j\})| \leq 1\}.$$

Then, it is easy to see that the Submodular Welfare Problem is equivalent to the problem $\max_{S \in \mathcal{I}} f(S)$.

Due to our results on submodular maximization under a matroid constraint (see Chapter 3), we obtain immediately the following.

Theorem 4.1. *There is a randomized polynomial-time* $(1-1/e)$*-approximation to the Submodular Welfare Problem in the value oracle model.*

In addition, we include the following observations.

- The factor of $1 - 1/e$, or in fact $(1 - (1 - 1/n)^n$ for n players, can be achieved in expectation without *any* queries when the n players all have the same submodular utility function.

- We give a $(1-1/e)$-approximation algorithm for n players (with possibly different utility functions) which is simpler than the algorithm for a general matroid constraint. In particular, we do not need the pipage rounding technique.

Due to [36], $(1-1/e)$-approximation for Submodular Welfare in the value oracle model is optimal unless $P = NP$. We provide unconditional evidence that the $1 - 1/e$ ratio cannot be improved in the value oracle model.

- Using polynomially many value queries, a factor better than $1 - (1 - 1/n)^n$ for n players is impossible to achieve (regardless of $P = NP$). This holds even when all players have the same utility function.

- The same result holds for maximizing a monotone submodular function subject to a cardinality constraint; i.e. a better than $(1 - 1/e)$-approximation for $\max\{f(S) : |S| \leq k\}$ is impossible with polynomially many value queries.

4.1 A sampling lemma on submodular functions

We prove a lemma about submodular function which will be useful in the following. Roughly speaking, the lemma says that by combining random subsets of given sets, we recover a good fraction of their values.

Lemma 4.2. *Let $w : 2^X \to \mathbb{R}_+$ be a monotone submodular function and $A_1(p_1)$, …, $A_n(p_n)$ random sets, where each item of A_i appears in $A_i(p_i)$ independently with probability p_i. If $\sum_{i=1}^n p_i \leq 1$, then*

$$\mathbf{E}[w(A_1(p_1) \cup A_2(p_2) \cup \ldots \cup A_n(p_n))] \geq \left(1 - \left(1 - \frac{1}{n}\right)^n\right) \sum_{i=1}^n p_i w_i(A_i).$$

We note that, although this is not important for our applications, the sets $A_1, \ldots, A_n$ need not be necessarily disjoint, and also the elements of each A_i need not be sampled independently. The crucial assumption is that the subsets $A_i(p_i)$ are sampled independently for different values of i.

First, we prove the following inequality, which is a generalization of Lemma 2.3.

Lemma 4.3. *Let $w : 2^X \to \mathbb{R}$ be submodular and $A_1, A_2, \ldots, A_n \subseteq X$. For each i independently, sample a random subset $A_i(p_i)$ which contains each element of A_i with probability p_i. Then*

$$\mathbf{E}[w(A_1(p_1) \cup \ldots \cup A_n(p_n))] \geq \sum_{I \subseteq [n]} \prod_{i \in I} p_i \prod_{i \notin I} (1 - p_i)\ w\left(\bigcup_{j \in I} A_j\right).$$

Proof. We proceed by induction on n. Let's condition on $A_1(p_1) = A'_1$, …, $A_{n-1}(p_{n-1}) = A'_{n-1}$. By Lemma 2.2 applied to $g(S) = w(A'_1 \cup \ldots \cup A'_{n-1} \cup S)$, we have

$$\begin{aligned}\mathbf{E}[w(A'_1 \cup A'_2 \cup \ldots \cup A'_{n-1} \cup A_n(p_n))] \ &\geq\ p_n w(A'_1 \cup A'_2 \cup \ldots \cup A'_{n-1} \cup A_n) \\ &\quad + (1 - p_n) w(A'_1 \cup A'_2 \cup \ldots \cup A'_{n-1}).\end{aligned}$$

Plugging in $A'_i = A_i(p_i)$ and taking the expectation, we get

$$\begin{aligned}\mathbf{E}[w(A_1(p_1) \cup \ldots \cup A_n(p_n))] \ &\geq\ p_n \mathbf{E}[w(A_1(p_1) \cup \ldots \cup A_{n-1}(p_n) \cup A_n)] \\ &\quad + (1 - p_n) \mathbf{E}[w(A_1(p_1) \cup \ldots \cup A_{n-1}(p_{n-1}))].\end{aligned}$$

Now we can use the inductive hypothesis for $n - 1$ sets:

$$\mathbf{E}[w(A_1(p_1) \cup \ldots \cup A_{n-1}(p_{n-1}))] \geq \sum_{I \subseteq [n-1]} \prod_{i \in I} p_i \prod_{i \in [n-1] \setminus I} (1 - p_i)\ w\left(\bigcup_{j \in I} A_j\right)$$

and the same for the submodular function $h(S) = w(S \cup A_n)$:

$$\mathbf{E}[w(A_1(p_1) \cup \ldots A_{n-1}(p_{n-1}) \cup A_n)] \geq \sum_{I \subseteq [n-1]} \prod_{i \in I} p_i \prod_{i \in [n-1] \setminus I} (1-p_i)\, w \left(\bigcup_{j \in I \cup \{n\}} A_j \right),$$

We multiply the first inequality by $1 - p_n$ and the second one by p_n, which yields

$$\begin{aligned}
\mathbf{E}[w(A_1(p_1) \cup \ldots A_n(p_n))] &\geq (1-p_n) \sum_{I \subseteq [n-1]} \prod_{i \in I} p_i \prod_{i \in [n-1] \setminus I} (1-p_i)\, w \left(\bigcup_{j \in I} A_j \right) \\
&+ p_n \sum_{I \subseteq [n-1]} \prod_{i \in I} p_i \prod_{i \in [n-1] \setminus I} (1-p_i)\, w \left(\bigcup_{j \in I \cup \{n\}} A_j \right) \\
&= \sum_{I \subseteq [n]} \prod_{i \in I} p_i \prod_{i \notin I} (1-p_i)\, w \left(\bigcup_{j \in I} A_j \right).
\end{aligned}$$

□

Now we are ready to prove Lemma 4.2.

Proof. Using Lemma 4.3, we only need to estimate $\sum_{I \subseteq [n]} \prod_{i \in I} p_i \prod_{i \notin I} (1-p_i)$ $w\left(\bigcup_{j \in I} A_j\right)$. Let's assume that $w(A_1) \geq w(A_2) \geq \ldots \geq w(A_n)$. We define

$$\mathcal{I}_m = \{ I \subseteq [n] : \min(I) = m \},$$

the collection of subsets I whose minimum element is m. For any $I \in \mathcal{I}_m$, we can use monotonicity to replace $\bigcup_{j \in I} A_j$ by A_m. Then we get

$$\begin{aligned}
\sum_{I \subseteq [n]} \prod_{i \in I} p_i \prod_{i \notin I} (1-p_i)\, w \left(\bigcup_{j \in I} A_j \right) &\geq \sum_{m=1}^{n} w(A_m) \sum_{I \in \mathcal{I}_m} \prod_{i \in I} p_i \prod_{i \notin I} (1-p_i) \\
&= \sum_{m=1}^{n} w(A_m)\, p_m \prod_{i=1}^{m-1} (1-p_i).
\end{aligned}$$

We used the fact that $\sum_{I \in \mathcal{I}_m} \prod_{i \in I} p_i \prod_{i \notin I} (1-p_i)$ can be interpreted as the probability that m is the minimum of a random set, where i is sampled with probability p_i. Next, we prove that

$$\sum_{m=1}^{n} w(A_m)\, p_m \prod_{i=1}^{m-1} (1-p_i) \geq \left(1 - \left(1 - \frac{1}{n} \right)^n \right) \sum_{m=1}^{n} p_m w(A_m). \tag{4.1}$$

Since this is a linear inequality in the parameters $w(A_m)$, it's enough to prove it for a special case where $w(A_1) = w(A_2) = \ldots = w(A_\ell) = 1$ and $w(A_{\ell+1}) = \ldots = w(A_n) = 0$. (A general decreasing sequence of $w(A_i)$ can be obtained as a positive linear combination of such special cases.) For this special choice of $w(A_i)$, the left-hand side becomes

$$\sum_{m=1}^{\ell} p_m \prod_{i=1}^{m-1}(1-p_i) = 1 - \prod_{i=1}^{\ell}(1-p_i) \geq 1 - \left(1 - \frac{1}{\ell}\sum_{i=1}^{\ell} p_i\right)^{\ell}$$

using the arithmetic-geometric mean inequality. Finally, using the concavity of $\phi(x) = 1 - (1 - \frac{1}{\ell}x)^\ell$, and the fact that $x = \sum_{i=1}^{\ell} p_i \in [0,1]$ and $\phi(0) = 0$, we get $\phi(x) \geq \phi(1) \cdot x$; i.e.,

$$1 - \left(1 - \frac{1}{\ell}\sum_{i=1}^{\ell} p_i\right)^{\ell} \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right)\sum_{i=1}^{\ell} p_i \geq \left(1 - \left(1 - \frac{1}{n}\right)^{n}\right)\sum_{i=1}^{\ell} p_i$$

because $\ell \leq n$. This proves (4.1) and hence the lemma. □

4.2 Submodular Welfare with equal players

Let us consider a special case of the Submodular Welfare Problem where all players have the same utility function. We show that a uniformly random allocation for n players achieves expected approximation factor at least $1 - (1-1/n)^n > 1 - 1/e$. Note that no queries at all are needed to generate this allocation.

Theorem 4.4. *For n players with the same submodular utility function, a uniformly random allocation yields expected value at least* $(1-(1-1/n)^n)\ OPT$.

Proof. This is a simple application of Lemma 4.2. Let $(A_1, A_2, \ldots, A_n)$ be the optimal allocation, and $(R_1, R_2, \ldots, R_n)$ a uniformly random allocation. This means that each R_i has the same distribution, containing each element independently with probability $p = 1/n$. Equivalently, we can write

$$R_i = X(p) = A_1(p) \cup A_2(p) \cup \ldots \cup A_n(p).$$

Applying Lemma 4.2, we get

$$\mathbf{E}[w(R_i)] \geq \left(1 - \left(1 - \frac{1}{n}\right)^{n}\right)\sum_{i=1}^{n} p\ w(A_i) = \frac{1}{n}\left(1 - \left(1 - \frac{1}{n}\right)^{n}\right) OPT.$$

Thus all players together obtain $\sum_{i=1}^{n} \mathbf{E}[w(R_i)] \geq (1-(1-1/n)^n) OPT$. □

4.3 Submodular Welfare with general players

Theorem 4.5. *There is a randomized algorithm for Maximum Submodular Welfare in the value oracle model which returns a* $(1 - 1/e - o(1))$*-approximation in expectation.*

This is a consequence of Theorem 3.1. However, we show a simpler algorithm in this case, which is self-contained and does not require the pipage rounding technique. Before describing our algorithm formally, we explain our intuition behind it. The algorithm works as follows:

1. We run a variant of the greedy algorithm, whose output is a fractional solution y_{ij}. The interpretation of y_{ij} is the extent to which player i receives item j. This solution will satisfy $\sum_{i=1}^{n} y_{ij} = 1$ for every item j. Our aim is to optimize the following function:

$$F(y) = \sum_{i=1}^{n} \mathbf{E}[w_i(R_i)]$$

   where R_i contains each item j independently with probability y_{ij}. We prove that we can find a fractional solution of value $F(y) \geq (1 - 1/e - o(1))OPT$.

2. We allocate each item j independently at random, with probability y_{ij} to player i. This yields a solution of expected value $F(y)$.

The way we build up the fractional solution y_{ij} is through a process that is best viewed as running continuously over a unit time interval. At time $t \in [0, 1]$, we have a fractional solution $y_{ij}(t)$. We start with $y_{ij}(0) = 0$. At time t, we increase exactly one of the variables y_{ij} for each item j, namely the one corresponding to the player who derives the maximum marginal value $\mathbf{E}[w_i(R_i(t) + j) - w_i(R_i(t))]$ (as above, $R_i(t)$ is a random set sampled with probabilities $y_{ij}(t)$). We call this player the *preferred player* for item j at time t. The rate of increase is 1, i.e. we increase y_{ij} exactly by dt over a small time interval dt. In order to implement this in polynomial time, though, we need to discretize the time scale and proceed in finite steps of length δ. It is enough to choose δ sufficiently small, e.g. $\delta = 1/m^2$ where m is the number of items. Next, we describe the algorithm formally.

The Smooth Greedy Algorithm.

1. Let $\delta = 1/m^2$. Start with $t = 0$ and $y_{ij}(0) = 0$ for all i, j.

2. Let $R_i(t)$ be a random set containing each item j independently with probability $y_{ij}(t)$. For all i, j, estimate the expected marginal profit of player i from item j,

$$\omega_{ij}(t) = \mathbf{E}[w_i(R_i(t) + j) - w_i(R_i(t))].$$

By repeated sampling, we can get an arbitrarily close estimate in polynomial time w.h.p.

3. For each j, let $i_j(t) = \text{argmax}_i\, \omega_{ij}(t)$ be the preferred player for item j (breaking possible ties arbitrarily). Set $y_{ij}(t+\delta) = y_{ij}(t)+\delta$ if $i = i_j(t)$ and $y_{ij}(t + \delta) = y_{ij}(t)$ otherwise.

4. Set $t := t+\delta$; if $t < 1$, go back to Step 2. Otherwise, return $y_{ij} = y_{ij}(1)$.

Lemma 4.6. *The fractional solution found by the Smooth Greedy Algorithm satisfies $\sum_{i=1}^n y_{ij} = 1$ for each item j. Let $F(y) = \sum_{i=1}^n \mathbf{E}[w_i(R_i)]$ where R_i contains each item j independently with probability y_{ij}. Then*

$$F(y) \geq \left(1 - \frac{1}{e} - o(1)\right) \cdot OPT$$

Proof. For each item j, the algorithm increments the value of y_{ij} for exactly one player i in each step. Hence, $\sum_{i=1}^n y_{ij}$ increases by δ in each step and equals 1 at the end.

As the fractional solution y_{ij} evolves, $F(y) = \sum_{i=1}^n \mathbf{E}[w_i(R_i)]$ grows, by the monotonicity of w_i. To analyze the growth of $F(y)$, we use the following bound on OPT: For any $y \in [0,1]^N$,

$$OPT \leq F(y) + \sum_j \max_i \omega_{ij} \tag{4.2}$$

where $\omega_{ij} = \mathbf{E}[w_i(R_i + j) - w_i(R_i)]$ and R_i is sampled with probabilities y_{ij}. This can be seen as follows: Consider an optimal solution $(O_1, \ldots, O_n)$. By submodularity, $w_i(O_i) \leq w_i(R_i) + \sum_{j \in O_i}(w_i(R_i + j) - w_i(R_i))$ for any set R_i, and therefore

$$\begin{aligned} OPT &= \sum_i w_i(O_i) \leq \sum_i \mathbf{E}[w_i(R_i) + \sum_{j \in O_i}(w_i(R_i + j) - w_i(R_i))] \\ &= F(y) + \sum_{i, j \in O_i} \omega_{ij} \leq F(y) + \sum_j \max_i \omega_{ij} \end{aligned}$$

since every item j is in at most one set O_i. This proves (4.2).

Our goal is to estimate how much $F(y)$ increases during one step of the algorithm. Consider the random set $R_i(t)$ defined by $y_{ij}(t)$, and an independently random set $D_i(t)$ that contains each item j independently with probability $\Delta_{ij}(t) = y_{ij}(t+\delta) - y_{ij}(t)$. Note that this difference can be either δ or 0, depending on whether y_{ij} was incremented at time t. Let $E_i(t)$ denote the set of items j for which $\Delta_{ij}(t) = \delta$; this is the set of items for which i is the preferred player at time t. The sets $E_i(t)$ form a partition of all items and $D_i(t)$ is a random subset of $E_i(t)$ where each element appears independently with probability δ.

The first (easy) claim is that $F(y(t+\delta)) = \sum_i \mathbf{E}[w_i(R_i(t+\delta))] \geq \sum_i \mathbf{E}[w_i(R_i(t) \cup D_i(t))]$. This follows from monotonicity, because the random set $R_i(t+\delta)$ contains items independently with probabilities $y_{ij}(t) + \Delta_{ij}(t)$, while $R_i(t) \cup D_i(t)$ contains items independently with (smaller) probabilities $1-(1-y_{ij}(t))(1-\Delta_{ij}(t))$.

Now we are ready to estimate how much $F(y)$ gains at time t. The important fact here is that the probability that any item appears in $D_i(t)$ is very small, so we can focus on the contributions from sets $D_i(t)$ that turn out to be singletons.

$$\begin{aligned}
F(y(t+\delta)) - F(y(t)) &\geq \sum_i \mathbf{E}[w_i(R_i(t) \cup D_i(t)) - w_i(R_i(t))] \\
&\geq \sum_i \sum_j \Pr[D_i(t) = \{j\}]\ \mathbf{E}[w_i(R_i(t)+j) - w_i(R_i(t))] \\
&= \sum_i \sum_{j \in E_i(t)} \delta(1-\delta)^{|E_i(t)|-1} \omega_{ij}(t) \\
&\geq \delta(1-\delta)^{m-1} \sum_i \sum_{j \in E_i(t)} \omega_{ij}(t).
\end{aligned}$$

Recall that each item j is in exactly one set $E_i(t)$, for i such that $\omega_{ij}(t)$ is maximized. Hence,

$$\begin{aligned}
F(y(t+\delta)) - F(y(t)) &\geq \delta(1-\delta)^{m-1} \sum_j \max_i \omega_{ij}(t) \\
&\geq \delta(1-\delta)^{m-1}(OPT - F(y(t)))
\end{aligned}$$

using (4.2). From here, $OPT - F(y(t+\delta)) \leq (1-\delta(1-\delta)^{m-1})(OPT - F(y(t)))$ and by induction,

$$OPT - F(y(k\delta)) \leq (1-\delta(1-\delta)^{m-1})^k (OPT - F(y(0))) = (1-\delta(1-\delta)^{m-1})^k\, OPT.$$

With our choice of $\delta = 1/m^2$ and $k = m^2$, we get

$$OPT - F(y(1)) \leq \left(1 - \frac{1}{m^2}\left(1 - \frac{1}{m^2}\right)^{m-1}\right)^{m^2} OPT \leq e^{-(1-1/m)} OPT.$$

Hence, $F(y(1)) \geq (1 - 1/e - o(1))OPT$ as we claimed. □

In the second stage of our algorithm, we need to convert the fractional solution into an integral one. This is very easy here, since we can use randomized rounding directly to find a desired solution.

Randomized rounding.

- Allocate each item j independently at random, to player i with probability y_{ij}.

The random sets R_i received by different players are not independent, but the items in each set appear independently, just like in the definition of $F(y)$. The total expected value of our solution is $\sum_{i=1}^{n} \mathbf{E}[w_i(R_i)] = F(y) \geq (1 - 1/e - o(1))OPT$.

4.4 Value query complexity bounds

On the negative side, we construct examples showing the following bounds on the value-query complexity of the Submodular Welfare Problem.

Theorem 4.7. *For any fixed $\beta > 0$, there is no $(3/4 + \beta)$-approximation algorithm for 2 players with submodular utility functions in the value oracle model, using a subexponential number of queries.*

More generally, we show the following.

Theorem 4.8. *For any fixed $\beta > 0$ and $n \geq 2$, there is no $(1-(1-1/n)^n+\beta)$-approximation algorithm for n players with submodular utility functions in the value oracle model, using a subexponential number of queries.*

Corollary 4.9. *For any fixed $\beta > 0$, there is no $(1-1/e+\beta)$-approximation for an arbitrary number of players, using a subexponential number of queries.*

We note that our examples use the same submodular utility function for all players. We showed in Section 4.2 that in this setting, the $1 - (1 - 1/n)^n$ approximation factor is obtained by a uniformly random allocation. Therefore, in some sense value queries do not bring any benefit when the utility functions are equal.

4.4.1 The bound for 2 players

First we prove the bound for 2 players (Theorem 4.7). This bound can be obtained easily from our example showing the value-query hardness of $(1/2+\epsilon)$-approximation for maximization of symmetric submodular functions [23]. Here, we choose a slightly more involved approach, using a construction that generalizes more easily to the case of n players.

We define monotone submodular functions on a ground set X. The ground set is partitioned into (X_1, X_2). This partition can be thought of as random and "hidden" from the algorithm. The functions $f(S)$ that we define depend only on the fractions of X_1, X_2 that S contains: $x_1 = |S \cap X_1|/|X_1|$, $x_2 = |S \cap X_2|/|X_2|$. We think of x_1, x_2 as real variables in $[0,1]$ and define our functions as $f(x_1, x_2)$ - we refer the reader to Section 2.3.2 for a justification of this. Due to Lemma 2.18, the fact that f is monotone is implied by $\frac{\partial f}{\partial x_i} \geq 0$ for $i \in \{1,2\}$. The fact that f is submodular is implied by $\frac{\partial^2 f}{\partial x_i \partial x_j} \leq 0$ for all $i, j \in \{1,2\}$ (possibly $i = j$). Thus we will try to construct functions with these analytic properties. In accordance with Section 2.3.2, we call these functions *smooth submodular*.

We will consider instances of the Submodular Welfare Problem where players have the same utility function. A solution to the problem corresponds to a choice of $x_1, x_2 \in [0,1]$ such that player 1 obtains value $f(x_1, x_2)$ and player 2 obtains $f(1-x_1, 1-x_2)$. Thus, it will be sufficient to prove the following.

Lemma 4.10. *For any $\beta > 0$, there is $\epsilon > 0$ and two smooth submodular functions $f, g : [0,1]^2 \to \mathbb{R}_+$ such that*

- *For $|x_1 - x_2| \leq \epsilon$, $f(x_1, x_2) = g(x_1, x_2)$.*
- $\max_{x_1,x_2}(f(x_1, x_2) + f(1-x_1, 1-x_2)) \geq 2 - \beta$.
- $\max_{x_1,x_2}(g(x_1, x_2) + g(1-x_1, 1-x_2)) \leq 3/2 + \beta$.

From these functions, we can reconstruct discrete set functions using the formula $f(S) = f(|S \cap X_1|/|X_1|, |S \cap X_2|/|X_2|)$. These functions are monotone, submodular, and the optima of the respective allocation problems differ by a factor close to 3/4. For any algorithm using a subexponential number of value queries, the two functions are indistinguishable, because $f(x_1, x_2) = g(x_1, x_2)$ for all queries such that $|x_1 - x_2| \leq \epsilon$. Due to standard Chernoff bounds, this happens with high probability when the number of elements tends to infinity. Hence, no algorithm using subexponentially many value queries can decide whether the optimum is $2-\beta$ or $3/2+\beta$. It remains to prove Lemma 4.10.

Proof. We start by considering two smooth submodular functions,

- $f(x_1, x_2) = x_1 + x_2 - x_1 x_2$
- $g(x_1, x_2) = x_1 + x_2 - \frac{1}{4}(x_1 + x_2)^2$.

In the discrete picture, these functions can be interpreted as the numbers of edges incident with S in a complete bipartite graph on (X_1, X_2) (for $f(S)$) or a complete graph with edge weights $1/2$ (for $g(S)$). Observe that the maximum of $f(x_1, x_2) + f(1-x_1, 1-x_2)$ is 2, obtained for $x_1 = 1, x_2 = 0$, while the maximum of $g(x_1, x_2) + g(1-x_1, 1-x_2)$ is $3/2$, obtained for $x_1 = x_2 = 1/2$. Also, the two functions coincide for $x_1 = x_2$. However, it is not true that the two functions coincide for $|x_1 - x_2| \le \epsilon$. Our goal is to perturb $f(x_1, x_2)$ in order to make it equal to $g(x_1, x_2)$ on this "diagonal zone", without corrupting its submodularity or the value of its optimum significantly. A suitable value of ϵ will be chosen at the end.

Let $h(x_1, x_2)$ denote the difference of the two functions,

- $h(x_1, x_2) = f(x_1, x_2) - g(x_1, x_2) = \frac{1}{4}(x_1 - x_2)^2$.

We construct the perturbed function as

- $\tilde{f}(x_1, x_2) = f(x_1, x_2) - \phi(h(x_1, x_2))$

where $\phi : \mathbb{R} \to \mathbb{R}$ has the following properties:

- For $t \in [0, \epsilon_1], \epsilon_1 = \frac{1}{4}\epsilon^2$, we set $\phi(t) = t$. Hence, in this range, which corresponds to $|x_1 - x_2| \le \epsilon$, we have $\tilde{f}(x_1, x_2) = f(x_1, x_2) - h(x_1, x_2) = g(x_1, x_2)$.
- For $t \in [\epsilon_1, \epsilon_2]$, we define ϕ by specifying that its first derivative is continuous at $t = \epsilon_1$ and its second derivative is $\phi''(t) = -\alpha/t$ for $t \in [\epsilon_1, \epsilon_2]$. (We choose a suitable constant $\alpha > 0$ later.) Hence,

$$\phi'(t) = 1 - \int_{\epsilon_1}^{t} \frac{\alpha}{\tau} d\tau = 1 - \alpha \ln \frac{t}{\epsilon_1}.$$

  We choose $\alpha = 2/\ln\frac{1}{\epsilon_1}$ and $\epsilon_2 = \sqrt{\epsilon_1}$, so that $\phi'(\epsilon_2) = 0$. In other words, we make ϕ concave, with a controlled second derivative, up to a point where its first derivative becomes zero. The value at this point can be computed, but it suffices for us to observe that $\phi(\epsilon_2) \le \epsilon_2 \le \epsilon$.
- For $t > \epsilon_2$, we set $\phi(t) = \phi(\epsilon_2)$.

We have $0 \le \phi(t) \le \epsilon$ everywhere and therefore

$$\tilde{f}(x_1, x_2) = f(x_1, x_2) - \phi(h(x_1, x_2)) \ge f(x_1, x_2) - \epsilon.$$

It remains to show that we didn't corrupt the monotonicity and submodularity of f too badly. We have

$$\frac{\partial \tilde{f}}{\partial x_j} = \frac{\partial f}{\partial x_j} - \phi'(h)\frac{\partial h}{\partial x_j} = (1 - \phi'(h))\frac{\partial f}{\partial x_j} + \phi'(h)\frac{\partial g}{\partial x_j}.$$

The way we constructed ϕ, we have $0 \le \phi'(h) \le 1$, therefore $\frac{\partial \tilde{f}}{\partial x_j}$ is a convex combination of $\frac{\partial f}{\partial x_j}$ and $\frac{\partial g}{\partial x_j}$ which are both nonnegative. So, $\frac{\partial \tilde{f}}{\partial x_j} \ge 0$.

For the second partial derivatives, we get

$$\begin{aligned}\frac{\partial^2 \tilde{f}}{\partial x_i \partial x_j} &= \frac{\partial^2 f}{\partial x_i \partial x_j} - \phi'(h)\frac{\partial^2 h}{\partial x_i \partial x_j} - \phi''(h)\frac{\partial h}{\partial x_i}\frac{\partial h}{\partial x_j} \\ &= (1 - \phi'(h))\frac{\partial^2 f}{\partial x_i \partial x_j} + \phi'(h)\frac{\partial^2 g}{\partial x_i \partial x_j} - \phi''(h)\frac{\partial h}{\partial x_i}\frac{\partial h}{\partial x_j}.\end{aligned}$$

Again, we get a convex combination of the second derivatives of f and g, which are non-positive, but also an additional term $-\phi''(h)\frac{\partial h}{\partial x_i}\frac{\partial h}{\partial x_j}$ which could potentially cause some trouble. (This is why we have to control the second derivative of ϕ carefully.) Recall that $h(x_1, x_2) = \frac{1}{4}(x_1 - x_2)^2$ and so

$$\left|\frac{\partial h}{\partial x_i}\right| = \frac{1}{2}|x_1 - x_2| = \sqrt{h(x_1, x_2)}.$$

Since $|\phi''(h)| \le \alpha/h$, we can conclude that

$$\frac{\partial^2 \tilde{f}}{\partial x_i \partial x_j} \le \left|\phi''(h)\frac{\partial h}{\partial x_i}\frac{\partial h}{\partial x_j}\right| \le \alpha.$$

Recall that $\alpha = 2/\ln\frac{1}{\epsilon_1} = 1/\ln\frac{2}{\epsilon}$. Thus, the second partial derivatives of $\tilde{f}$ could be positive, but not very large. This can be fixed easily by adding a small multiple of $g(x_1, x_2)$ to both functions: let $\hat{f}(x_1, x_2) = \tilde{f}(x_1, x_2) + 2\alpha g(x_1, x_2)$ and $\hat{g}(x_1, x_2) = (1 + 2\alpha)g(x_1, x_2)$. Since $\frac{\partial^2 g}{\partial x_i \partial x_j} = -\frac{1}{2}$ for all i, j, this makes both $\hat{f}$ and $\hat{g}$ smooth submodular.

For a given $\beta > 0$, we choose $\epsilon = 2e^{-2/\beta}$, so that $\beta = 2\alpha$ and we increase g only by a factor of $1 + \beta$. We also get $\epsilon \le \beta$, so $\hat{f}(x_1, x_2) \ge f(x_1, x_2) - \beta$. Therefore, $\hat{f}$ and $\hat{g}$ satisfy the conditions of the lemma. $\square$

4.4.2 The bound for n players

Here we extend the ideas of the previous section to n players. We assume throughout this section that $n \geq 3$. The intuition for 2 players is that it is hard to distinguish between the submodular functions corresponding to the complete bipartite graph and the complete graph with edge weights 1/2. For n players, we design an example based on n-uniform hypergraphs. The submodular functions here correspond to the number of hyperedges incident with a set S. We will show essentially that it is hard to distinguish between the submodular functions corresponding to the complete n-partite hypergraph, and the complete n-uniform hypergraph with appropriately scaled edge weights.

We consider a ground set X partitioned into $X_1 \cup X_2 \cup \ldots \cup X_n$. As before, the functions $f(S)$ that we define depend only on the fractions of X_i that S contains: $x_i = |S \cap X_i|/|X_i|$. We seek functions $f(x_1, \ldots, x_n)$ satisfying $\frac{\partial f}{\partial x_i} \geq 0$ and $\frac{\partial^2 f}{\partial x_i \partial x_j} \leq 0$ for all $i, j \in [n]$, which implies monotonicity and submodularity in the discrete case (see Lemma 2.18). To shorten notation, we write $f(\mathbf{x}) = f(x_1, \ldots, x_n)$. In each instance, all players have the same utility function.

We find two functions f, g such that we have $f(\mathbf{x}) = g(\mathbf{x})$ whenever $\max_{i,j} |x_i - x_j| \leq \epsilon$. This will be the case for most queries, since a typical query is partitioned into n equal parts by $X_1, \ldots, X_n$. (It should be noted here that we keep n constant, while the number of vertices N tends to infinity.) Therefore, f and g will be indistinguishable by a subexponential number of queries. If the gap between the optima of f and g is close to $1-(1-1/n)^n$, this implies that any approximation algorithm improving this factor would require an exponential number of value queries. Therefore it remains to prove the following.

Lemma 4.11. *For any $\beta > 0$ and integer $n \geq 3$, there is $\epsilon > 0$ and two smooth submodular functions $f, g : [0,1]^n \to \mathbb{R}_+$ such that*

- *If $\max_{i,j} |x_i - x_j| \leq \epsilon$, then $f(\mathbf{x}) = g(\mathbf{x})$.*
- $\max\{\sum_i f(x_{i1}, \ldots, x_{in}) \mid \forall j; \sum_i x_{ij} = 1\} \geq (1-\beta)n$.
- $\max\{\sum_i g(x_{i1}, \ldots, x_{in}) \mid \forall j; \sum_i x_{ij} = 1\} \leq (1-(1-1/n)^n + \beta)n$.

Proof. We start by considering two smooth submodular functions, motivated by the discussion above.

- $f(\mathbf{x}) = 1 - \prod_{i=1}^n (1 - x_i)$.
- $g(\mathbf{x}) = 1 - (1 - \bar{x})^n$, where $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$.

In the discrete picture, these function correspond to the following:

$$f(S) = 1 - \prod_{i=1}^{n} \left(1 - \frac{|S \cap X_i|}{|X_i|}\right),$$

the vertex cover function of a complete k-partite hypergraph, and

$$g(S) = 1 - \left(1 - \frac{|S|}{\sum_{i=1}^{n} |X_i|}\right)^n,$$

the (scaled) vertex cover function of a complete k-uniform hypergraph.

The optimal solution with utility function f is $x_{ii} = 1$, $x_{ij} = 0$ for $i \neq j$. This way, each player gets the maximum possible value 1. For g, on the other hand, the value depends only on the average of the coordinates $\bar{x}$. By the concavity of g, the optimum solution is to allocate $\bar{x} = 1/n$ to each player, which gives her a value of $1 - (1 - 1/n)^n$.

It remains to perturb the functions so that $f(\mathbf{x}) = g(\mathbf{x})$ for vectors satisfying $\max_{i,j} |x_i - x_j| \leq \epsilon$. Let $h(\mathbf{x})$ denote the difference of the two functions,

- $h(\mathbf{x}) = f(\mathbf{x}) - g(\mathbf{x}) = (1 - \bar{x})^n - \prod_{i=1}^{n}(1 - x_i)$.

Again, we denote $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$. Also, let $\delta = \max_{i,j} |x_i - x_j|$. First, we estimate $h(\mathbf{x})$ and its first derivatives in terms of $\bar{x}$ and δ. We use very crude bounds, to simplify the analysis.

Claim.

1. $h(\mathbf{x}) \leq n\delta(1 - \bar{x})^{n-1}$.
2. $h(\mathbf{x}) \geq n^{-4}\delta^2(1 - \bar{x})^{n-2}$.
3. $|\frac{\partial h}{\partial x_j}| \leq n\delta(1 - \bar{x})^{n-2}$, i.e. $|\frac{\partial h}{\partial x_j}| \leq n^3(1 - \bar{x})^{n/2-1}\sqrt{h(\mathbf{x})}$.

1. We have $h(\mathbf{x}) = (1-\bar{x})^n - \prod_{i=1}^{n}(1-x_i)$. If $n\delta \geq 1-\bar{x}$, we get immediately $h(\mathbf{x}) \leq (1 - \bar{x})^n \leq n\delta(1 - \bar{x})^{n-1}$. So let's assume $n\delta < 1 - \bar{x}$. Then, since $x_i \leq \bar{x} + \delta$ for all i, we get

$$h(\mathbf{x}) \leq (1-\bar{x})^n - (1-\bar{x}-\delta)^n = (1-\bar{x})^n \left(1 - \left(1 - \frac{\delta}{1 - \bar{x}}\right)^n\right) \leq (1-\bar{x})^n \frac{n\delta}{1 - \bar{x}}.$$

2. For a lower bound on $h(\mathbf{x})$, suppose that $\delta = x_2 - x_1$ and define $\eta = \frac{1}{n-2}(\bar{x} - \frac{1}{2}(x_1 + x_2))$. I.e., $x_1 = \bar{x} - (n-2)\eta - \delta/2$, $x_2 = \bar{x} - (n-2)\eta + \delta/2$, and the average of the remaining coordinates is $\bar{x} + 2\eta$. By the arithmetic-geometric mean inequality, $\prod_{i \neq 1,2}(1 - x_i)$ is maximized when these variables are all equal:

$$\begin{aligned} h(\mathbf{x}) &\geq (1-\bar{x})^n - (1 - \bar{x} + (n-2)\eta + \frac{1}{2}\delta)(1 - \bar{x} + (n-2)\eta - \frac{1}{2}\delta)(1 - \bar{x} - 2\eta)^{n-2} \\ &= (1-\bar{x})^n - (1 - \bar{x} + (n-2)\eta)^2(1 - \bar{x} - 2\eta)^{n-2} + \frac{1}{4}\delta^2(1 - \bar{x} - 2\eta)^{n-2}. \end{aligned}$$

Again by the arithmetic-geometric inequality, the first term is always larger than the second term. If $\eta \leq \frac{1}{n}(1-\bar{x})$, we are done because then the last term is at least $\frac{1}{4e^2}\delta^2(1-\bar{x})^{n-2}$. So we can assume $\eta > \frac{1}{n}(1-\bar{x})$. In this case, we throw away the last term and write

$$\begin{aligned} h(\mathbf{x}) &\geq (1-\bar{x})^n - (1 - \bar{x} + (n-2)\eta)^2(1 - \bar{x} - 2\eta)^{n-2} \\ &= (1-\bar{x})^n \left(1 - \left(1 + (n-2)\frac{\eta}{1-\bar{x}}\right)^2 \left(1 - \frac{2\eta}{1-\bar{x}}\right)^{n-2}\right) \\ &\geq (1-\bar{x})^n \left(1 - \left(1 + \frac{\eta}{1-\bar{x}}\right)^{2(n-2)} \left(1 - \frac{\eta}{1-\bar{x}}\right)^{2(n-2)}\right) \\ &= (1-\bar{x})^n \left(1 - \left(1 - \frac{\eta^2}{(1-\bar{x})^2}\right)^{2(n-2)}\right) \\ &\geq (1-\bar{x})^n \left(1 - \left(1 - \frac{1}{n^2}\right)^{2(n-2)}\right) \geq \frac{1}{n^2}(1-\bar{x})^n. \end{aligned}$$

We observe it always holds that $\delta \leq n(1-\bar{x})$: If the minimum coordinate is x_{min}, we have $\bar{x} \leq \frac{1}{n}x_{min} + \frac{n-1}{n} \cdot 1$, hence $x_{min} \geq n\bar{x} - (n-1)$ and $\delta \leq 1 - x_{min} \leq n(1-\bar{x})$.

Consequently, $h(\mathbf{x}) \geq n^{-2}(1-\bar{x})^n \geq n^{-4}\delta^2(1-\bar{x})^{n-2}$.

3. Let $\delta = \max_{i,j}|x_i - x_j|$. We estimate the partial derivative

$$\frac{\partial h}{\partial x_j} = \prod_{i \neq j}(1 - x_i) - (1-\bar{x})^{n-1}.$$

Define $\eta = \frac{1}{n-1}(x_j - \bar{x})$. I.e., $x_j = \bar{x} + (n-1)\eta$ and the average of the remaining coordinates is $\bar{x} - \eta$. By the arithmetic-geometric inequality,

$$\frac{\partial h}{\partial x_j} \leq (1 - \bar{x} + \eta)^{n-1} - (1-\bar{x})^{n-1} = (1-\bar{x})^{n-1}\left(\left((1 + \frac{\eta}{1-\bar{x}}\right)^{n-1} - 1\right).$$

Since $\eta = \frac{1}{n-1}(x_j - \bar{x}) \leq \frac{1}{n-1}(1-\bar{x})$, we can estimate $(1+\frac{\eta}{1-\bar{x}})^{n-1} \leq 1+2n\frac{\eta}{1-\bar{x}}$. Also, we know that all coordinates differ from $\bar{x}-\eta$ by at most δ, in particular $x_j = \bar{x} + (n-1)\eta \leq \bar{x} - \eta + \delta$, hence $n\eta \leq \delta$ and

$$\frac{\partial h}{\partial x_j} \leq (1-\bar{x})^{n-1} \cdot 2n\frac{\eta}{1-\bar{x}} \leq 2\delta(1-\bar{x})^{n-2}.$$

For a lower bound, it's enough to observe that each coordinate is at most $\bar{x}+\delta$, and so

$$\begin{aligned} \frac{\partial h}{\partial x_j} &\geq (1-\bar{x}-\delta)^{n-1} - (1-\bar{x})^{n-1} \\ &= (1-\bar{x})^{n-1}\left(\left(1-\frac{\delta}{1-\bar{x}}\right)^{n-1} - 1\right) \\ &\geq (1-\bar{x})^{n-1}\left(-(n-1)\frac{\delta}{1-\bar{x}}\right) \\ &= -(n-1)\delta(1-\bar{x})^{n-2} \end{aligned}$$

assuming that $(n-1)\frac{\delta}{1-\bar{x}} \leq 1$; otherwise we get the same bound directly from $\frac{\partial h}{\partial x_j} \geq -(1-\bar{x})^{n-1}$. This finishes the proof of the claim.

We return to our construction. We define $\tilde{f}(\mathbf{x}) = f(\mathbf{x}) - \phi(h(\mathbf{x}))$ where $\phi : \mathbb{R} \to \mathbb{R}$ is defined similarly as in the proof of Lemma 4.10.

- For $t \in [0, \epsilon_1]$, we set $\phi(t) = t$. We choose $\epsilon_1 = n\epsilon$. I.e., for $\max_{i,j} |x_i - x_j| \leq \epsilon$, we have $h(\mathbf{x}) \leq \epsilon_1$ by Claim 1 and then $\tilde{f}(\mathbf{x}) = g(\mathbf{x})$.
- For $t \in [\epsilon_1, \epsilon_2]$, the first derivative of ϕ is continuous at $t = \epsilon_1$ and its second derivative is $\phi''(t) = -\alpha/t$ for $t \in [\epsilon_1, \epsilon_2]$. Hence,

  $$\phi'(t) = 1 - \int_{\epsilon_1}^{t} \frac{\alpha}{\tau} d\tau = 1 - \alpha \ln \frac{t}{\epsilon_1}.$$

  We choose $\alpha = 2/\ln\frac{1}{\epsilon_1}$ and $\epsilon_2 = \sqrt{\epsilon_1}$, so that $\phi'(\epsilon_2) = 0$. Since $\phi'(t) \leq 1$ everywhere, we have $\phi(\epsilon_2) \leq \epsilon_2$.
- For $t > \epsilon_2$, we set $\phi(t) = \phi(\epsilon_2)$.

Hence, we have $0 \leq \phi(t) \leq \epsilon_2$ everywhere and $\tilde{f}(\mathbf{x}) = f(\mathbf{x}) - \phi(h(\mathbf{x})) \geq f(\mathbf{x}) - \epsilon_2$. Next, we want to show that we didn't corrupt the monotonicity and submodularity of f too badly. We have

$$\frac{\partial \tilde{f}}{\partial x_j} = \frac{\partial f}{\partial x_j} - \phi'(h)\frac{\partial h}{\partial x_j} = (1-\phi'(h))\frac{\partial f}{\partial x_j} + \phi'(h)\frac{\partial g}{\partial x_j}.$$

Exactly as in the case of 2 players, $0 \leq \phi'(h) \leq 1$, and $\frac{\partial f}{\partial x_j}$, $\frac{\partial g}{\partial x_j}$ are both nonnegative. So, $\frac{\partial \tilde{f}}{\partial x_j} \geq 0$. For the second partial derivatives, we get

$$\begin{aligned}\frac{\partial^2 \tilde{f}}{\partial x_i \partial x_j} &= \frac{\partial^2 f}{\partial x_i \partial x_j} - \phi'(h)\frac{\partial^2 h}{\partial x_i \partial x_j} - \phi''(h)\frac{\partial h}{\partial x_i}\frac{\partial h}{\partial x_j} \\ &= (1-\phi'(h))\frac{\partial^2 f}{\partial x_i \partial x_j} + \phi'(h)\frac{\partial^2 g}{\partial x_i \partial x_j} - \phi''(h)\frac{\partial h}{\partial x_i}\frac{\partial h}{\partial x_j}.\end{aligned}$$

The first two terms form a convex combination of non-positive values. To control the third term, we have $|\phi''(h)| \leq \alpha/h$. We also showed $\left|\frac{\partial h}{\partial x_i}\right| \leq n^3(1-\bar{x})^{n/2-1}\sqrt{h(\mathbf{x})}$ (Claim 3). We can conclude that

$$\frac{\partial^2 \tilde{f}}{\partial x_i \partial x_j} \leq \left|\phi''(h)\frac{\partial h}{\partial x_i}\frac{\partial h}{\partial x_j}\right| \leq \alpha n^6 (1-\bar{x})^{n-2}.$$

We need to make the second partial derivatives non-positive. Since $\frac{\partial^2 g}{\partial x_i \partial x_j} = -\frac{n-1}{n}(1-\bar{x})^{n-2}$, it is enough to add a suitable multiple of g to both functions: $\hat{f} = \tilde{f} + 2\alpha n^6 g$, $\hat{g} = (1+2\alpha n^6)g$. Then $\hat{f}, \hat{g}$ are smooth submodular.

Recall that we have $\alpha = 2/\ln\frac{1}{n\epsilon}$. For a given $\beta > 0$, we choose $\epsilon = \frac{1}{n}e^{-4n^6/\beta}$, so that $\beta = 2\alpha n^6$ and we increase g only by a factor of $1+\beta$. We also get $\epsilon_2 = \sqrt{n\epsilon} \leq \beta$, and therefore $\hat{f}(\mathbf{x}) \geq \tilde{f}(\mathbf{x}) \geq f(\mathbf{x}) - \epsilon_2 \geq f(\mathbf{x}) - \beta$. Thus $\hat{f}$ and $\hat{g}$ satisfy the conditions of the lemma. □

Remark. In fact, our construction shows the following: In the value query model, it is impossible to distinguish between instances with n players and kn items where each player can get value close to 1 and instances where no set of k items has value much more than $1-1/e$.

Chapter 5

Submodular Welfare with demand queries

Here we continue studying the Submodular Welfare Problem, however now in the demand oracle model. Let us recall the statement of the problem.

Problem: *Given n players with utility functions $w_i : 2^X \to \mathbb{R}_+$ which are assumed to be monotone and submodular, find a partition $X = S_1 \cup S_2 \cup \ldots \cup S_n$ in order to maximize $\sum_{i=1}^{n} w_i(S_i)$.*

Following the greedy $\frac{1}{2}$-approximation of [38] and a lack of further progress in the value oracle model, several works [13, 14, 18] considered the following linear programming relaxation of the problem.

Configuration LP.

$$\begin{aligned} \max \sum_{i,S} x_{i,S} w_i(S); \\ \forall j; \sum_{i,S: j \in S} x_{i,S} \leq 1, \\ \forall i; \sum_{S} x_{i,S} \leq 1, \\ \forall i, S; x_{i,S} \geq 0. \end{aligned}$$

Here, $x_{i,S}$ is intended to be an indicator variable that specifies whether player i gets set S. The constraints express the facts that every player chooses at most one set and every item can be allocated to at most one player. This linear program has an exponential number of variables but only a polynomial number of constraints. It can be solved optimally in the demand oracle model, since the separation oracle for the dual is exactly the demand oracle. See [44, 13] for more details.

Therefore, if we show that any fractional solution of this LP can be rounded to an integral solution while losing at most a factor of ρ, then we obtain an ρ-approximation algorithm in the demand oracle model. Using this approach, a $(1-1/e)$-approximation was obtained in [14]. However, it was not clear whether this is the best approximation one can obtain using the Configuration LP.

Some intuition might indicate that the $1-1/e$ is indeed the integrality gap of the Configuration LP. In particular, this LP is a special case of an optimization problem over a partition matroid polytope:

$$\max\{f^+(y) : y \in P(\mathcal{M})\}.$$

See Section 3.5 for the definition of $f^+(y)$ and Section 1.2.3 for a reduction of the Submodular Welfare Problem to a partition matroid constraint. We know that $\max\{f^+(y) : y \in P(\mathcal{M})\}$ has an integrality gap arbitrarily close to $1-1/e$ (Section 3.8), even for a coverage-type f and a partition matroid. Also, the Configuration LP itself can have an integrality gap arbitrarily close to $1-1/e$ when the utility functions are fractionally subadditive rather than submodular [18].

Yet, $1-1/e$ is not the optimal answer for submodular utility functions. We prove the following.

Theorem 5.1. *There is some universal constant $\epsilon > 0$ and a randomized rounding procedure for the Configuration LP, such that given any feasible fractional solution, the rounding procedure produces a feasible allocation whose expected value is at least a $(1-1/e+\epsilon)$-fraction of the value of the fractional solution.*

Our rounding procedure is oblivious in the sense of [18]: its only input is a fractional LP solution and it need not know anything about the actual utility functions of the players.

The following corollary follows from combining Theorem 5.1 with the fact that demand queries suffice in order to find an optimal fractional solution to the Configuration LP.

Corollary 5.2. *The Submodular Welfare Problem can be approximated in the demand oracle model within a ratio of $1-1/e+\epsilon$ (in expectation), for some absolute constant $\epsilon > 0$.*

5.1 Overview of techniques

First, let us recall how one achieves an approximation ratio of $1 - 1/e$ (or in fact, $1 - (1 - 1/n)^n$) for the Submodular Welfare Problem [14, 18]. One uses the given feasible solution to the Configuration LP in the following way. Every player independently selects a tentative set, where the probability that player i selects set S is precisely $x_{i,S}$. (If for some player, $\sum_S x_{i,S} < 1$, then it may happen that the player will select no tentative set at all.) Per player, the expected utility after this step is equal to her contribution to the value of the fractional solution of the Configuration LP. However, the tentative allocation may not be feasible, because some items may be in more than one tentative set. To resolve this issue, one uses a "contention resolution procedure". In [14], contention resolution is based on further queries to the players is order to determine which player will benefit the most from getting the contended item. In [18], contention resolution is done in an oblivious way, without further interaction with the players. In both cases it is shown that contention resolution can be done while preserving (in expectation) at least a $(1 - 1/e)$-fraction of the total utility.

The above two-step rounding procedure seems to have a lot of slackness. Namely, there are many items that are not allocated at all because they are not in any tentative set. In fact, on what appear to be worst case instances of the two-step rounding procedure, every item has probability roughly $1/e$ of not being allocated at all. It appears that adding a third step to this rounding procedure, in which the remaining items are allocated, could potentially allow one to improve the $1 - 1/e$ ratio.

Somewhat surprisingly, one can design instances (see Section 5.4.2) where the utility functions are submodular, and regardless of how contention resolution is performed, and of how items outside the tentative sets are allocated, one cannot obtain an approximation ratio better than $1 - 1/e$.

We show in Section 5.4.1 that there is also a simpler *one-step* randomized rounding procedure (namely, item j is given to player i with probability $\sum_{j \in S} x_{i,S}$, without a need to first choose tentative sets), which achieves an approximation ratio of $1-1/e$ (though not $(1-(1-1/n)^n)$). One may hope that on every instance, the best of the two rounding procedures (the three-step procedure and the one-step procedure) would give an approximation ratio better than $1 - 1/e$. Nevertheless, again, this is not true.

Our new algorithm (that does improve $1 - 1/e$) is based on a random choice between two rounding procedures. The analysis of our algorithm uses algebraic manipulations that might not be easy to follow. To help the reader see the reasoning behind our final algorithm, we break its derivation into five stages.

The first stage (Section 5.2.3) is perhaps the most instructive one, as it addresses a simple case which is relatively easy to understand. There are only two players, and the fractional solution is half-integral. The goal is to design a rounding procedure that improves the ratio of $1-(1-1/2)^2 = 3/4$ guaranteed by previous rounding procedures. Neither the three-step rounding procedure (as described above, based on each player choosing one tentative set) nor the one-step rounding procedure achieve such an improvement. We present a new rounding procedure that achieves an approximation ratio of 5/6. Moreover, our analysis of the approximation ratio serves as an introduction to the kind of algebraic manipulations that will be used in the more complicated proofs. We also show that the 5/6 ratio for this case is best possible, by showing a matching integrality gap.

Section 5.2.4 deals with the case of two players and a balanced fractional solution, in the sense that for every item j, $\sum_{S|j\in S} x_{1,S} = \sum_{S|j\in S} x_{2,S} = 1/2$. We design an algorithm using two tentative sets S, S' for player 1 and T, T' for player 2, sampled independently according to the fractional solution. Using two sets instead of one allows us to allocate more items overall, and also it gives us more freedom in designing an allocation scheme. Using a submodularity argument inspired by the half-integral case, we show that either a player gains by taking the entire complement of the other player's tentative set, or she can combine items from S, S' to obtain what we call a "diagonal set" Y. Similarly, player 2 obtains a diagonal set Z. The sets Y and Z are designed so that their average overlap is less than that of S and T. Thus by resolving contention between Y and Z, we get a factor better than 3/4. Specifically, we obtain a 7/9-approximation in this case.

Section 5.2.5 combines the balanced and unbalanced case for two players. We gain for different reasons in the balanced and unbalanced cases, but we always beat the factor of 3/4. For two players, we obtain an approximation factor of 13/17. This result convinced us that an improvement over $1 - 1/e$ in the general case of n players should be possible.

Section 5.3.1 presents the "Fair Rounding Procedure" for n players, which achieves at least a $(1 - 1/e)$-approximation, and actually beats it unless the fractional solution is balanced. This stage is based on a *Fair Contention Resolution* technique which might be interesting on its own. It is a variation and simplification of the contention resolution technique used in [18]. Apart from other interesting features (that we discuss in Section 5.1.1 below), the Fair Rounding Procedure procedure has the following property. Call a fractional solution to the LP *unbalanced* if for a "typical" item j, there is some player i (that may depend on j) for which $\sum_{S|j\in S} x_{i,S} > \epsilon$, where $\epsilon > 0$ is some fixed constant. Then the approximation ratio provided by our procedure is

$1 - 1/e + \epsilon^{O(1)}$.

It remains to handle the case of balanced fractional solutions. In Section 5.3.2, we develop the "Butterfly Rounding Procedure" which achieves an improvement over $1 - 1/e$ for any balanced fractional solution. This is the key part of the proof of Theorem 5.1. It is based on ideas used for the two-player case, but again, with added complications. The main structural difference between this rounding procedure and earlier ones is that we let every player choose two tentative sets rather than one. Thereafter, we perform contention resolution for every item that is in tentative sets of more than one player. The exact way in which we perform contention resolution is rather complicated.

Finally, the two rounding procedures are combined to obtain an approximation ratio strictly above $1-1/e$ for any fractional solution (Section 5.3.3).

5.1.1 Fair Contention Resolution

A key component in previous $(1 - 1/e)$-approximation algorithms for Submodular Welfare (and for more general classes of utility functions as well) is a method for resolving contention among several tentative sets that contain the same item. In our current work, we generalize and improve upon the method used for this purpose in [18] so that it can be combined more easily with other parts of our new rounding procedure. Our method gives a solution to a problem that we call *Fair Contention Resolution.* We now describe this problem.

There are n players and one item. Every player i is associated with a probability $0 \leq p_i \leq 1$ of requesting the item. Our goal is to allocate the item to at most one player, and have the probability that a player i receives the item be proportional to its respective p_i. We call this *balanced contention resolution.* Among all such contention resolution schemes, we wish to find one that maximizes for the players the probability that they get the item (the balancing requirement implies that maximizing this probability for one player maximizes it for all players). Given complete coordination among the players, we may assign the item to player i with probability $p_i / \sum_j p_j$, and this would be optimal. But we will be dealing with a two-step situation in which there is only partial coordination.

1. In step 1, there is no coordination among the players. Every player i independently requests the item with probability p_i.

2. In step 2, those players who requested the item in step 1 may coordinate a (randomized) strategy for allocating the item to one of them.

The probability that the item is allocated at all is at most the probability that the set of players reaching step 2 is nonempty, namely, $1 - \prod_j (1 - p_j)$. Hence in balanced contention resolution, player i can get the item with probability at most $\frac{p_i}{\sum_j p_j}(1 - \prod_j (1 - p_j))$. What we call *Fair Contention Resolution* is a method which indeed attains this maximum. In previous work [18], such contention resolution methods were designed for some special cases. Here, we prove the following general statement.

There is an explicit Fair Contention Resolution method. More specifically, there is a simple formula that for every set A and player $i \in A$, determines the probability with which the item is given to i conditioned on A being the set of players that reach step 2. This formula depends on the size of A, on the values of p_j for those $j \in A$, and on the sum $\sum_{j=1}^{n} p_j$.

See Lemma 5.12 for a proof of this claim.

5.2 The allocation problem for two players

In this section, we start working towards the goal of proving Theorem 5.1. First we consider a special case of the welfare maximization problem where only two players are interested in a given set of m items. The analysis of this case is not formally needed for the proof of Theorem 5.1 but we consider it instructive for the reader to understand this simpler case before proceeding to the proof of Theorem 5.1. Also, our techniques are really geared to the case of two rather than n players. The improvement we achieve here (from 3/4 to 13/17) is relatively significant, as opposed to the purely theoretical improvement in Theorem 5.1.

In the setting with two players, it is not very difficult to achieve an approximation factor of 3/4 (even assuming only fractional subaditivity), as shown in [18]. Since our improvements are built on top of the 3/4-approximation algorithm, let's review it first.

5.2.1 3/4-approximation for two players

The basic idea is to use the fractional LP solution to generate random sets suitable for each player. When we say that "player 1 samples a random set from his distribution", it means that he chooses set S with probability $x_{1,S}$. (Since $\sum_S x_{1,S} \leq 1$, this is a valid probability distribution.) Similarly, player 2 samples a random set T from her distribution defined by $x_{2,T}$. We define

- $p_j = \Pr[j \in S] = \sum_{S: j \in S} x_{1,S}$.

- $q_j = \Pr[j \in T] = \sum_{T: j \in T} x_{2,T}$.

Ideally, we would like to assign S to player 1 and T to player 2 which would yield an expected value equal to the LP optimum. The only issue is that the sets S and T can overlap, so we cannot satisfy the players' requests exactly. One way to allocate the disputed items is by making a random decision. It turns out that the best way to do this is to allocate disputed items to one of the two players with *reversed probabilities* compared to the fractional solution (see [18]). For this purpose, we use a random "splitting set" X which contains each item j with probability p_j.

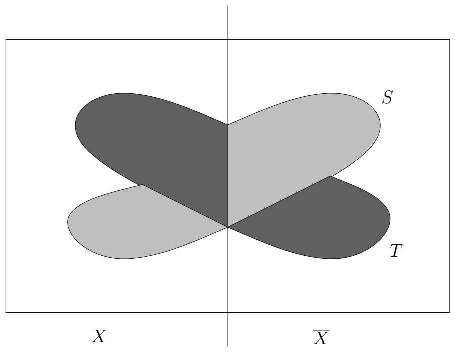

Figure 5.1: Algorithm 1.

Algorithm 1. (3/4-approximation for 2 players)

- Let player 1 sample a random set S and let player 2 sample a random set T from their respective distributions.
- Independently, generate a random set X which contains item j with probability p_j.
- Assign $S \setminus T$ to player 1.
- Assign $T \setminus S$ to player 2.
- Divide $S \cap T$ into $S \cap T \setminus X$ for player 1 and $S \cap T \cap X$ for player 2.

Before analysing Algorithm 1, let us introduce a class of utility functions that is more general than submodular functions (see more details in [18]).

Definition 5.3. *A function w is* fractionally subadditive *if $w(S) \leq \sum \alpha_i w(T_i)$ with $0 \leq \alpha_i \leq 1$ for all i, whenever the following condition holds: for every item $j \in S$, $\sum_{i|j \in T_i} \alpha_i \geq 1$. (Namely, if the sets T_i form a "fractional cover" of S, then the sum of their utilities weighted by the corresponding coefficients is at least as large as that of S.)*

The key to the analysis of Algorithm 1 is the following lemma stated in [18].

Lemma 5.4. *Let $p \in [0,1]$ and $w : 2^{[m]} \rightarrow \mathbb{R}_+$ fractionally subadditive. For a set S, consider a probability distribution over subsets $S' \subseteq S$ such that each element of S is included in S' with probability at least p. Then*

$$\mathbf{E}[w(S')] \geq p\ w(S).$$

Now consider Algorithm 1 from the point of view of player 1, conditioned on a specific choice of S. He receives each element of S, unless it also appears in $T \cap X$. This set is sampled independently of S and

$$\Pr[j \in T \cap X] = q_j p_j \leq \frac{1}{4}$$

because $p_j + q_j \leq 1$. Therefore, conditioned on S, each element is taken away with probability at most 1/4. By Lemma 5.4,

$$\mathbf{E}[w_1(S \setminus (T \cap X)) \mid S] \geq \frac{3}{4} w_1(S).$$

Taking the expectation over S, we get

$$\mathbf{E}[w_1(S \setminus (T \cap X))] \geq \frac{3}{4}\mathbf{E}[w_1(S)] = \frac{3}{4}\sum_S x_{1,S} w_1(S).$$

Similarly, player 2 gets at least $\frac{3}{4}\sum_T x_{2,T} w_2(T)$, since any element appears in $S \cap \overline{X}$ with probability $p_j(1-p_j) \leq 1/4$. This shows that in expectation, we not only recover at least 3/4 of the optimum, but each player individually obtains at least 3/4 of his share in the LP.

5.2.2 Examples and integrality gaps

The 3/4-approximation algorithm for two players is optimal for fractionally subadditive utility functions, in the sense that our LP can have an integrality gap equal to 3/4. The proof is a simple example with 4 items which we present here. As shown in [18], the class of fractionally subadditive functions is equal to "XOS", the class of functions obtained as a maximum of several linear functions. We use this property here to define our example.

Example 5.5 (3/4 integrality gap for two players with XOS functions).

	T_1	T_2
S_1	a	b
S_2	c	d

Consider 4 items $\{a,b,c,d\}$ partitioned in two ways: $S_1 = \{a,b\}, S_2 = \{c,d\}$ and $T_1 = \{a,c\}, T_2 = \{a,d\}$. For a set of items $A \subseteq \{a,b,c,d\}$, we define two utility functions by

- $w_1(A) = \max\{|A \cap S_1|, |A \cap S_2|\}$
- $w_2(A) = \max\{|A \cap T_1|, |A \cap T_2|\}$.

In other words, S_1, S_2 are the sets desired by player 1 and T_1, T_2 are the sets desired by player 2. The optimal LP solution is

$$x_{1,S_1} = x_{1,S_2} = x_{2,T_1} = x_{2,T_2} = \frac{1}{2}$$

which makes each player maximally happy and yields a total value of $LP = 4$. On the other hand, there is no integral solution of value 4. Such a solution would require two disjoint sets S_i and T_j of value 2 but no such pair of disjoint sets exists. The optimum integral solution has value $3 = 3/4 \cdot LP$.

The question arises whether 3/4 is also optimal for submodular functions. It can be seen easily that the utility functions above are not submodular - for example $w_1(\{a,c\}) + w_1(\{b,c\}) = 2$ but $w_1(\{c\}) + w_1(\{a,b,c\}) = 3$. The easiest way to make the functions submodular is to increase the value of the diagonal sets $\{b,c\}$ and $\{a,d\}$ to 2.

Example 5.6 (two players with submodular functions). *Consider the items as above, where we also define $Y = \{a,d\}$ and $Z = \{b,c\}$. Each singleton has value* 1 *and any set of at least* 3 *elements has value* 2. *For pairs of items, define the utility function of player 1 as follows:*

$w_1(Y) = w_1(Z) = 2$	$w_1(T_1) = 1$	$w_1(T_2) = 1$
$w_1(S_1) = 2$	a	b
$w_1(S_2) = 2$	c	d

In other words, player 1 wants at least one of items $\{a,c\}$ and at least one of $\{b,d\}$. Symmetrically, player 2 wants at least one of items $\{a,b\}$ and at least one of $\{c,d\}$. Her utility function is $w_2(S_1) = w_2(S_2) = 1$, $w_2(Y) = w_2(Z) = 2$ and $w_2(T_1) = w_2(T_2) = 2$. The functions w_1 and w_2 can be verified to be submodular (being the rank functions of partition matroids). As before, a fractional solution assigning each of the sets S_1, S_2, T_1, T_2 with weight $1/2$ has value $LP = 4$. However, here the integrality gap is equal to 1, since there is an integral solution (Y, Z) of value 4 as well!

This example illustrates a different phenomenon: Any optimal solution *must combine* items from the two sets desired by each player. If we allocate one set from the fractional solution to each player and resolve conflicts arbitrarily, we get only a value of $3/4 \cdot LP$. Moreover, even allocating the remaining item does not help! Regardless of who gets the last item, the objective value is still only $3/4 \cdot LP$. Therefore, we must combine different sets and the only optimal solution uses the two diagonals.

Observe that instead of increasing the value of the diagonals, we could have increased the value of the orthogonal sets (T_1, T_2 for player 1; S_1, S_2 for player 2). This produces submodular functions as well and again the integrality gap is 1. Here, it's enough to allocate for example S_1 to player 1 and the complement S_2 to player 2.

These examples might suggest that there is a chance to recover the full LP value by either taking sets from the fractional solution or "diagonals" as above. However, this is not the case. A linear combination of these two cases gives the following example.

Example 5.7 (5/6 for two players with submodular functions)**.** *Each singleton has value* 1 *and any set of at least* 3 *elements has value* 2*. For pairs of items, define the utility function of player 1 as follows:*

$w_1(Y) = w_1(Z) = \frac{5}{3}$	$w_1(T_1) = \frac{4}{3}$	$w_1(T_2) = \frac{4}{3}$
$w_1(S_1) = 2$	a	b
$w_1(S_2) = 2$	c	d

Symmetrically, the utility function of player 2 is the same except that $w_2(S_1) = w_2(S_2) = 4/3$ *and* $w_2(T_1) = w_2(T_2) = 2$*. This can be verified to be a submodular function.*

As before, a fractional solution assigning each of the sets S_1, S_2, T_1, T_2 with weight $1/2$ has value $LP = 4$. Whereas, any integral solution such as (S_1, S_2), (Y, Z) or (T_1, T_2) has value at most $10/3 = 5/6 \cdot LP$.

There are extensions of these examples to the case of n players. We defer them to Section 5.4.3.

5.2.3 Two players with a half-integral solution

We have seen that the integrality gap for two players with submodular functions can be 5/6. We show that an approximation factor of 5/6 can be achieved in a special case as above, where the optimum fractional solution is *half-integral*: $x_{i,S} \in \{0, \frac{1}{2}, 1\}$.

If there is a variable $x_{i,S} = 1$, we can assign S to player i and all the remaining items to the other player, in which case we recover the LP value without any loss. Therefore, we can assume that there are sets S_1, S_2, T_1, T_2 such that $x_{1,S_1} = x_{1,S_2} = x_{2,T_1} = x_{2,T_2} = 1/2$. Items that appear only in sets requested by one player can be simply assigned to the respective player, which can only improve the approximation factor. So let's assume that each item appears in two sets assigned to different players. I.e, (S_1, S_2) and (T_1, T_2) are two (possibly different) partitions of the set of all items.

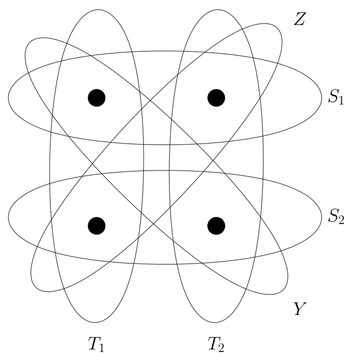

Figure 5.2: A half-integral solution.

It is necessary to allow the option to combine items from the two sets desired by each player, as described in Example 2. Any solution which starts by allocating one set to each player and resolving conflicts cannot beat the factor of 3/4. On the other hand, it is thanks to submodularity that we can extract improved profit by combining two different sets. In analogy with Example 2, we define two "diagonal sets"

- $Y = (S_1 \cap T_1) \cup (S_2 \cap T_2)$.
- $Z = (S_1 \cap T_2) \cup (S_2 \cap T_1)$.

Our algorithm then chooses a random allocation scheme according to the following table:

Probability	1/6	1/6	1/6	1/6	1/6	1/6
Player 1	S_1	S_2	Y	Z	T_1	T_2
Player 2	S_2	S_1	Z	Y	T_2	T_1

We use submodularity to analyze the expected profit of this allocation procedure. For player 1, submodularity yields

$$w_1(Y) + w_1(T_1) \geq w_1(S_1 \cap T_1) + w_1(S_2 \cup T_1)$$

and

$$w_1(Z) + w_1(T_2) \geq w_1(S_1 \cap T_2) + w_1(S_2 \cup T_2).$$

Intuitively, T_1 and T_2 are not the sets desired by player 1 and their values could be as low as $w_1(S_1 \cap T_1)$ and $w_1(S_1 \cap T_2)$. However, if that is the case, submodularity implies that the diagonal sets Y, Z are very desirable for player 1.

Since $(S_1 \cap T_1) \cup (S_1 \cap T_2) = S_1$, we have $w_1(S_1 \cap T_1) + w_1(S_1 \cap T_2) \geq w_1(S_1)$, and by monotonicity $w_1(S_2 \cup T_j) \geq w_1(S_2)$. In total, player 1 gets expected profit

$$\frac{1}{6}(w_1(S_1)+w_1(S_2)+w_1(Y)+w_1(Z)+w_1(T_1)+w_1(T_2)) \geq \frac{1}{6}(2w_1(S_1)+3w_1(S_2))).$$

Alternatively, we can also estimate $w_1(Y)+w_1(T_2) \geq w_1(S_2 \cap T_2)+w_1(S_1 \cup T_2)$ and $w_1(Z) + w_1(T_1) \geq w_1(S_2 \cap T_1) + w_1(S_1 \cup T_1)$ which yields

$$\frac{1}{6}(w_1(S_1)+w_1(S_2)+w_1(Y)+w_1(Z)+w_1(T_1)+w_1(T_2)) \geq \frac{1}{6}(3w_1(S_1)+2w_1(S_2))).$$

By averaging the two estimates, player 1 gets expected profit at least $\frac{5}{12}(w_1(S_1)+w_1(S_2))$. We get a similar estimate for player 2, so the expected value of our integral solution is at least $\frac{5}{6}LP$.

5.2.4 Two players with a balanced fractional solution

Our goal now is to generalize the ideas of the previous section in order to improve the approximation factor of 3/4 for two players in general. First, we relax the condition that the fractional solution is half-integral. Instead, we assume that for any item j,

$$\sum_{S:j \in S} x_{1,S} = \sum_{T:j \in T} x_{2,T} = \frac{1}{2}.$$

We call such a fractional solution *balanced*. Observe that this is the worst case for Algorithm 1. If the fractional solution is significantly unbalanced, then even Algorithm 1 performs better than 3/4, since items are removed from each player with probabilities smaller than 1/4.

So, let's assume for now that the fractional solution is balanced. In particular, we can make this assumption in case both players have the same utility function, since then any solution $x_{i,S}$ can be replaced by $\tilde{x}_{1,S} = \tilde{x}_{2,S} = (x_{1,S} + x_{2,S})/2$ without change of value.

Intuition. Suppose that each player gets value 1 from the fractional solution. Let S denote a random set sampled from the distribution of player 1 and T a random set sample from the distribution of player 2. Due to our assumption of balance, $\Pr[j \in S] = \Pr[j \in T] = 1/2$ for any item j. We can try to allocate S to player 1, and the entire complement of S to player 2. Then player 1 gets expected value 1, while player 2 obtains $\mathbf{E}[w_2(\overline{S})] \geq \mathbf{E}[w_2(T \cap \overline{S})] \geq 1/2$, but this might be tight and we still don't achieve more than 3/4 of the fractional value.

However, this is essentially the only case in which we do not gain compared to 3/4. Assuming that the complement of S is not very valuable for player 2, let's consider another set: $Z = (T \cap S) \cup (T' \cap \overline{S})$ where T, T' are sets sampled independently from the distribution of player 2. (This is analogous to one of the diagonal sets in the previous section.) Linearity of expectation allows us to use submodularity for expected values of random sets just like for values of deterministic sets:

$$\mathbf{E}[w_2(Z)] + \mathbf{E}[w_2(\overline{S})] \geq \mathbf{E}[w_2(Z \cup \overline{S})] + \mathbf{E}[w_2(Z \cap \overline{S})] \geq \mathbf{E}[w_2(T)] + \mathbf{E}[w_2(T' \cap \overline{S})].$$

In other words, if $\overline{S}$ presents no improvement on the average over $T' \cap \overline{S}$, then Z is a set as good as T which is exactly what player 2 would desire. Similarly, let player 1 generate two independent sets S, S'. If he does not gain by taking $\overline{T}$ instead of $S' \cap \overline{T}$, then $Y = (S \cap T) \cup (S' \cap \overline{T})$ is by submodularity just as good as S or S'. Note that each player uses the other player's set to "combine" two of his/her sets. Let's be more specific and define which of the other player's sets is used for this purpose:

- $Y = (S \cap T) \cup (S' \cap \overline{T}), \quad Z = (T \cap S') \cup (T' \cap \overline{S'})$.

Note that unlike the diagonal sets in the previous section, Y and Z are not disjoint here. They are random sets, typically intersecting; however, the punch line is that Y and Z are not independent, and in fact the events

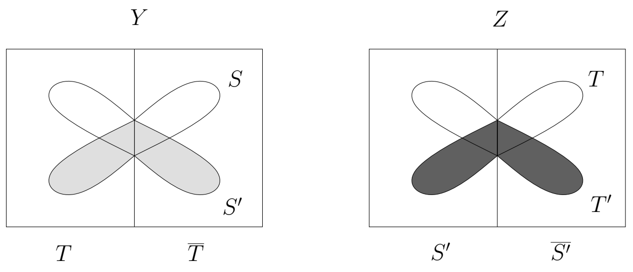

Figure 5.3: Diagonal sets Y and Z.

$j \in Y, j \in Z$ are *negatively correlated*. Observe that Z intersects Y only inside the first part $(S \cap T)$, and more precisely

$$Y \cap Z = (S \cap T) \cap (S' \cup (T' \cap \overline{S'})) = (S \cap T) \cap (S' \cup T').$$

The sets S, S', T, T' are sampled independently and contain each item with probability $1/2$. Similarly, $\Pr[j \in Y] = \Pr[j \in Z] = 1/2$ for any item j, while

$$\Pr[j \in Y \cap Z] = \Pr[j \in (S \cap T) \cap (S' \cup T')] = \frac{3}{16}$$

rather than $1/4$ which is the probability of appearance in $S \cap T$. Thus the interests of the two players are closer to being disjoint and we are able to allocate more items to each of them, using Y and Z. Our next algorithm takes advantage of this fact, combining the two allocation schemes outlined above.

Algorithm 2. (37/48-approximation for 2 balanced players)

- Let player 1 sample independently random sets S, S'.
- Let player 2 sample independently random sets T, T'.
- Let X contain each item independently with probability $1/2$.
- Let $Y = (S \cap T) \cup (S' \cap \overline{T})$, $\quad Z = (T \cap S') \cup (T' \cap \overline{S'})$.

We assign items randomly based on the following table:

Probability	1/3	1/3	1/3
Player 1	S'	$\overline{T}$	$Y \setminus (Z \cap X)$
Player 2	$\overline{S'}$	T	$Z \setminus (Y \cap \overline{X})$

Theorem 5.8. *For any balanced fractional solution $x_{i,S}$, Algorithm 2 gives expected profit at least* $37/48\ \sum_S x_{i,S}\ w_i(S)$ *to each player i.*

Proof. Consider player 1. The sets S and S' are sampled from the same distribution. $\mathbf{E}[w_1(S)] = \mathbf{E}[w_1(S')] = \sum_S x_{1,S} w_1(S)$ is the share of player 1 in the fractional solution, ideally what we would like player 1 to receive. For the sets allocated to him in the second and third scheme, $\overline{T}$ and $Y \setminus (Z \cap X)$, we use submodularity. We use the fact that $Y \cap Z = (S \cap T) \cap (S' \cup T')$, i.e.

$$Y \setminus (Z \cap X) = ((S \cap T) \setminus ((S' \cup T') \cap X)) \cup (S' \cap \overline{T}).$$

By submodularity,

$$\begin{aligned} w_1(\overline{T}) + w_1(Y \setminus (Z \cap X)) &\geq w_1(((S \cap T) \setminus ((S' \cup T') \cap X)) \cup \overline{T}) + w_1(S' \cap \overline{T}) \\ &\geq w_1(S \setminus ((S' \cup T') \cap X \cap T)) + w_1(S' \cap \overline{T}). \end{aligned}$$

Now we use the linearity of expectation and Lemma 5.4: each item appears in $(S' \cup T') \cap X \cap T$ with probability $3/16$, and in $\overline{T}$ with probability $1/2$. Therefore

$$\mathbf{E}[w_1(\overline{T})] + \mathbf{E}[w_1(Y \setminus (Z \cap X))] \ \geq\ \frac{13}{16}\mathbf{E}[w_1(S)] + \frac{1}{2}\mathbf{E}[w_1(S')] = \frac{21}{16}\mathbf{E}[w_1(S)].$$

The expected profit of player 1 is

$$\frac{1}{3}\mathbf{E}[w_1(S)] + \frac{1}{3}\mathbf{E}[w_1(\overline{T})] + \frac{1}{3}\mathbf{E}[w_1(Y \setminus (Z \cap X))] \geq \frac{37}{48}\mathbf{E}[w_1(S)] = \frac{37}{48}\sum_S x_{i,S}\, w_1(S).$$

For player 2, the analysis is similar, although not exactly symmetric: here, we have

$$Z \setminus (Y \cap \overline{X}) = (T \cap S' \setminus (S \cap \overline{X})) \cup (T' \cap \overline{S'} \setminus (S \cap T \cap \overline{X})).$$

Submodularity yields

$$\begin{aligned} w_2(\overline{S'}) + w_2(Z \setminus (Y \cap \overline{X})) &\geq w_2((T \cap S' \setminus (S \cap \overline{X})) \cup \overline{S'}) + w_2(T' \cap \overline{S'} \setminus (S \cap T \cap \overline{X})) \\ &\geq w_2(T \setminus (S' \cap S \cap \overline{X})) + w_2(T' \cap \overline{S'} \setminus (S \cap T \cap \overline{X})). \end{aligned}$$

Finally, we apply Lemma 5.4:

$$\mathbf{E}[w_2(\overline{S'})] + \mathbf{E}[w_2(Z \setminus (Y \cap \overline{X}))] \geq \frac{7}{8}\mathbf{E}[w_2(T)] + \frac{7}{16}\mathbf{E}[w_2(T')] = \frac{21}{16}\mathbf{E}[w_2(T)]$$

and the rest follows as for player 1. $\square$

For two players with a balanced fractional solution, we also have a slightly improved algorithm which has an approximation factor $7/9 \doteq 0.777$ (as compared to $37/48 \doteq 0.771$). Its analysis is slightly more involved; it uses a new trick, the application of Lemma 5.4 to marginal values of subsets. This trick and the structure of the next algorithm will be useful for our final solution for n players.

Algorithm 2'. (7/9-approximation for 2 balanced players)

- Let player 1 sample independently random sets S, S'.
- Let player 2 sample independently random sets T, T'.
- Let $Y = (S \cap T) \cup (S' \cap \overline{T}), \quad Z = (T \cap S) \cup (T' \cap \overline{S})$.
- Let $Y' = (S \cap T') \cup (S' \cap \overline{T'}), \quad Z' = (T \cap S') \cup (T' \cap \overline{S'})$.

We assign items according to the following table:

Probability	5/18	5/18	4/18	4/18
Player 1	S'	$\overline{T}$	Y'	$Y \setminus Z'$
Player 2	$\overline{S'}$	T	$Z \setminus Y'$	Z'

Theorem 5.9. *For any balanced fractional solution $x_{i,S}$, Algorithm 2' gives expected profit at least $7/9 \sum_S x_{i,S}\ w_i(S)$ to each player i.*

Proof. Let's define $\alpha = \mathbf{E}[w_1(S)]$, $\beta = \mathbf{E}[w_2(T)]$, $\gamma = \mathbf{E}[w_1(\overline{T})]$ and $\delta = \mathbf{E}[w_2(\overline{S})]$. I.e., $\alpha + \beta = \sum_{i,S} x_{i,S}\ w_i(S)$ is the value of the fractional solution, while $\alpha + \delta$ and $\beta + \gamma$ are the expected values achieved by the first two allocation schemes. It remains to estimate the expected profit obtained in the last two schemes. By submodularity and linearity of expectation, we get

$$\mathbf{E}[w_2(Z')] + \mathbf{E}[w_2(\overline{S'})] \geq \mathbf{E}[w_2(T \cup \overline{S'})] + \mathbf{E}[w_2(T' \cap \overline{S'})] \geq \mathbf{E}[w_2(T)] + \frac{1}{2}\mathbf{E}[w_2(T')]$$

using Lemma 5.4. I.e.,

$$\mathbf{E}[w_2(Z')] \geq \frac{3}{2}\beta - \delta.$$

It is a little bit more complicated to estimate the value of $Y \setminus Z'$. We use the fact that $Y \setminus Z'$ simplifies to $(S \cap T) \setminus (S' \cup T') \cup (S' \cap \overline{T})$, and then by submodularity:

$$\begin{aligned}
\mathbf{E}[w_1(Y \setminus Z')] + \mathbf{E}[w_1(\overline{T})] &= \mathbf{E}[w_1(((S \cap T) \setminus (S' \cup T')) \cup (S' \cap \overline{T}))] + \mathbf{E}[w_1(\overline{T})] \\
&\geq \mathbf{E}[w_1(((S \cap T) \setminus (S' \cup T')) \cup \overline{T})] + \mathbf{E}[w_1(S' \cap \overline{T})] \\
&\geq \mathbf{E}[w_1(((S \cap T) \setminus (S' \cup T')) \cup \overline{T})] + \frac{1}{2}\alpha.
\end{aligned}$$

To estimate the last expectation, let's consider $g(A) = w_1(A \cup \overline{T}) - w_1(\overline{T})$, a function on subsets $A \subseteq T$ which gives the marginal value of adding some items to $\overline{T}$. This is also a submodular function and therefore Lemma 5.4 applies to it; in particular, conditioning on S, T, we get

$$\mathbf{E}_{S',T'}[g((S \cap T) \setminus (S' \cup T')) \mid S, T] \geq \frac{1}{4} g(S \cap T)$$

because $S' \cup T'$ contains each element with probability 3/4. This means,

$$\begin{aligned} &\mathbf{E}[w_1(((S \cap T) \setminus (S' \cup T')) \cup \overline{T}) - w_1(\overline{T}) \mid S, T] \\ \geq\ &\frac{1}{4}(w_1((S \cap T) \cup \overline{T}) - w_1(\overline{T})) \geq \frac{1}{4}(w_1(S) - w_1(\overline{T})), \end{aligned}$$

$$\begin{aligned} &\mathbf{E}[w_1(((S \cap T) \setminus (S' \cup T')) \cup \overline{T})) - w_1(\overline{T})] \\ &\qquad \geq \frac{1}{4}\mathbf{E}[w_1(S) - w_1(\overline{T})] = \frac{1}{4}(\alpha - \gamma) \end{aligned}$$

and therefore

$$\mathbf{E}[w_1(Y \setminus Z')] \geq \frac{1}{4}(\alpha - \gamma) + \frac{1}{2}\alpha = \frac{3}{4}\alpha - \frac{1}{4}\gamma.$$

By symmetry, we also get $\mathbf{E}[w_1(Y')] \geq \frac{3}{2}\alpha - \gamma$ and $\mathbf{E}[w_2(Z \setminus Y')] \geq \frac{3}{4}\beta - \frac{1}{4}\delta$. By combining all of the above, we obtain that the total expected profit of player 1 is

$$\begin{aligned} &\frac{5}{18}\mathbf{E}[w_1(S)] + \frac{5}{18}\mathbf{E}[w_1(\overline{T})] + \frac{4}{18}\mathbf{E}[w_1(Y \setminus Z')] + \frac{4}{18}\mathbf{E}[w_1(Y')] \\ &\geq \frac{5}{18}\alpha + \frac{5}{18}\gamma + \frac{4}{18}\left(\frac{3}{4}\alpha - \frac{1}{4}\gamma\right) + \frac{4}{18}\left(\frac{3}{2}\alpha - \gamma\right) = \frac{7}{9}\alpha. \end{aligned}$$

By symmetry, the expected profit of player 2 is at least $\frac{7}{9}\beta$. □

5.2.5 Two players with an arbitrary fractional solution

Given our algorithm for balanced fractional solutions, it seems plausible that we should be able to obtain an improvement in the general case as well. This is because Algorithm 1 gives a better approximation than 3/4 if the fractional solution is unbalanced. However, a fractional solution can be balanced on some items and unbalanced on others, so we have to analyze our profit more carefully, item by item. For this purpose, we prove the following generalization of Lemma 5.4.

Lemma 5.10. *Fix an ordering of items $[m] = \{1, 2, \ldots, m\}$; we denote by $[j] = \{1, 2, \ldots, j\}$ the first j items in this ordering. Let S and X be random subsets of $[m]$ such that conditioned on any S, $\Pr[j \in X \mid S] \geq p_j$. Let w be a monotone submodular function and define*

$$\sigma_j = \mathbf{E}[w(S \cap [j]) - w(S \cap [j-1])].$$

Then

$$\mathbf{E}[w(S \cap X)] \geq \sum_{j=1}^{m} p_j \sigma_j.$$

This implies Lemma 5.4 as a special case, since we can have X contain each item with probability $p_j = p$ and then $\mathbf{E}[w(S')] \geq \mathbf{E}[w(S \cap X)] \geq \sum_j p_j \sigma_j = p\ w(S)$.

Proof. Using the marginal value definition of submodularity, we obtain

$$\begin{aligned} w(S \cap X) &= \sum_{j=1}^{m} (w(S \cap X \cap [j]) - w(S \cap X \cap [j-1])) \\ &\geq \sum_{j=1}^{m} (w((S \cap [j-1]) \cup (S \cap X \cap \{j\})) - w(S \cap [j-1])). \end{aligned}$$

Conditioned on S, j appears in X with probability at least p_j, so taking expectation over X yields

$$\mathbf{E}_X[w(S \cap X) \mid S] \geq \sum_{j=1}^{m} p_j (w(S \cap [j]) - w(S \cap [j-1]))$$

and finally

$$\mathbf{E}[w(S \cap X)] \geq \sum_{j=1}^{m} p_j \mathbf{E}[w(S \cap [j]) - w(S \cap [j-1])] = \sum_{j=1}^{m} p_j \sigma_j.$$

□

In the following, we assume that S is a set sampled by player 1, T is a set sampled by player 2, and we set

- $p_j = \Pr[j \in S]$, $\sigma_j = \mathbf{E}[w_1(S \cap [j]) - w_1(S \cap [j-1])]$.
- $q_j = \Pr[j \in T]$, $\tau_j = \mathbf{E}[w_2(T \cap [j]) - w_2(T \cap [j-1])]$.

We seek to estimate our profit in terms of $\mathbf{E}[w_1(S)] = \sum_j \sigma_j$ and $\mathbf{E}[w_2(T)] = \sum_j \tau_j$ which are the shares of the two players in the fractional solution.

First, let's give a sketch of an argument that a strict improvement over 3/4 is possible in the general case. Let's choose a very small constant $\epsilon > 0$ and define "unbalanced items" by $U = \{j : |p_j - 1/2| > \epsilon\}$. We distinguish two cases:

- If a non-negligible value comes from unbalanced items, e.g. $\sum_{j \in U}(\sigma_j + \tau_j) \geq \epsilon \cdot LP$, then we use Algorithm 1. A refined analysis using Lemma 5.10 implies than the algorithm yields profit at least

$$\begin{aligned} &\textstyle\sum_j (1 - p_j(1-p_j))\sigma_j + \sum_j (1 - p_j(1-p_j))\tau_j \\ &\geq \tfrac{3}{4}LP + \textstyle\sum_{j \in U}(\tfrac{1}{4} - p_j(1-p_j))(\sigma_j + \tau_j) \geq (\tfrac{3}{4} + \epsilon^3)LP. \end{aligned}$$

- If the contribution of unbalanced items is negligible, $\sum_{j \in U}(\sigma_j + \tau_j) < \epsilon \cdot LP$, then let's remove the unbalanced items. Also, we scale the remaining fractional solution by $1/(1+2\epsilon)$ and possibly extend some sets so that we get a balanced solution with $p_j = q_j = 1/2$. We incur at most a factor of $(1 - 3\epsilon)$ compared to the original fractional solution. Then we run Algorithm 2 on this balanced fractional solution which yields expected value at least $\frac{37}{48}(1-3\epsilon) \cdot LP$.

Choosing for example $\epsilon = 1/150$ gives an approximation factor slightly better than 3/4. However, we can do better than this, by analyzing more carefully how Algorithm 2 performs on unbalanced items. For balanced items ($p_j = 1/2$) Algorithm 1 gives factor 3/4, while Algorithm 2 gives factor 37/48. On the other hand, observe that items which are extremely unbalanced ($p_j \to 0$) are recovered by Algorithm 1 with probability almost 1, whereas Algorithm 2 recovers them with probability only 2/3. The best we can hope for (by combining these two algorithms) is to take a convex combination which optimizes the minimum of these two cases. This convex combination takes Algorithm 1 with probability 5/17 and Algorithm 2 with probability 12/17 which yields a factor of 13/17 in the two extreme cases. We show that this is indeed possible in the entire range of probabilities p_j. Since Algorithm 2 turns out to favor the player whose share (p_j or q_j) is higher than 1/2, we offset this advantage by generating a splitting set X which gives more advantage to the player with a smaller share.

Algorithm 3. (13/17-approximation for 2 players)

- Let player 1 sample independently random sets S, S'.

- Let player 2 sample independently random sets T, T'.
- Let
$$f(x) = \frac{4x}{(1-x)(9-4x(1-x))}.$$
Generate independently a random set X containing item j with probability $\phi(p_j) = f(p_j)$ for $p_j \leq 1/2$, or with probability $\phi(p_j) = 1 - f(1 - p_j)$ for $p_j > 1/2$.
- Let $Y = (S \cap T) \cup (S' \cup \overline{T}), \quad Z = (T \cap S') \cup (T' \cap \overline{S'})$.

We assign items randomly based on the following table:

Probability	5/17	4/17	4/17	4/17
Player 1	$S \setminus (T \cap X)$	S'	$\overline{T}$	$Y \setminus (Z \cap X)$
Player 2	$T \setminus (S \cap \overline{X})$	$\overline{S'}$	T	$Z \setminus (Y \cap \overline{X})$

Theorem 5.11. *For 2 players with an arbitrary fractional solution, Algorithm 3 yields expected profit at least* $13/17 \sum_S x_{i,S}\ w_i(S)$ *for player* i.

Proof. By definition, we have $\mathbf{E}[w_1(S)] = \mathbf{E}[w_1(S')] = \sum_j \sigma_j$ and $\mathbf{E}[w_2(T)] = \mathbf{E}[w_2(T')] = \sum_j \tau_j$. Now consider the first allocation scheme. The definition of set X is symmetric in the sense that $\phi(1-p_j) = 1-\phi(p_j)$, i.e. if $p_j+q_j = 1$, the set is generated equivalently from the point of view of either player. Since item j appears in $T \cap X$ with probability $q_j\phi(p_j) \leq (1-p_j)\phi(p_j)$, and in $S \cap \overline{X}$ with probability $p_j(1-\phi(p_j)) = p_j\phi(1-p_j)$, Lemma 5.10 implies

$$\begin{aligned}
\mathbf{E}[w_1(S \setminus (T \cap X))] &\geq \sum_j (1-(1-p_j)\phi(p_j))\sigma_j, \\
\mathbf{E}[w_2(T \setminus (S \cap \overline{X}))] &\geq \sum_j (1-p_j\phi(1-p_j))\tau_j.
\end{aligned}$$

To estimate the combined profit of the remaining allocation schemes, we use submodularity and Lemma 5.10:

$$\begin{aligned}
& \mathbf{E}[w_1(Y \setminus (Z \cap X))] + \mathbf{E}[w_1(\overline{T})] \\
\geq\ & \mathbf{E}[w_1((S \cap T) \setminus ((S' \cup T') \cap X) \cup \overline{T})] + \mathbf{E}[w_1(S' \cap \overline{T})] \\
\geq\ & \mathbf{E}[w_1(S \setminus (T \cap X \cap (S' \cup T')))] + \mathbf{E}[w_1(S' \cap \overline{T})] \\
\geq\ & \sum_j (1 - q_j\phi(p_j)(1-(1-p_j)(1-q_j)))\sigma_j + \sum_j (1-q_j)\sigma_j \\
\geq\ & \sum_j (1 + p_j - (1-p_j)\phi(p_j)(1-p_j(1-p_j)))\sigma_j.
\end{aligned}$$

The total expected profit of player 1 is:

$$\begin{aligned}
& \frac{5}{17}\mathbf{E}[w_1(S \setminus (T \cap X))] + \frac{4}{17}\mathbf{E}[w_1(S)] + \frac{4}{17}\mathbf{E}[w_1(\overline{T})] + \frac{4}{17}\mathbf{E}[w_1(Y \setminus (Z \cap X))] \\
\geq\ & \frac{5}{17}\sum_j (1 - \phi(p_j)(1-p_j))\sigma_j + \frac{4}{17}\sum_j (2 + p_j - (1-p_j)\phi(p_j)(1 - p_j(1-p_j)))\sigma_j \\
=\ & \frac{13}{17}\sum_j \sigma_j + \frac{1}{17}\sum_j (4p_j - \phi(p_j)(1-p_j)(9 - 4p_j(1-p_j))\sigma_j.
\end{aligned}$$

We show that the last sum is nonnegative. It can be verified that the function

$$f(x) = \frac{4x}{(1-x)(9-4x(1-x))}$$

is increasing and convex on the interval $(0,1)$. Also, $f(1/2) = 1/2$ and by convexity $f(p_j) + f(1-p_j) \geq 1$. We have $\phi(p_j) = f(p_j)$ for $p_j \in [0, \frac{1}{2}]$ and $\phi(p_j) = 1 - f(1-p_j)$ for $p_j \in [\frac{1}{2}, 1]$; i.e., $\phi(p_j) \leq f(p_j)$ for any $p_j \in (0,1)$. Consequently, $\phi(p_j)(1-p_j)(9-4p_j(1-p_j)) \leq 4p_j$ and so player 1 gets expected profit at least $\frac{13}{17}\sum_j \sigma_j$.

For player 2, we get the following:

$$\begin{aligned}
& \mathbf{E}[w_2(Z \setminus (Y \cap \overline{X}))] + \mathbf{E}[w_2(\overline{S'})] \\
\geq\ & \mathbf{E}[w_2(((S' \cap T) \setminus (Y \cap \overline{X}))) \cup \overline{S'}] + \mathbf{E}[w_2((T' \cap \overline{S'}) \setminus (Y \cap \overline{X}))] \\
\geq\ & \mathbf{E}[w_2(T \setminus (S \cap S' \cap \overline{X}))] + \mathbf{E}[w_2(T' \cap \overline{S'} \setminus (S \cap T \cap \overline{X}))] \\
\geq\ & \sum_j (1 - \phi(1-p_j)p_j^2)\tau_j + \sum_j (1-p_j)(1 - \phi(1-p_j)p_j q_j)\tau_j \\
\geq\ & \sum_j (1 + (1-p_j) - \phi(1-p_j)p_j(1 - p_j(1-p_j)))\tau_j.
\end{aligned}$$

This is just like the expression for player 1 after substituting $p_j \to 1 - p_j$. The same analysis gives that the total expected profit of player 2 is at least $\frac{13}{17}\sum_j \tau_j$. □

5.3 The allocation problem for n players

As we discussed, our final algorithm will use a combination of two rounding techniques.

5.3.1 The Fair Rounding Technique

The first technique can be seen as an alternative way to achieve a factor of $1 - 1/e$. This is a refinement and simplification of the previously used techniques for fractionally subadditive utility functions [18]. Compared to previous approaches, it has several advantages; the most important for us now being that if the fractional solution is significantly unbalanced (some items are allocated to some players with a significantly large probability), we already gain over $1 - 1/e$. This rounding procedure is based on a technique that we call *Fair Contention Resolution.*

Fair Contention Resolution. Suppose n players compete for an item independently with probabilities $p_1, p_2, \ldots, p_n$. Denote by A the random set of players who request the item, i.e. $\Pr[i \in A] = p_i$ independently for each i.

- If $A = \emptyset$, do not allocate the item.
- If $A = \{k\}$, allocate the item to player k.
- If $|A| > 1$, allocate the item to each $k \in A$ with probability

$$r_{A,k} = \frac{1}{\sum_{i=1}^{n} p_i} \left(\sum_{i \in A \setminus \{k\}} \frac{p_i}{|A| - 1} + \sum_{i \notin A} \frac{p_i}{|A|} \right).$$

It can be seen that $r_{A,k} \geq 0$ and $\sum_{k \in A} r_{A,k} = 1$ so this is a valid probability distribution.

Lemma 5.12. *Conditioned on player k requesting the item, she obtains it with probability exactly*

$$\rho = \frac{1 - \prod_{i=1}^{n}(1 - p_i)}{\sum_{i=1}^{n} p_i}.$$

This is the best possible value of ρ, since the total probability that the item is allocated should be $\rho \sum_i p_i$ and this cannot be higher than the probability that somebody requests the item, $1 - \prod_i (1 - p_i)$.

Proof. First, suppose that we allocate the item to player k with probability $r_{A,k}$ for any A containing k, even $A = \{k\}$. (For the sake of the proof, we interpret the sum over $A \setminus \{k\}$ for $A = \{k\}$ as zero, although the summand is undefined.) Then the conditional probability that player k receives the item, when she competes for it, would be $\mathbf{E}[r_{A,k} \mid k \in A]$.

However, when $A = \{k\}$, our technique actually allocates the item to player k with probability 1, rather than

$$r_{\{k\},k} = \frac{\sum_{i \neq k} p_i}{\sum_{i=1}^{n} p_i} = 1 - \frac{p_k}{\sum_{i=1}^{n} p_i}.$$

So player k gains an additional probability $\Pr[A=\{k\}](1-r_{\{k\},k}) = \Pr[A=\{k\}]\ p_k/\sum_i p_i$ which makes the total probability that player k obtains the item equal to

$$q_k = p_k\ \mathbf{E}[r_{A,k} \mid k \in A] + \frac{p_k}{\sum_{i=1}^n p_i}\Pr[A=\{k\}]. \tag{5.1}$$

We would like to show that $q_k = \frac{p_k}{\sum p_i}\Pr[A \neq \emptyset]$. This means that $\mathbf{E}[r_{A,k} \mid k \in A]$ should be equal to $\frac{1}{\sum p_i}\Pr[A \setminus \{k\} \neq \emptyset]$. Let's define $B = [n] \setminus \{k\}$ and let $A' = A \setminus \{k\}$ be the set of players competing with k. The probability of a particular set A' occurring is $p(A') = \prod_{i \in A'} p_i \prod_{i \in B \setminus A'}(1-p_i)$. Let's write $\mathbf{E}[r_{A,k} \mid k \in A]$ as a weighted sum over all possible subsets $A' \subseteq B$:

$$\begin{aligned}\mathbf{E}[r_{A,k} \mid k \in A] &= \sum_{A' \subseteq B} p(A')\ r_{A' \cup \{k\},k} \\ &= \frac{1}{\sum_{i=1}^n p_i}\sum_{A' \subseteq B} p(A')\left(\sum_{i \in A'}\frac{p_i}{|A'|} + \sum_{i \in B \setminus A'}\frac{p_i}{|A'|+1}\right).\end{aligned}$$

Ideally, we would like to see $\frac{1}{\sum p_i}\sum_{A' \neq \emptyset} p(A') = \frac{1}{\sum p_i}\Pr[A' \neq \emptyset]$ instead, but we have to perform a certain redistribution of terms to achieve this. Observe that for i, A' such that $i \in B \setminus A'$, the contribution can be also written as

$$p(A')\frac{p_i}{|A'|+1} = \frac{1-p_i}{p_i}p(A' \cup \{i\})\frac{p_i}{|A'|+1} = p(A' \cup \{i\})\frac{1-p_i}{|A' \cup \{i\}|}.$$

Using this equality to replace all the terms for $i \in B \setminus A'$, we get

$$\begin{aligned}\mathbf{E}[r_{A,k} \mid k \in A] &= \frac{1}{\sum_{i=1}^n p_i}\left(\sum_{A' \subseteq B} p(A')\sum_{i \in A'}\frac{p_i}{|A'|} + \sum_{A' \subseteq B}\sum_{i \in B \setminus A'} p(A' \cup \{i\})\frac{1-p_i}{|A' \cup \{i\}|}\right) \\ &= \frac{1}{\sum_{i=1}^n p_i}\left(\sum_{A' \subseteq B} p(A')\sum_{i \in A'}\frac{p_i}{|A'|} + \sum_{\emptyset \neq A'' \subseteq B}\sum_{i \in A''} p(A'')\frac{1-p_i}{|A''|}\right) \\ &= \frac{1}{\sum_{i=1}^n p_i}\sum_{\emptyset \neq A' \subseteq B} p(A')\sum_{i \in A'}\left(\frac{p_i}{|A'|} + \frac{1-p_i}{|A'|}\right) \\ &= \frac{1}{\sum_{i=1}^n p_i}\sum_{\emptyset \neq A' \subseteq B} p(A') = \frac{1}{\sum_{i=1}^n p_i}\Pr[A \setminus \{k\} \neq \emptyset].\end{aligned}$$

So indeed, from (5.1), player k receives the item with probability

$$q_k = \frac{p_k}{\sum_{i=1}^n p_i}(\Pr[A \setminus \{k\} \neq \emptyset] + \Pr[A=\{k\}]) = \frac{p_k}{\sum_{i=1}^n p_i}\Pr[A \neq \emptyset].$$

□

This gives immediately an approximation factor of $1-(1-1/n)^n > 1-1/e$, assuming only fractional additivity.

Algorithm 4. (Fair Rounding Procedure)

- Let each player i sample a set S_i from his/her probability distribution.
- Using the Fair Contention Resolution technique, obtain disjoint sets S_i^* and allocate them to the respective players.

Lemma 5.13. *For n players with fractionally subadditive utility functions, Algorithm 4 delivers expected value at least $(1-(1-1/n)^n)\sum_S x_{i,S}\, w_i(S)$ to player i.*

Proof. Each player requests a set of expected value $\mathbf{E}[w_i(S_i)] = \sum_S x_{i,S}\, w_i(S)$. Define

- $y_{ij} = \sum_{S:j\in S} x_{i,S} = \Pr[j \in S_i]$

i.e., the probability that player i competes for item j. By Lemma 5.12, conditioned on player i competing for item j, the item is allocated to him/her with probability

$$\rho = \frac{1-\prod_{i=1}^n (1-y_{ij})}{\sum_{i=1}^n y_{ij}} \geq 1-\left(1-\frac{1}{n}\right)^n$$

since $\sum_{i=1}^n y_{ij} \leq 1$. This is done independently of the particular set S_i containing j that the player has chosen. Therefore, conditioned on any particular chosen set S_i, the player obtains each of its items with probability at least $1-(1-1/n)^n$. The result follows by Lemma 5.4. □

This already improves the approximation factor of $1-1/e$ for any fixed number of players. However, our goal is to obtain an absolute constant larger than $1-1/e$, independent of n. We will also show that the Fair Rounding Procedure achieves a factor better than $1-1/e$ whenever the fractional solution is *unbalanced*, similarly to the case of two players. However, the most difficult case to deal with is when the fractional solution is *balanced*.

5.3.2 The Butterfly Rounding Technique

Let's assume for now that the fractional solution is balanced in the sense that for each player i and item j, we have $y_{ij} = \sum_{S:j\in S} x_{i,S} = 1/n$[1], which

[1]There is always a balanced optimal LP solution when all players have the same utility function. Hence the results in this section can be applied directly to this special case.

is the worst case for the Fair Rounding Procedure. We also assume that the number of players n is very large; otherwise, we have an improvement over $1 - 1/e$ already. In other words, we consider the variables y_{ij} infinitesimal and we write $(1-1/n)^n \doteq e^{-1}$, $(1-1/n)^{n/2} \doteq e^{-1/2}$, etc.

In this case, we use ideas from the two-player case in order to obtain a small improvement over $1 - 1/e$. Roughly speaking, we divide the players into two groups and treat them as two super-players, using our algorithm for two players. The items obtained by the super-players are allocated within each group. We would have liked to employ Algorithm 3 as a black box for the two super-players, but the additional complication of conflicts inside each group forces us to modify the previous algorithms slightly.

Let us also assume that we can split the players evenly into two groups $\mathcal{A}, \mathcal{B}$ such that for each item j,

$$\sum_{i \in \mathcal{A}} y_{ij} = \sum_{i \in \mathcal{B}} y_{ij} = \frac{1}{2}.$$

For a collection of sets $\{S_i : i \in \mathcal{A}\}$ sampled by players in one group, we will use Lemma 5.12 to make the sets disjoint; we call this "resolving conflicts among a group of players". Recall that players in a group $\mathcal{A}$ such that $\sum_{i \in \mathcal{A}} y_{ij} = 1/2$ can resolve conflicts in such a way that each requested item is allocated with probability $(1 - \prod_{i \in \mathcal{A}}(1 - y_{ij}))/(\sum_{i \in \mathcal{A}} y_{ij}) \geq 2(1 - e^{-1/2}) \doteq 0.787$. This is significantly better than $1 - e^{-1} \doteq 0.632$; however, 0.787 is not the approximation factor we can achieve. First we have to distribute items between the two groups, and for this purpose we use ideas inspired by the two-player case. In the end, we recover roughly 0.645 of the LP value for each player.

Algorithm 5. (Butterfly Rounding Procedure)

- Let each player in group $\mathcal{A}$ sample two independent random sets S_i, S'_i.
- Let each player in group $\mathcal{B}$ sample two independent random sets T_i, T'_i.
- Let $U = \bigcup_{i \in \mathcal{A}} S_i$, $U' = \bigcup_{i \in \mathcal{A}} S'_i$, $V = \bigcup_{i \in \mathcal{B}} T_i$, $V' = \bigcup_{i \in \mathcal{B}} T'_i$.
- Let the players in $\mathcal{A}$ resolve conflicts among S_i to obtain disjoint sets S^*_i. Similarly, resolve conflicts among S'_i to obtain disjoint sets S'^*_i.
- Let the players in $\mathcal{B}$ resolve conflicts among T_i to obtain disjoint sets T^*_i. Similarly, resolve conflicts among T'_i to obtain disjoint sets T'^*_i.
- Let $Y^*_i = (S^*_i \cap V) \cup (S'^*_i \cap \overline{V})$, $Z^*_i = (T^*_i \cap U) \cup (T'^*_i \cap \overline{U})$.

- Let $Y_i'^* = (S_i^* \cap V') \cup (S_i'^* \cap \overline{V'})$, $\quad Z_i'^* = (T_i^* \cap U') \cup (T_i'^* \cap \overline{U'})$.

We assign the items using one of four allocation schemes:

1. With probability $e^{1/2}/(1+2e^{1/2})$:
   Each player $i \in \mathcal{A}$ receives S_i^*. Each player $i \in \mathcal{B}$ receives $(T_i^* \setminus U) \cup (T_i'^* \setminus U \setminus V)$.

2. With probability $e^{1/2}/(1+2e^{1/2})$:
   Each player $i \in \mathcal{B}$ receives T_i^*. Each player $i \in \mathcal{A}$ receives $(S_i^* \setminus V) \cup (S_i'^* \setminus V \setminus U)$.

3. With probability $1/(2+4e^{1/2})$:
   Each player $i \in \mathcal{A}$ receives $Y_i'^*$. Each player $i \in \mathcal{B}$ receives $Z_i^* \setminus \bigcup_{i \in \mathcal{A}} Y_i'^*$.

4. With probability $1/(2+4e^{1/2})$:
   Each player $i \in \mathcal{B}$ receives $Z_i'^*$. Each player $i \in \mathcal{A}$ receives $Y_i^* \setminus \bigcup_{i \in \mathcal{B}} Z_i'^*$.

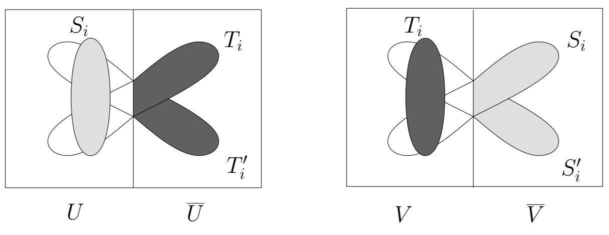

Figure 5.4: Allocation schemes 1 and 2.

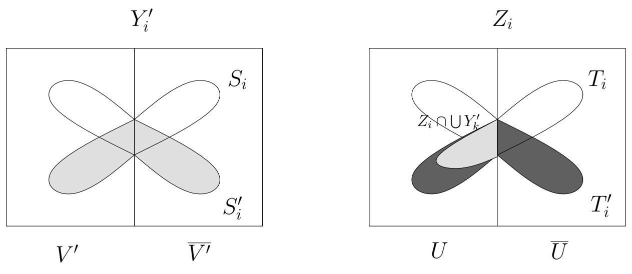

Figure 5.5: Allocation scheme 3.

See the figures depicting the four allocation schemes (without considering conflicts inside each group of players). In the following, we also use $Y_i = (S_i \cap V) \cup (S_i' \cap \overline{V})$, $Y_i' = (S_i \cap V') \cup (S_i' \cap \overline{V'})$, etc., to denote sets before

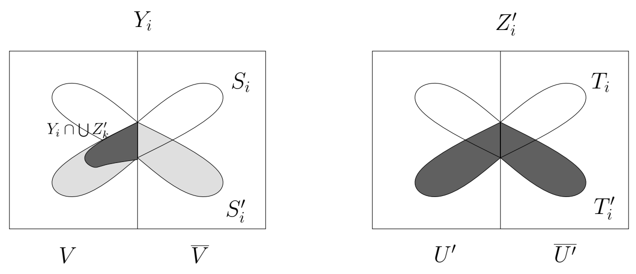

Figure 5.6: Allocation scheme 4.

resolving conflicts. Taking the union over each group, we get the same set regardless of resolving conflicts or not: $\bigcup_{i\in\mathcal{A}} S_i = \bigcup_{i\in\mathcal{A}} S_i^* = U$, $\bigcup_{i\in\mathcal{A}} Y_i = \bigcup_{i\in\mathcal{A}} Y_i^* = (U \cap V) \cup (U' \cap \overline{V})$, etc.

Intuition. If we use only the first two schemes, we get an approximation factor of at least $1 - 1/e$. In fact, we get $1 - 1/e$ even without using the sets S'_i, T'_i - if each player chooses one set and we resolve conflicts first in one preferred group and then in the second group, we get exactly $1 - 1/e$ (see the proof below). Adding some elements of T'_i to a player $i \in \mathcal{B}$ and some elements of S'_i to a player $i \in \mathcal{A}$ might or might not help - but if it doesn't help, we prove that we can construct other good sets Y_i, Y'_i for player $i \in \mathcal{A}$ and Z_i, Z'_i for $i \in \mathcal{B}$, which have the property of negative correlation (we repeat the trick of Algorithm 2). Then we can extract more than $1 - 1/e$ of their value for each player.

Theorem 5.14. *For n players with a balanced fractional solution and $n \to \infty$, Algorithm 5 yields expected profit at least $0.645 \sum_S x_{i,S} w_i(S)$ for player i.*

Proof. We set the following notation:

- For $i \in \mathcal{A}$, $\alpha_i = \mathbf{E}[w_i(S_i)]$. For $i \in \mathcal{B}$, $\beta_i = \mathbf{E}[w_i(T_i)]$.
- For $i \in \mathcal{A}$, $\gamma_i = \mathbf{E}[w_i((S_i \cup S'_i) \setminus V) - w_i(S_i \setminus V)]$. For $i \in \mathcal{B}$, $\delta_i = \mathbf{E}[w_i((T_i \cup T'_i) \setminus U) - w_i(T_i \setminus U)]$.
- For every i, j: $y_{ij} = \sum_{S:j\in S} x_{i,S}$; i.e. $\sum_{i\in\mathcal{A}} y_{ij} = \sum_{i\in\mathcal{B}} y_{ij} = 1/2$.

First, recall that in each group of sets like $\{S_i : i \in \mathcal{A}\}$, Lemma 5.12 allows us to resolve conflicts in such a way that each item in S_i is retained in S_i^* with conditional probability at least $2(1 - e^{-1/2})$. We will postpone this step until the end, which will incur a factor of $2(1 - e^{-1/2})$ on the expected value of all allocated sets. Instead, we analyze the sets "requested" by each player, which are formally obtained by removing the stars from the sets appearing in each allocation scheme. Note that some requested sets are formed by combining S_i and S'_i, such as $Y_i = (S_i \cap V) \cup (S'_i \cap \overline{V})$; however, the resolution of conflicts for each fixed item requested by player i involves only one of the sets S_i, S'_i.

Consider a player $i \in \mathcal{A}$. In the first allocation scheme, he requests the set S_i of expected value $\mathbf{E}[w_i(S_i)] = \alpha_i$. In the second allocation scheme, he requests $(S_i \setminus V) \cup (S'_i \setminus V \setminus U)$. First, let's just consider $S_i \setminus V$. Each item survives outside of V with probability $(1 - 1/n)^{n/2} \doteq e^{-1/2}$, therefore

$$\mathbf{E}[w_i(S_i \setminus V)] \geq e^{-1/2} \alpha_i.$$

Observe that at this point, we already have an approximation factor of $1 - 1/e$: By averaging the two cases, we get $\frac{1}{2}\mathbf{E}[w_i(S_i) + w_i(S_i \setminus V)] \geq \frac{1}{2}(1 + e^{-1/2})\alpha_i$. Player i actually receives each requested item with probability at least $2(1-e^{-1/2})$, so his expected profit is at least $2(1-e^{-1/2}) \cdot \frac{1}{2}(1+e^{-1/2})\alpha_i = (1 - e^{-1})\alpha_i$.

However, rather than $S_i \setminus V$, player i requests $(S_i \setminus V) \cup (S'_i \setminus V \setminus U)$. This might yield some gain or not; we would like to express this gain in terms of γ_i. Let's write $\tilde{U} = \bigcup_{k \in \mathcal{A} \setminus \{i\}} S_k$; we can use this instead of $U = \tilde{U} \cup S_i$ here, since $(S_i \setminus V) \cup (S'_i \setminus V \setminus \tilde{U}) = (S_i \setminus V) \cup (S'_i \setminus V \setminus U)$. The way we analyze the contribution of $S'_i \setminus V \setminus \tilde{U}$ is that we look at the marginal value

$$\gamma'_i = \mathbf{E}[w_i((S_i \setminus V) \cup (S'_i \setminus V \setminus \tilde{U})) - w_i(S_i \setminus V)]$$

as opposed to $\gamma_i = \mathbf{E}[w_i((S_i \setminus V) \cup (S'_i \setminus V)) - w_i(S_i \setminus V)]$. Let's fix S_i, V and define $g_i(A) = w_i((S_i \setminus V) \cup A) - w_i(S_i \setminus V)$, the marginal value of a set added to $S_i \setminus V$. This is also a submodular function. We are interested in $g_i(S'_i \setminus V \setminus \tilde{U})$, as opposed to $g_i(S'_i \setminus V)$. Each item present in $S'_i \setminus V$ survives in $S'_i \setminus V \setminus \tilde{U}$ with conditional probability $\prod_{k \in \mathcal{A} \setminus \{i\}} (1 - y_{kj}) \doteq e^{-1/2}$. Since $\tilde{U}$ is sampled independently of S_i, S'_i and V, Lemma 5.10 implies

$$\mathbf{E}_{\tilde{U}}[g_i(S'_i \setminus V \setminus \tilde{U})] \geq e^{-1/2} g_i(S'_i \setminus V) = e^{-1/2}(w_i((S_i \setminus V) \cup (S'_i \setminus V)) - w_i(S_i \setminus V)).$$

Taking expectation over the remaining random sets, we get

$$\gamma'_i = \mathbf{E}[g_i(S'_i \setminus V \setminus \tilde{U})] \geq e^{-1/2}\mathbf{E}[w_i((S_i \setminus V) \cup (S'_i \setminus V)) - w_i(S_i \setminus V)] = e^{-1/2}\gamma_i.$$

To summarize,

$$\mathbf{E}[w_i((S_i \setminus V) \cup (S'_i \setminus V \setminus U))] = \mathbf{E}[w_i(S_i \setminus V)] + \gamma'_i \geq e^{-1/2}(\alpha_i + \gamma_i).$$

Now, let's turn to the third allocation scheme. Player i requests $Y_i' = (S_i \cap V') \cup (S_i' \cap \overline{V'})$ which by submodularity satisfies

$$\begin{aligned}\mathbf{E}[w_i(Y_i')] + \mathbf{E}[w_i((S_i \cup S_i') \cap \overline{V'})] &\geq \mathbf{E}[w_i(Y_i' \cup (S_i \cap \overline{V'}))] + \mathbf{E}[w_i(S_i' \cap \overline{V'})] \\ &\geq \mathbf{E}[w_i(S_i)] + \mathbf{E}[w_i(S_i' \cap \overline{V'})],\end{aligned}$$

i.e.

$$\mathbf{E}[w_i(Y_i')] \geq \mathbf{E}[w_i(S_i)] - (\mathbf{E}[w_i((S_i \cup S_i') \cap \overline{V'}) - \mathbf{E}[w_i(S_i' \cap \overline{V'})]) = \alpha_i - \gamma_i.$$

Note that either player i gains in the second allocation scheme (when γ_i is large), or otherwise Y_i' has very good expected value, close to α_i.

In the fourth allocation scheme, player i requests $Y_i \setminus \bigcup_{k\in\mathcal{B}} Z_k'$, where $Y_i = (S_i \cap V) \cup (S_i' \cap \overline{V})$ and $\bigcup_{k\in\mathcal{B}} Z_k' = (V \cap U') \cup (V' \cap \overline{U'})$, and so

$$\begin{aligned}Y_i \setminus \bigcup_{k\in\mathcal{B}} Z_k' &= ((S_i \cap V) \cup (S_i' \cap \overline{V})) \setminus ((V \cap U') \cup (V' \cap \overline{U'})) \\ &= ((S_i \cap V) \setminus (U' \cup V')) \cup (S_i' \cap \overline{V}).\end{aligned}$$

By submodularity,

$$\begin{aligned}&\mathbf{E}[w_i(Y_i \setminus \bigcup_{k\in\mathcal{B}} Z_k')] + \mathbf{E}[w_i((S_i \cup S_i') \cap \overline{V})] \\ \geq\ &\mathbf{E}[w_i(S_i \setminus (V \cap (U' \cup V')))] + \mathbf{E}[w_i(S_i' \cap \overline{V})]\end{aligned}$$

and therefore, using $\gamma_i = \mathbf{E}[w_i((S_i \cup S_i') \cap \overline{V})] - \mathbf{E}[w_i(S_i' \cap \overline{V})]$,

$$\begin{aligned}\mathbf{E}[w_i(Y_i \setminus \bigcup_{k\in\mathcal{B}} Z_k')] &\geq \mathbf{E}[w_i(S_i \setminus (V \cap (U' \cup V')))] - \gamma_i \\ &\geq (1 - (1 - e^{-1/2})(1 - e^{-1}))\alpha_i - \gamma_i\end{aligned}$$

since item j appears in $V \cap (U' \cup V')$ with probability $(1 - e^{-1/2})(1 - e^{-1})$. This makes the profit of player i significantly smaller compared to the third allocation scheme; nonetheless, he does not lose as much as if we removed from him a union of independently sampled sets for players in $\mathcal{B}$ (which would contain each element with probability $(1 - e^{-1/2})$ rather than $(1 - e^{-1/2})(1 - e^{-1})$). Here, we benefit from the negative correlation between sets Y_i and Z_k'.

Finally, conflicts are resolved within each group which incurs an additional factor of $2(1 - e^{-1/2})$. The overall expected profit of player $i \in \mathcal{A}$ is

$$\frac{e^{1/2}}{1 + 2e^{1/2}} 2(1 - e^{-1/2}) \left(\alpha_i + e^{-1/2}(\alpha_i + \gamma_i)\right)$$

$$+\frac{1-e^{-1/2}}{1+2e^{1/2}}\left(\alpha_i-\gamma_i+(1-(1-e^{-1/2})(1-e^{-1}))\alpha_i-\gamma_i\right)$$
$$=\frac{1-e^{-1/2}}{1+2e^{1/2}}\left(2e^{1/2}+4-(1-e^{-1/2})(1-e^{-1})\right)\alpha_i \geq 0.645\ \alpha_i.$$

The analysis for a player $i \in \mathcal{B}$ would be exactly the same, yielding expected profit at least 0.645 β_i. □

5.3.3 Sketch of a small improvement in the general case

Let's convince the reader that a tiny (but constant) improvement over $1-1/e$ in the general case is possible. We have the Butterfly Rounding Procedure which requires that players be divided into groups $\mathcal{A}, \mathcal{B}$ with balanced interest in each item, namely

$$\sum_{i\in\mathcal{A}} y_{ij} = \sum_{i\in\mathcal{B}} y_{ij} = \frac{1}{2},$$

and we regard the values y_{ij} as infinitesimal. In fact, the analysis of the Butterfly Rounding Procedure is quite exact, provided that the values y_{ij} are not too large. Also, in this case we can argue that a random partition is likely to be approximately balanced. So, let's propose a variant of the Butterfly Rounding Procedure for the general case.

Algorithm 5'.

- Partition the players into groups $\mathcal{A}, \mathcal{B}$ by assigning each player uniformly and independently to $\mathcal{A}$ or $\mathcal{B}$. Define

$$z_j = \max\left\{\sum_{i\in\mathcal{A}} y_{ij},\ \sum_{i\in\mathcal{B}} y_{ij},\ \frac{1}{2}\right\}.$$

- Now consider the fractional solution as a probability distribution over subsets S for each player. Let X be an independently sampled random set, where item j is present with probability $1/(2z_j)$. We modify the probability distribution of each player by taking $S \cap X$ instead of S. This defines a probability distribution corresponding to a new fractional solution $\tilde{x}_{i,S\cap X}$ where

$$\tilde{x}_{i,S\cap X} = x_{i,S}\Pr[X] = x_{i,S}\prod_{j\in X}\frac{1}{2z_j}\prod_{j\notin X}\left(1-\frac{1}{2z_j}\right).$$

Then, we get $\tilde{y}_{ij} = \sum_{S:j\in S} \tilde{x}_{i,S} = y_{ij}/(2z_j)$ so that $\max(\sum_{i\in\mathcal{A}} \tilde{y}_{ij}, \sum_{i\in\mathcal{B}} \tilde{y}_{ij}) \leq 1/2$.

- Then run Algorithm 5 on the fractional solution $\tilde{x}_{i,S}$. Note that the fractional solution $\tilde{x}_{i,S}$ is "sub-balanced" rather than balanced, but this is sufficient for Algorithm 5 to work properly.

Let's fix a very small $\epsilon > 0$ and call an item "unbalanced" if some player requests it with probability $y_{ij} > \epsilon$. We claim that Algorithm 5' works well for fractional solutions where no item is unbalanced. This is true because then, $\sum_{i\in\mathcal{A}} y_{ij}$ and $\sum_{i\in\mathcal{B}} y_{ij}$ are random variables well concentrated around $\frac{1}{2}\sum_{i=1}^{n} y_{ij}$; more precisely, their variance is

$$Var\left[\sum_{i\in\mathcal{A}} y_{ij}\right] = \sum_{i=1}^{n} \frac{1}{4} y_{ij}^2 \leq \frac{\epsilon}{4} \sum_{i=1}^{n} y_{ij} \leq \frac{\epsilon}{4}.$$

Therefore, the expected amount by which either sum exceeds 1/2 is

$$\mathbf{E}[z_j - 1/2] \leq \mathbf{E}\left[\left|\sum_{i\in\mathcal{A}} y_{ij} - \frac{1}{2}\sum_{i=1}^{n} y_{ij}\right|\right] \leq \sqrt{Var\left[\sum_{i\in\mathcal{A}} y_{ij}\right]} = \frac{1}{2}\sqrt{\epsilon}.$$

The way we obtain the new fractional solution $\tilde{x}_{i,S}$ corresponds to a sampling procedure where each item remains with probability $1/(2z_j)$. Therefore, the expected factor we lose here is

$$\mathbf{E}\left[\frac{1}{2z_j}\right] \geq \frac{1}{2\mathbf{E}[z_j]} \geq \frac{1}{1+\sqrt{\epsilon}}.$$

Moreover, the analysis of Algorithm 5 (which assumed infinitesimal values of y_{ij}) is quite precise for such a fractional solution. For $0 \leq y_{ij} \leq \epsilon$, we have $-y_{ij} \geq \log(1 - y_{ij}) \geq -y_{ij}/(1-\epsilon)$, i.e.

$$e^{-\sum_i y_{ij}} \geq \prod_i (1 - y_{ij}) \geq e^{-\frac{1}{1-\epsilon}\sum_i y_{ij}} \geq (1-2\epsilon)e^{-\sum_i y_{ij}}.$$

Thus all the estimates in the analysis of Algorithm 5 are precise up to an error of $O(\epsilon)$. Accounting for the balancing step, we get a solution of expected value $(0.645 - O(\sqrt{\epsilon}))LP$.

If some items are unbalanced, then running Algorithm 5' might present a problem. However, then we gain by running Algorithm 4. As in Section 5.2.5, we decide which algorithm to use based on the importance of unbalanced items in the fractional solution. Let U denote the set of unbalanced items, and define

$$\sigma_{ij} = \mathbf{E}[w_i(S_i \cap [j]) - w_i(S_i \cap [j-1])],$$

the expected contribution of item j to player i. Then we distinguish two cases:

- If a non-negligible value comes from unbalanced items, e.g. $\sum_i \sum_{j \in U} \sigma_{ij} \geq \epsilon \cdot LP$, then we use Algorithm 4. For each unbalanced item $j \in U$, since $y_{ij} > \epsilon$ for some i, Lemma 5.12 allocates the item to each player with conditional probability

  $$\rho_j \geq 1 - \prod_i (1 - y_{ij}) \geq 1 - \frac{1-\epsilon}{e^{-\epsilon}} e^{-\sum_i y_{ij}} \geq 1 - \left(1 - \frac{1}{2}\epsilon^2\right) e^{-1}.$$

  By Lemma 5.10, the expected value of our solution is at least

  $$\sum_i \sum_j \rho_j \sigma_{ij} \geq \sum_i \sum_j \left(1 - \frac{1}{e}\right) \sigma_{ij} + \sum_i \sum_{j \in U} \frac{1}{2e} \epsilon^2 \sigma_{ij} \geq \left(1 - \frac{1}{e} + \frac{\epsilon^3}{2e}\right) \cdot LP.$$

- If the contribution of unbalanced items is negligible, $\sum_i \sum_{j \in U} \sigma_{ij} < \epsilon \cdot LP$, then let's remove the unbalanced items. This incurs a factor of $(1 - \epsilon)$ on the value of the fractional solution. Then we run Algorithm 5' which yields expected value at least $(0.645 - O(\sqrt{\epsilon})) \cdot LP$.

For a very small $\epsilon > 0$, one of the two algorithms beats $1 - 1/e$ by a positive constant amount. A rough estimate shows that we should choose $\epsilon \doteq 10^{-4}$ in order to keep $0.645 - O(\sqrt{\epsilon})$ above $1 - 1/e$, and then the first case gives an improvement on the order of 10^{-12}.

5.3.4 Our currently best algorithm in the general case

Let's analyze Algorithm 5' more precisely. For each item j, we measure the "granularity" of the fractional solution by $\sum_{i=1}^n y_{ij}^2$, which could range from $1/n$ (for a perfectly balanced solution) to 1 (when only one player requests item j). We show later that the analysis of Algorithm 5 can be carried through for any balanced fractional solution, with error terms depending on the granularity of the solution. But first, let's look at the balancing procedure.

Lemma 5.15. *Let player i have a distribution over sets S_i with expected value $\mathbf{E}[w_i(S_i)] = \sum_j \sigma_{ij}$ where*

$$\sigma_{ij} = \mathbf{E}_{S_i}[w_i(S_i \cap [j]) - w_i(S_i \cap [j-1])].$$

Then after the balancing procedure executed for a given partitioning of players, the new marginal values for player i are

$$\tilde{\sigma}_{ij} = \mathbf{E}_{S_i,X}[w_i(S_i \cap X \cap [j]) - w_i(S_i \cap X \cap [j-1])]$$

such that averaging over random partitions,

$$\mathbf{E}_{(\mathcal{A},\mathcal{B})}[\tilde{\sigma}_{ij}] \geq \frac{\sigma_{ij}}{1 + \sqrt{\sum_{i=1}^n y_{ij}^2}}.$$

Proof. Conditioned on a specific partition $(\mathcal{A}, \mathcal{B})$, the balancing procedure gives to player i a random set $S_i \cap X$ where $\Pr[j \in X] = 1/(2z_j)$. The proof of Lemma 5.10 implies that the new marginal values will be at least $\tilde{\sigma}_{ij} \geq \sigma_{ij}/(2z_j)$. Averaging over random partitions $(\mathcal{A}, \mathcal{B})$, and using $\mathbf{E}[1/z_j] \geq 1/\mathbf{E}[z_j]$, we get

$$\mathbf{E}_{(\mathcal{A},\mathcal{B})}[\tilde{\sigma}_{ij}] \geq \frac{\sigma_{ij}}{2\mathbf{E}[z_j]}.$$

To estimate $\mathbf{E}[z_j]$, we use the second moment. Let Y_{ij} be independent random variables that take values 0 or y_{ij} with probability $1/2$, corresponding to player i being assigned to group $\mathcal{A}$ or $\mathcal{B}$. We have $\sum_{i=1}^n \mathbf{E}[Y_{ij}] = 1/2$ and $z_j = \max(\sum_{i=1}^n Y_{ij}, 1 - \sum_{i=1}^n Y_{ij}) = 1/2 + |\sum_{i=1}^n Y_{ij} - 1/2|$. Due to independence, $Var[\sum_{i=1}^n Y_{ij}] = \sum_{i=1}^n Var[Y_{ij}] = \frac{1}{4}\sum_{i=1}^n y_{ij}^2$ and therefore

$$\mathbf{E}[z_j] = \frac{1}{2} + \mathbf{E}\left[\left|\sum_{i=1}^n Y_{ij} - \frac{1}{2}\right|\right] \leq \frac{1}{2} + \sqrt{Var\left[\sum_{i=1}^n Y_{ij}\right]} = \frac{1}{2} + \frac{1}{2}\sqrt{\sum_{i=1}^n y_{ij}^2}.$$

□

Thus, we are likely to get a good balanced solution for fractional solutions with low granularity. Now we apply Algorithm 5 to this modified fractional solution, and estimate the expected profit. In the analysis, we need the following bounds.

Lemma 5.16. *For any $y_{1j}, \ldots, y_{nj} \geq 0$ such that $\sum_{i=1}^n y_{ij} \leq 1$,*

$$e^{-\sum_{i=1}^n y_{ij}}\left(1 - \sum_{i=1}^n y_{ij}^2\right) \leq \prod_{i=1}^n (1 - y_{ij}) \leq e^{-\sum_{i=1}^n y_{ij}}\left(1 - \frac{1}{2}\sum_{i=1}^n y_{ij}^2\right).$$

Proof. By taking logs and Taylor's series,

$$\log \prod_{i=1}^n (1 - y_{ij}) = \sum_{i=1}^n \log(1 - y_{ij}) = -\sum_{i=1}^n y_{ij} - \frac{1}{2}\sum_{i=1}^n y_{ij}^2 - \frac{1}{3}\sum_{i=1}^n y_{ij}^3 - \ldots,$$

$$\log\left(e^{-\sum_{i=1}^n y_{ij}}\left(1-\sum_{i=1}^n y_{ij}^2\right)\right) = -\sum_{i=1}^n y_{ij} - \sum_{i=1}^n y_{ij}^2 - \frac{1}{2}\left(\sum_{i=1}^n y_{ij}^2\right)^2 - \frac{1}{3}\left(\sum_{i=1}^n y_{ij}^2\right)^3 - \ldots,$$

$$\log\left(e^{-\sum_{i=1}^n y_{ij}}\left(1-\frac{1}{2}\sum_{i=1}^n y_{ij}^2\right)\right) = -\sum_{i=1}^n y_{ij} - \frac{1}{2}\sum_{i=1}^n y_{ij}^2 - \frac{1}{8}\left(\sum_{i=1}^n y_{ij}^2\right)^2 - \frac{1}{24}\left(\sum_{i=1}^n y_{ij}^2\right)^3 - \ldots$$

To compare the first two series, we simply note that

$$\frac{1}{k}\left(\sum y_{ij}^2\right)^k \geq \frac{1}{k}\sum y_{ij}^{2k} \geq \frac{1}{2k}\sum y_{ij}^{2k} + \frac{1}{2k+1}\sum y_{ij}^{2k+1}$$

and so the first inequality in the lemma follows.

To compare the first and the third series, observe that the first two terms match, while starting from the third we have to compare $\sum_i y_{ij}^k$ with $(\sum_i y_{ij}^2)^{k-1}$. We use Hölder's inequality

$$(\sum a_i^p)^{1/p}\,(\sum b_i^q)^{1/q}) \geq \sum a_i b_i$$

with $a_i = y_{ij}^{k/(k-1)}$, $b_i = y_{ij}^{(k-2)/(k-1)}$, $p = k-1$ and $q = (k-1)/(k-2)$:

$$\left(\sum_{i=1}^n y_{ij}^k\right)^{\frac{1}{k-1}}\left(\sum_{i=1}^n y_{ij}\right)^{\frac{k-2}{k-1}} \geq \sum_{i=1}^n y_{ij}^{\frac{k-2}{k-1}} y_{ij}^{\frac{k}{k-1}} = \sum_{i=1}^n y_{ij}^2.$$

Using $\sum_{i=1}^n y_{ij} \leq 1$, we get $\sum_i y_{ij}^k \geq (\sum_i y_{ij}^2)^{k-1}$, and therefore the second inequality holds as well. □

Lemma 5.17. *For n players with an arbitrary fractional solution, the balancing procedure followed by Algorithm 5 gives player i expected profit at least*

$$0.645\sum_j \frac{1-\sum_{i=1}^n y_{ij}^2}{1+\sqrt{\sum_{i=1}^n y_{ij}^2}}\sigma_{ij}.$$

Proof. Let player i have a fractional solution of value $\mathbf{E}[w_i(S_i)] = \sum_j \sigma_{ij}$. We proved that after the balancing procedure, this solution is modified to a new (random) one with marginal values $\tilde{\sigma}_{ij}$ where $\mathbf{E}[\tilde{\sigma}_{ij}] \geq \sigma_{ij}/(1+\sqrt{\sum_{i=1}^n y_{ij}^2})$. We denote the new fractional solution by $\tilde{x}_{i,S}$ and $\tilde{y}_{ij} = \sum_{S:j\in S}\tilde{x}_{i,S}$.

Now we apply Algorithm 5. We have to go through the analysis of Algorithm 5 more carefully, keeping in mind that the solution has finite granularity. We need to be especially cautious about estimates like $\prod(1 - y_{ij}) \geq e^{-\sum y_{ij}}$ which incur an error term depending on the granularity (see

Lemma 5.16). Inequalities like $\prod(1-y_{ij}) \leq e^{-\sum y_{ij}}$ hold without change, as well as $(1-\prod(1-y_{ij}))/\sum y_{ij} \geq 2(1-e^{-1/2})$ for $\sum y_{ij} \leq 1/2$.

In the first allocation scheme, player $i \in \mathcal{A}$ receives

$$\mathbf{E}[w_i(S_i^*)] \geq \sum_j \frac{1-\prod_{i\in\mathcal{A}}(1-\tilde{y}_{ij})}{\sum_{i\in\mathcal{A}}\tilde{y}_{ij}} \tilde{\sigma}_{ij} \geq 2(1-e^{-1/2}) \sum_j \tilde{\sigma}_{ij};$$

here, the inequality goes the right way.

In the second allocation scheme, player $i \in \mathcal{A}$ receives

$$\begin{aligned}\mathbf{E}[w_i((S_i^* \setminus V) \quad \cup \quad (S_i'^* \setminus V \setminus U))] &\geq \sum_j \frac{1-\prod_{i\in\mathcal{A}}(1-\tilde{y}_{ij})}{\sum_{i\in\mathcal{A}}\tilde{y}_{ij}} \prod_{i\in\mathcal{B}}(1-\tilde{y}_{ij})\ \tilde{\sigma}_{ij} + \gamma_i^* \\ &\geq \quad 2e^{-1/2}(1-e^{-1/2}) \left(\sum_j \left(1-\sum_{i=1}^n \tilde{y}_{ij}^2\right) \tilde{\sigma}_{ij} + \gamma_i \right)\end{aligned}$$

where $\gamma_i^* = g_i(S_i'^* \setminus V \setminus U)$ and $g_i(A) = w_i((S_i \setminus V) \cup A) - w_i(S_i \setminus V)$, similarly to the analysis of Algorithm 4. We needed to employ Lemma 5.16 to estimate $\prod_{i\in\mathcal{B}}(1-\tilde{y}_{ij}) \geq e^{-1/2}(1-\sum_{i\in\mathcal{B}}\tilde{y}_{ij}^2)$. The remaining estimates are okay, including $\gamma_i^* \geq 2(1-e^{-1/2})\gamma_i$; since each element of $S_i' \setminus V$ survives in $S_i'^* \setminus V \setminus U$ with probability $\prod_{i\in\mathcal{A}}(1-\tilde{y}_{ij})\ (1-\prod_{i\in\mathcal{A}}(1-\tilde{y}_{ij}))/\sum_{i\in\mathcal{A}}\tilde{y}_{ij} \geq 2e^{-1/2}(1-e^{-1/2})$ for $\sum_{i\in\mathcal{A}}\tilde{y}_{ij} \leq 1/2$, regardless of granularity.

In the third allocation scheme, player $i \in \mathcal{A}$ receives

$$\mathbf{E}[w_i(Y_i'^*)] \geq 2(1-e^{-1/2}) \left(\sum_j \tilde{\sigma}_{ij} - \gamma_i \right)$$

which is valid without any change, since the inequality goes the right way.

In the fourth allocation scheme, we have

$$\begin{aligned}&\mathbf{E}[w_i(Y_i \setminus \bigcup_{k\in\mathcal{B}} Z_k')] \geq \mathbf{E}[w_i(S_i \setminus (V \cap (U' \cup V')))] - \gamma_i \\ \geq \quad & \sum_j \left(1-\left(1-\prod_{i\in\mathcal{A}}(1-\tilde{y}_{ij})\right)\left(1-\prod_{i\in\mathcal{A}\cup\mathcal{B}}(1-\tilde{y}_{ij})\right)\right)\tilde{\sigma}_{ij} - \gamma_i \\ \geq \quad & \sum_j \left(1-\left(1-\left(1-\sum_{i=1}^n \tilde{y}_{ij}^2\right)e^{-1/2}\right)\left(1-\left(1-\sum_{i=1}^n \tilde{y}_{ij}^2\right)e^{-1}\right)\right)\tilde{\sigma}_{ij} - \gamma_i \\ \geq \quad & \sum_j \left(1-\sum_{i=1}^n \tilde{y}_{ij}^2\right)\left(e^{-1/2}+e^{-1}-e^{-3/2}\right)\tilde{\sigma}_{ij} - \gamma_i\end{aligned}$$

using Lemma 5.16 and $\max(\sum_{i\in\mathcal{A}}\tilde{y}_{ij}, \sum_{i\in\mathcal{B}}\tilde{y}_{ij}) \leq 1/2$. Finally, we use Y_i^* instead of Y_i which costs us a factor of $2(1-e^{-1/2})$:

$$\mathbf{E}[w_i(Y_i^*\setminus\bigcup_{k\in\mathcal{B}} Z'_k)] \geq 2(1-e^{-1/2})\left(\sum_j\left(1-\sum_{i=1}^n \tilde{y}_{ij}^2\right)\left(e^{-1/2}+e^{-1}-e^{-3/2}\right)\tilde{\sigma}_{ij}-\gamma_i\right).$$

Observe that compared to the proof of Theorem 5.14, some terms get the error factor $(1-\sum_i \tilde{y}_{ij}^2)$. By taking the appropriate linear combination of the four allocation schemes, player i obtains expected profit at least

$$\frac{1-e^{-1/2}}{1+2e^{1/2}}\left(2e^{1/2}+3+e^{-1/2}+e^{-1}-e^{-3/2})\right)\sum_j\left(1-\sum_{i=1}^n\tilde{y}_{ij}^2\right)\tilde{\sigma}_{ij}$$

$$\geq 0.645\sum_j\left(1-\sum_{i=1}^n\tilde{y}_{ij}^2\right)\tilde{\sigma}_{ij}.$$

This is for a fixed balanced solution with marginal values $\tilde{\sigma}_{ij}$. For a random partition $(\mathcal{A},\mathcal{B})$ and the associated balanced solution, we have $\mathbf{E}[\tilde{\sigma}_{ij}] \geq \sigma_{ij}/(1+\sqrt{\sum_{i=1}^n y_{ij}^2})$. Also, observe that by the balancing procedure, granularity can only decrease: $\sum_{i=1}^n \tilde{y}_{ij}^2 \leq \sum_{i=1}^n y_{ij}^2$, so player i gets at least

$$0.645\sum_j\left(1-\sum_{i=1}^n\tilde{y}_{ij}^2\right)\tilde{\sigma}_{ij} \geq 0.645\sum_j\frac{1-\sum_{i=1}^n y_{ij}^2}{1+\sqrt{\sum_{i=1}^n y_{ij}^2}}\sigma_{ij}.$$

□

This procedure works well for fractional solutions of low granularity. On the other hand, if the granularity is not low, then Algorithm 4 performs better than $1-1/e$. A combination of the two is our final algorithm.

Algorithm 6. ($(1-1/e+0.00007)$-approximation for general n players)

1. With probability 0.99, run Algorithm 4.
2. With probability 0.01, run Algorithm 5'.

Theorem 5.18. *For any fractional solution, Algorithm 6 gives each player expected value at least* $1-1/e+0.00007 \doteq 0.63219$ *of their share in the fractional solution.*

Proof. Consider a fractional solution with σ_{ij} and y_{ij} defined as before. By Lemma 5.12 and 5.16, the first allocation scheme gives each player a set S_i^* of expected value

$$\mathbf{E}[w_i(S_i^*)] \geq \sum_j \left(1 - \prod_{i=1}^n (1 - y_{ij})\right) \sigma_{ij} \geq \sum_j \left(1 - \frac{1}{e}\left(1 - \frac{1}{2}\sum_{i=1}^n y_{ij}^2\right)\right) \sigma_{ij}.$$

By Lemma 5.17, the balancing procedure followed by Algorithm 4 yields an allocation where player i has expected profit at least

$$0.645 \sum_j \left(\frac{1 - \sum_{i=1}^n y_{ij}^2}{1 + \sqrt{\sum_{i=1}^n y_{ij}^2}}\right) \sigma_{ij}.$$

A numerical analysis shows that

$$0.99\left(1 - \frac{1}{e}\left(1 - \frac{1}{2}\sum_{i=1}^n y_{ij}^2\right)\right) + 0.01 \cdot 0.645 \left(\frac{1 - \sum_{i=1}^n y_{ij}^2}{1 + \sqrt{\sum_{i=1}^n y_{ij}^2}}\right) \geq 1 - \frac{1}{e} + 0.00007$$

for any $\sum_i y_{ij}^2 \in [0,1]$ and therefore the overall expected profit of player i is at least

$$\sum_j \left(1 - \frac{1}{e} + 0.00007\right) \sigma_{ij} = \left(1 - \frac{1}{e} + 0.00007\right) \sum_S x_{i,S} w_i(S).$$

□

5.4 Other notes

In this section, we present some related notes and observations which are not essential to our main result (Theorem 5.1). First, we show that a very simple rounding technique already achieves a $(1-1/e)$-approximation. On the other hand, some natural attempts to improve or combine the known $(1 - 1/e)$-approximations based on the Configuration LP are doomed to fail. Hence, it seems that improving the factor of $1 - 1/e$ requires non-trivial techniques.

Finally, we address the question of further possible improvements using the Configuration LP. We present an example with an integrality gap of 0.782 (recall that $1 - 1/e \simeq 0.632$), which shows that a factor better than 0.782 certainly cannot be obtained using the same LP.

5.4.1 A simple $(1-1/e)$-approximation

The Simple Rounding Procedure.

- Let $y_{ij} = \sum_{S:j\in S} x_{i,S}$. For each j, we have $\sum_{i=1}^n y_{ij} \leq 1$.
- Assign each item independently, to player i with probability y_{ij}.

Using the formalism of submodular maximization subject to a matroid constraint, we show that this yields a $(1-1/e)$-approximation. For a submodular function w_i, recall how we define the extension w_i^+ (Section 3.5):

$$w_i^+(y_i) = \max\left\{\sum_S x_{i,S} w_i(S) : \sum_S x_{i,S} \leq 1, x_{i,S} \geq 0 \;\&\; \forall j; \sum_{S:j\in S} x_{i,S} \leq y_{ij}\right\}.$$

Here y_i is the vector with coordinates y_{ij} for $j = 1, \ldots, m$. So the Configuration LP can be written as

$$\begin{aligned} \max \sum_i w_i^+(y_i); \\ \forall j; \sum_i y_{ij} \leq 1, \\ \forall i, j; y_{ij} \geq 0. \end{aligned}$$

and $w_i^+(y_i)$ corresponds to the share player i in the fractional solution. Our Simple Rounding Procedure allocates to player i a random set S_i which is obtained by rounding the coordinates of the vector y_i independently to 0 and 1, based on probabilities y_{ij}. From Lemma 3.7 and Lemma 3.8, we obtain

$$\mathbf{E}[w_i(S_i)] \geq (1-1/e) w_i^+(y_i).$$

In other words, each player receives in expectation at least a $(1-1/e)$-fraction of her LP share.

We remark that we could have used Lemma 4.2 to prove the same result. (And likewise, to analyze the relationship between $F(y)$ and $f^+(y)$ in Section 3.5.) We can imagine that we sample a random subset of S, each element with probability $x_{i,S}$, for each set appearing in the fractional solution, and then allocate the union of the respective subsets $\bigcup_S S(x_{i,S})$ to each player i. (This random assignment is very close to, and in fact dominated by what the Simple Rounding Procedure does.) We would obtain

$$\mathbf{E}\left[w_i\left(\bigcup_S S(x_{i,S})\right)\right] \geq (1-(1-1/m)^m) \sum_S x_{i,S} w_i(S)$$

where m is the number of sets for player i in the fractional solution. This is slightly better than $1 - 1/e$ if the number of items is small; however, this rounding procedure does not achieve factor $1 - (1 - 1/n)^n$ for n players like the Fair Rounding Procedure.

5.4.2 Obstacles to improving $1 - 1/e$

We show here that the Simple Rounding Procedure does not yield a factor better than $1 - 1/e$, and also another natural approach to achieve an improvement fails on the same example.

Example 5.19. *Consider n^n items arranged in an n-dimensional hypercube, $Q_n = \{1, 2, \dots, n\}^n$. For a vector $\vec{y} \in Q_n$, let $F_i(\vec{y})$ denote the "fiber" of $\vec{y}$ in direction i*

$$F_i(\vec{y}) = \{\vec{x} \in Q_n \mid \forall j \neq i; x_j = y_j\},$$

i.e. the set of elements coinciding with y in all coordinates except i. The goal of player i is to obtain at least one item from each fiber in direction i. We define her utility function as

$$w_i(S) = \Pr_y[F_i(\vec{y}) \cap S \neq \emptyset]$$

where $\vec{y} \in Q_n$ is a uniformly random element. This is a monotone submodular function, the probability measure of a union of events indexed by S.

An optimal fractional solution can be defined as follows. The sets desired by player i are layers orthogonal to dimension i:

$$H_{i,j} = \{\vec{x} \in Q_n : x_i = j\}.$$

Each of these sets has value $w_i(H_{i,j}) = 1$. In our fractional solution, player i receives each set $H_{i,j}$ with fractional value $x_{i,H_{i,j}} = 1/n$, for a value of 1.

Consider the Simple Rounding Procedure. It allocates each item independently and uniformly to a random player. For player i, the probability that she receives some item in a fixed fiber is $1 - (1 - 1/n)^n$. Averaging over all fibers, the utility of each player is $1 - (1 - 1/n)^n$ on the average.

However, the actual optimum gives value 1 to each player. This can be achieved by a "chessboard pattern" where item $\vec{x} \in S_n$ is allocated to player $p(x) = \sum_{i=1}^{n} x_i \bmod n$. So our Simple Rounding Procedure gets only $1 - (1 - 1/n)^n$ of the optimum.

As an alternative approach, recall Section 5.1 where we described a generic "three-step" procedure which might improve $1 - 1/e$. Here, each player chooses a random set S with probability $x_{i,S}$. Then, conflicts between players are resolved somehow and finally, the remaining items are allocated. For our fractional solution, this would mean that each player chooses a random layer $H_{i,j}$; we can assume WLOG that she chooses $H_{i,1}$. Regardless of how we resolve conflicts, only $n^n - (n-1)^n = (1-(1-1/n)^n)n^n$ items are allocated at this point, so we cannot get more value than $1-(1-1/n)^n$ per player. But the situation is even worse that this. We still have $(n-1)^n$ unallocated items, but regardless of how we assign them to players, we do not gain any additional value at all! This is because for any of these $(n-1)^n$ remaining items $\vec{x}$ and any player i, the fiber $F_i(\vec{y})$ already has an item in the first layer which was allocated to player i and hence player i is not interested in any more items from this fiber. Thus again, this approach cannot achieve more than $1-(1-1/n)^n$ of the optimum.

5.4.3 Integrality gap of 0.782

Example 5.19 does not actually have any integrality gap. But we can modify this example to achieve an integrality gap of 0.782, thus improving our example of 5/6 for two players (Section 5.2.2).

Example 5.20. *Consider a set of items Q_n as above. For each $i \in [n]$, consider a random set T_i which contains independently one random item from each layer $H_{i,j}$, $j = 1, \ldots, n$. Then we define the utility function of player i as*

$$w_i(S) = \Pr_{T_i}[S \cap T_i \neq \emptyset].$$

Observe that each layer $H_{i,j}$ is still optimal for player i, because the random set T_i intersects it with probability 1. We consider the same fractional solution, $x_{i,H_{i,j}} = 1/n$ for each i, j. This gives value 1 to each player, for a total LP optimum of n.

Lemma 5.21. *The optimal integral solution for Example 5.20 has value $OPT \leq (0.782 + o(1))n$.*

Consider a partition where S_i is the set obtained by player i. Let

$$z_{ij} = \frac{1}{n^{n-1}} |S_i \cap H_{i,j}|,$$

the fraction of layer $H_{i,j}$ obtained by player i. Recall that T_i contains one random element independently from each layer $H_{i,j}$ for $j = 1, \ldots, n$. Hence, we have

$$w_i(S_i) = \Pr_{T_i}[S_i \cap T_i \neq \emptyset] = 1 - \prod_{j=1}^{n} (1 - z_{ij}).$$

What can be a possible assignment of the variables z_{ij}? For a selection of subsets $K_1, K_2, \ldots, K_n \subseteq [n]$, the total number of items in the union of layers $\bigcup_{i=1}^n \bigcup_{j \in K_i} H_{i,j}$ is $n^n - \prod_{i=1}^n (n - |K_i|)$. All the players together cannot obtain more than this many items from the respective layers, which yields

$$\sum_{i=1}^{n} \sum_{j \in K_i} z_{ij} n^{n-1} \leq n^n - \prod_{i=1}^{n} (n - |K_i|).$$

We can also assume that the variables $z_{i1}, z_{i2}, \ldots$ are ordered decreasingly for each i, which makes it sufficient to consider $K_i = \{1, 2, \ldots, k_i\}$. Thus the maximum value of an integral solution is $OPT \leq OPT_1$ where

$$\begin{aligned} OPT_1 \;\; &= \;\; \max \Big\{ n^{n-1} \sum_{i=1}^{n} \left(1 - \prod_{j=1}^{n} (1 - z_{ij}) \right) : \\ \forall\, 1 \leq k_1, k_2, \ldots, k_n \leq n; \quad & \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k_i} z_{ij} \leq 1 - \prod_{i=1}^{n} \left(1 - \frac{k_i}{n} \right) \\ \forall\, 1 \leq i \leq n; \quad & 1 \geq z_{i1} \geq z_{i2} \geq \ldots \geq z_{in} \geq 0 \Big\} \end{aligned}$$

In the following, we estimate OPT_1 from above.

Lemma 5.22. $(1 - \epsilon(n)) OPT_1 \leq OPT_2$ *where* $\epsilon(n) \to 0$ *and*

$$\begin{aligned} OPT_2 \;\; &= \;\; \max \Big\{ n^{n-1} \sum_{i=1}^{n} \left(1 - \prod_{j=1}^{n} (1 - y_{ij}) \right) : \\ \forall\, 1 \leq k_1, k_2, \ldots, k_n \leq n; \quad & \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k_i} y_{ij} \leq 1 - e^{-\sum_{i=1}^{n} k_i / n} \\ \forall\, 1 \leq i \leq n; \quad & 1 \geq y_{i1} \geq y_{i2} \geq \ldots \geq y_{in} \geq 0 \Big\} \end{aligned}$$

Proof. Note that we are replacing the contraints by stronger conditions, so $OPT_1 \geq OPT_2$. But we are more interested in the opposite inequality. Choose an arbitrarily small $\epsilon > 0$ and consider an optimal solution z_{ij} to OPT_1. We modify this to a near-optimal solution z'_{ij} with a bounded sum $\sum_{j=1}^n z'_{ij}$ for each i. Recall that $z_{i1} \geq z_{i2} \geq \ldots$; set $C_\epsilon = \log(1/\epsilon)$ and if $\sum_{j=1}^n z_{ij} > C_\epsilon$, take k_i minimum such that $\sum_{j=1}^{k_i} z_{ij} > C_\epsilon$. Keep only these k_i values, $z'_{ij} = z_{ij}$ for $j \leq k_i$, and set the rest to $z'_{ij} = 0$ (for $j > k_i$). Now,

$$\sum_{j=1}^{n} z'_{ij} \leq C_\epsilon + 1 \leq 2C_\epsilon.$$

On the other hand, since we have $\sum_{j=1}^{n} z'_{ij} > C_\epsilon$ for any i where we modified the solution, and then

$$\prod_{j=1}^{n}(1 - z_{ij}) \leq \prod_{j=1}^{n}(1 - z'_{ij}) \leq e^{-C_\epsilon} = \epsilon,$$

we didn't lose more than ϵn^n in the objective function. We will see that the optimum is $OPT_1 = \Omega(n^n)$, so we lose only $O(\epsilon)OPT_1$.

We claim that for n sufficiently large and any $\ell \leq n$,

$$\frac{1-\epsilon}{n} \sum_{i=1}^{\ell} \sum_{j=1}^{k_i} z'_{ij} \leq 1 - e^{-\sum_{i=1}^{\ell} k_i/n}. \tag{5.2}$$

If this holds, then $y_{ij} = (1-\epsilon) z'_{ij}$ is a feasible solution to OPT_2, which means that $OPT_2 \geq (1 - O(\epsilon))OPT_1$.

Assume WLOG that $1 \leq k_1 \leq k_2 \leq \ldots \leq k_\ell$. If all the k_i's are bounded by a constant, for instance $\frac{4}{\epsilon} C_\epsilon$, then we are done, since then $1 - k_i/n = (1 - O(n^{-2}))e^{-k_i/n}$ and

$$1 - \prod_{i=1}^{\ell}\left(1 - \frac{k_i}{n}\right) = 1 - (1 - O(\ell n^{-2}))e^{-\sum_{i=1}^{\ell} k_i/n} \leq \frac{1}{1-\epsilon}\left(1 - e^{-\sum_{i=1}^{\ell} k_i/n}\right)$$

for a sufficiently large n.

If $k_\ell > \frac{4}{\epsilon} C_\epsilon$, we use induction on ℓ. We assume that (5.2) holds for $\ell - 1$. Going from $\ell - 1$ to ℓ, the right-hand side of (5.2) increases by

$$\left(1 - e^{-\sum_{i=1}^{\ell} k_i/n}\right) - \left(1 - e^{-\sum_{i=1}^{\ell-1} k_i/n}\right) = e^{-\sum_{i=1}^{\ell-1} k_i/n}\left(1 - e^{-k_\ell/n}\right)$$

We can assume that $e^{-\sum_{i=1}^{\ell} k_i/n} \geq \epsilon$, otherwise (5.2) is trivial. Also, we assumed $k_\ell > \frac{4}{\epsilon} C_\epsilon$, so the RHS increases by at least

$$\epsilon(1 - e^{-k_\ell/n}) \geq \epsilon(1 - e^{-1})\frac{k_\ell}{n} \geq \frac{2C_\epsilon}{n}.$$

Meanwhile, the LHS increases by

$$\frac{1-\epsilon}{n} \sum_{j=1}^{k_\ell} z'_{ij} \leq \frac{1-\epsilon}{n} \cdot 2C_\epsilon$$

which proves (5.2). □

Now we estimate OPT_2. The constraints imply that when we order the y_{ij}'s in a decreasing sequence, the sum of the m largest ones is bounded by $n(1 - e^{-m/n})$. We claim that the optimum is attained when this bound is tight for any m, i.e. the m-th largest y_{ij} is equal to $n(e^{-(m-1)/n} - e^{-m/n}) \doteq e^{-m/n}$. This can be seen by considering the first y_{ij}, being the m-th largest, which violates this condition and is smaller than $e^{-m/n}$. Denote by $P_m = \prod_{j' \neq j}(1 - y_{ij'})$ the product for the remaining entries in the i-th row, apart from y_{ij}. Similarly, let y_{kl} be the $(m+1)$-th largest entry and denote by $P_{m+1} = \prod_{l' \neq l}(1 - y_{kl'})$ the product for the remaining entries in its row. If $P_m > P_{m+1}$ then we can switch y_{ij} and y_{kl} and decrease the sum of row products $\prod_j(1 - y_{ij}) + \prod_l(1 - y_{kl})$. If $P_m \leq P_{m+1}$ then we can increase y_{ij} slightly and decrease y_{kl} by the same amount, which again decreases the average of the two row products. In both cases, we increase the objective function.

Thus we know exactly the optimal sequence of entries y_{ij}: the m-th largest one is roughly $e^{-m/n}$. We only have to find their optimal placement in the $n \times n$ matrix y_{ij}. Since the product $\prod_{i,j}(1 - y_{ij})$ is fixed, and we minimize the *sum of the row products* $\sum_i \prod_j (1 - y_{ij})$, we try to make these products as uniform as possible. We claim that $\sum_{i=1}^n \prod_{j=1}^n (1 - y_{ij})$ is minimized under these conditions, when

- There is m such that the first m rows contain $y_{i1} = e^{-i/n}$, and $y_{ij} \doteq 0$ for $j > 1$ (i.e., the smallest possible entries are placed here).
- The remaining entries are distributed in the remaining $n - m$ rows so that their products $\prod_{j=1}^n (1 - y_{ij})$ are all approximately equal to

$$\rho = \left(\prod_{r=m+1}^{n^2 - m(n-1)} \left(1 - e^{-r/n}\right) \right)^{1/(n-m)}.$$

- m is chosen so that $\rho \doteq 1 - e^{-m/n}$.

To see this, denote by M the rows containing the m largest entries (it's easy to see that these should be in m distinct rows). Any of these rows has a product $\prod_{j=1}^n (1 - y_{ij}) \leq 1 - e^{-m/n} = \rho$. Outside of M, consider the row with the largest product $\prod_{j=1}^n (1 - y_{ij})$. By averaging, this must be larger than ρ. If there is any entry in it smaller than some entry in M, switch these two entries and gain in the objective function. Therefore, all the smallest entries must be next to the m largest entries in M.

Outside of M, the optimal way is to make the products $\prod_{j=1}^n (1 - y_{ij})$ as uniform as possible, since we are minimizing their sum. So we make them all approximately equal to ρ. The value of m is chosen so that the last row in M has a product approximately equal to ρ as well.

We find the approximate solution of

$$1 - e^{-m/n} = \left(\prod_{r=m+1}^{n^2 - m(n-1)} \left(1 - e^{-r/n}\right) \right)^{1/(n-m)}$$

by taking logs and replacing a sum by an integral. We substitute $m = \mu n$ and $r = xn$, which yields

$$\log\left(1 - e^{-\mu}\right) = \frac{1}{1-\mu} \int_{\mu}^{\infty} \log\left(1 - e^{-x}\right) dx. \tag{5.3}$$

The numerical solution of this equation is $\mu \doteq 0.292$. The value of our optimal solution is then

$$\begin{aligned} OPT_2 &= n^{n-1} \left(\sum_{i=1}^{m} e^{-i/n} + (n-m) e^{-m/n} \right) \\ &\doteq n^n \left(\int_0^{\mu} e^{-x} dx + (1-\mu) e^{-\mu} \right) = n^n (1 - \mu e^{-\mu}). \end{aligned}$$

Substituting the solution of (5.3) yields $OPT_2 \doteq 0.782\ n^n$.

Chapter 6

The Generalized Assignment Problem

Problem: *Given m items and n bins; for each bin i and item j, there is given value v_{ij} and size s_{ij}. A partition of the items into disjoint sets $S_1, S_2, \ldots, S_n$ is feasible if $\sum_{j \in S_i} s_{ij} \leq 1$ for each bin i. We seek a feasible partition which maximizes $\sum_{i=1}^{n} v_i(S_i)$ where $v_i(S_i) = \sum_{j \in S_i} v_{ij}$.*

This problem has been considered in [22], where a $(1-1/e)$-approximation is given. The approximation algorithm is based on the following LP, very similar to the Configuration LP discussed in Chapter 5.

$$\begin{array}{rc} \max & \sum_i \sum_{S \in \mathcal{F}_i} x_{i,S} \; v_i(S); \\ \forall j; & \sum_i \sum_{S \in \mathcal{F}_i : j \in S} x_{i,S} \leq 1, \\ \forall i; & \sum_{S \in \mathcal{F}_i} x_{i,S} \leq 1, \\ & x_{i,S} \geq 0. \end{array}$$

Here, $\mathcal{F}_i$ denotes the set of all feasible assignments for bin i, i.e. sets such that $\sum_{j \in S_i} s_{ij} \leq 1$. As shown in [22], this LP can be solved to an arbitrary precision, and the fractional solution has polynomial size. Given a fractional solution to the LP, a randomized rounding technique is given in [22], which retrieves at least a $(1 - 1/e)$-fraction of the value of the fractional solution. A proof of NP-hardness of $(1 - 1/e + \epsilon)$-approximation is given in [22] for a more general problem, the Separable Assignment Problem (SAP) where $\mathcal{F}_i$ is an arbitrary down-monotone family. Also, GAP can be seen as a special case of submodular maximization under a matroid constraint (Section 1.2.4), for which a better than $(1-1/e)$-approximation is NP-hard. Still, it was not known whether $1 - 1/e$ is the optimal approximation factor for GAP.

We resolve this question here by showing an improved rounding technique for the same LP which gives a factor strictly better than $1 - 1/e$. Our improvement is of purely theoretical interest (on the order of 10^{-120}). To put things in perspective, the algorithm of [22] actually gives an approximation ratio of $1 - (1 - 1/n)^n \geq 1 - 1/e + 1/(2en)$ where n is the number of bins. Therefore, our main result is relevant only for instances with at least 10^{120} bins.

6.1 Integrality gap for 2 bins

Consider first the Generalized Allocation Problem with 2 bins. It can be seen that the approach of [22] gives a 3/4-approximation here. The following example shows that the approximation factor cannot be improved to more than 4/5, using the same LP.

Example with integrality gap 4/5. Consider 3 items $\{a, b, c\}$ and 2 bins. The following table shows the values and sizes for each bin:

Item	Size for bin 1	Value for bin 1	Size for bin 2	Value for bin 2
a	0.5	1	1.0	2
b	0.5	2	0.5	2
c	1.0	2	0.5	1

The optimal fractional solution is: $x_{1,\{a,b\}} = x_{1,\{c\}} = 1/2$ and $x_{2,\{a\}} = x_{2,\{b,c\}} = 1/2$ which has value 5. However, it is impossible to achieve value 5 integrally; since the item values are integers, this would require one bin to receive a set of value at least 3 but the only feasible set of value 3 for each bin leaves only an item of value 1 for the other bin. Therefore the integral optimum is 4 and the integrality gap is 4/5.

Half-integral LP solutions. It turns out that in the special case of a *half-integral* solution to the LP, the approximation factor of 4/5 can be achieved. Suppose that the fractional solution is

$$x_{1,S_1} = x_{1,S_2} = x_{2,T_1} = x_{2,T_2} = \frac{1}{2}$$

as shown in the figure.

We could choose a random set S_i, T_j for each bin and then allocate the requested items, resolving the conflict on $S_i \cap T_j$ randomly. This would give an approximation factor of 3/4. A useful observation is, however, that in

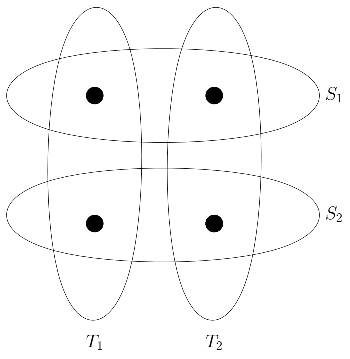

Figure 6.1: A half-integral solution.

some cases we can in fact allocate the entire complement of one set to the other bin. The fractional solution implies that the total size of all items is at most 2, for each of the two bins. This implies that one of the two sets S_1, S_2 must be feasible also for bin 2. Similarly, one of the two sets T_1, T_2 must be feasible for bin 1. Let's assume without loss of generality that S_2 is feasible for bin 2 and T_1 is feasible for bin 1. We use the following randomized allocation scheme.

Probability	1/5	1/5	1/10	1/5	1/5	1/10
Bin 1	S_1	S_2	S_2	T_1	$S_1 \cap T_2$	$S_2 \cap T_2$
Bin 2	S_2	$S_1 \cap T_2$	$S_1 \cap T_1$	T_2	T_1	T_1

By analysis per item, it can be seen that each item is allocated to each bin with total probability at least 2/5. Therefore, we recover at least 4/5 of the LP value on the average.

This is analogous to our improvement to a 5/6-approximation for Submodular Welfare with a half-integral LP solution (Section 5.2.3). The fact that 3/4 is not optimal for $n = 2$ seems to indicate that most probably the $1 - 1/e$ factor is also suboptimal in general.

6.2 The improved GAP algorithm

We proceed to the general case with n bins. We use the aforementioned LP, which can be solved to an arbitrary precision in polynomial time [22]. The remaining issue is how to convert a fractional LP solution to an actual assignment of items to bins. First, let's review the $(1 - 1/e)$-approximation rounding technique given in [22]. We call it *greedy* since it assigns an item preferably to the bin which gives it the maximum value.

Definition 6.1. *Given a fractional solution $x_{i,S}$, we say that bin i samples a random set from its distribution if set S is selected with probability $x_{i,S}$.*

The Greedy Allocation Procedure.

- Let each bin i sample a tentative set S_i from its distribution, independently for different bins.
- If an item appears in multiple tentative sets, we place it in the bin where it brings maximum value.

As shown in [22], this technique yields a solution of expected value at least $(1 - 1/e)$ times the value of the fractional solution. It is our goal here to improve upon this approximation factor. We prove the following.

Theorem 6.2. *There is some universal constant $\epsilon > 0$ and a randomized rounding procedure for the Generalized Assignment Problem such that given any feasible fractional solution, the rounding procedure produces a feasible allocation whose expected total welfare is at least a $(1 - 1/e + \epsilon)$-fraction of the value of the fractional solution.*

By combining this with the LP solver from [22], we obtain:

Corollary 6.3. *There is an absolute constant $\epsilon > 0$ such that the General Assignment Problem admits a $(1 - 1/e + \epsilon)$-approximation.*

Theorem 6.2 is analogous to Theorem 5.1 for the Submodular Welfare Problem. Nevertheless, the issues arising in this problem are quite different than those we had to deal with in the Submodular Welfare Problem. Since there are no capacity constraints in Submodular Welfare, we were able to sample two sets for each player and then combine them in some way to achieve an improvement. Here, this is impossible due to capacities. On the other hand, allocating additional items under a submodular utility function does not necessarily bring any additional profit. Here, the utility functions are linear and additional items always bring profit. Thus the main issue here is, how can we ensure that we can pack some valuable items in addition to the $(1 - 1/e)$-approximation algorithm?

6.2.1 A tight example for the Greedy Allocation Procedure

First, let us consider the Greedy Allocation Procedure and verify that the approximation factor achieved by it is indeed only $1 - 1/e$. The intuitive reason why we lose a $1/e$-fraction of the value is that a $1/e$-fraction of items can remain unallocated to any bin. It is natural to ask whether the procedure can be extended by adding another step where unused items are allocated to some bins if possible. We show that this is not the case.

Example 6.4. *We construct a very special instance where the values and sizes don't depend on bins (i.e. this is a "multiple knapsack problem"). Consider n bins and $n+1$ items, indexed by $\{0, 1, \ldots, n\}$. All item sizes are equal to 1 (the bin capacity), for all bins. Item 0 is the most valuable item, with $v_{i0} = 1$ for all $i \in [n]$. Items $1 \leq j \leq n$ have value $v_{ij} = 1/n^2$ for all $i \in [n]$.*

In this example, a possible LP solution is: $x_{i,\{0\}} = 1/n$ and $x_{i,\{i\}} = 1-1/n$ for all i. The value of this solution is $\sum x_{i,S} v_i(S) = n(1/n \cdot 1 + (1 - 1/n) \cdot 1/n^2) = 1 + 1/n - 1/n^2$. There is also an integral solution of the same value, e.g. the first bin gets item 0 and every other bin i gets item i.

Consider the Greedy Allocation Procedure (or in fact any procedure where tentative sets are sampled with probabilities $x_{i,S}$ in the first step, and contention is resolved arbitrarily). We sample a random set S_i for each bin i. With probability $(1 - 1/n)^n$, we choose $S_i = \{i\}$ for each bin. Then we allocate item i to bin i which yields value $1/n$. With probability $1 - (1 - 1/n)^n$, at least one bin chooses item 0. The obtained value depends on how many bins choose item 0, but in any case we cannot obtain more than the total value of all items, which is $1 + 1/n$. Hence, the expected value obtained by the algorithm is at most $1 - (1 - 1/n)^n + 1/n$, which converges to $1 - 1/e$.

Even worse, suppose that we want to extend this algorithm by allocating the remaining items to some bins. With probability $(1 - 1/n)^n$, bin i takes item i and no bin has space to accommodate the remaining item 0. With probability $1-(1-1/n)^n$, we get at most the total value of all items, $1+1/n$. Again, the expected value obtained is at most $1 - (1 - 1/n)^n + 1/n$.

Thus, we cannot hope to improve the Greedy Allocation Procedure (or any similar procedure with the same first step) by allocating additional items in a second stage. We have to redesign the algorithm, possibly by giving up some items in the first stage. We proceed in a number of stages, where in each stage we either achieve a definite improvement, or we prove that the fractional solution must be of a special form which allows us to proceed to the next stage.

6.2.2 Stage 1: Unbalanced fractional solutions

First, we apply the Fair Contention Resolution technique that we developed in Section 5.3.1 for the purpose of the Submodular Welfare Problem. We summarize this technique here without proof.

The Fair Allocation Procedure.

- Let each bin i sample independently a random tentative set S_i from its distribution. Each bin "competes" for the items in its tentative set.
- Consider an item j. Denote by A_j the random set of bins competing for item j and let y_{ij} be the probability that bin i competes for item j:

$$y_{ij} = \Pr[i \in A_j] = \sum_{S \in \mathcal{F}_i : j \in S} x_{i,S}.$$

  - If $A_j = \emptyset$, do not allocate the item.
  - If $A_j = \{k\}$, allocate the item to bin k.
  - If $|A_j| > 1$, allocate the item to each $k \in A_j$ with probability

$$r_{A_j,k} = \frac{1}{\sum_{i=1}^{n} y_{ij}} \left(\sum_{i \in A_j \setminus \{k\}} \frac{y_{ij}}{|A_j| - 1} + \sum_{i \notin A_j} \frac{y_{ij}}{|A_j|} \right).$$

  It can be seen that $r_{A_j,k} \geq 0$ and $\sum_{k \in A_j} r_{A_j,k} = 1$, so this is a probability distribution.

As we prove in Lemma 5.12, this technique ensures that conditioned on tentatively allocating item j to bin k, the bin actually receives the item with probability

$$\rho_j = \frac{1 - \prod_{i=1}^{n}(1 - y_{ij})}{\sum_{i=1}^{n} y_{ij}} \geq 1 - \frac{1}{e}.$$

This technique actually achieves a factor strictly better than $1 - 1/e$, unless the fractional solution is very well "balanced". (We also take advantage of this fact in the analysis of the Submodular Welfare Problem.)

Definition 6.5. *Fix $\epsilon_1 > 0$. Call an item j ϵ_1-*balanced *if $\forall i; y_{ij} \leq \epsilon_1$ and $\sum_{i=1}^{n} y_{ij} \geq 1 - \epsilon_1$.*

Let U denote the set of ϵ_1-unbalanced items.

1. If a non-negligible value comes from U, e.g. $\sum_i \sum_{j\in U} y_{ij}v_{ij} \geq \epsilon_1^3 \cdot LP$, then we gain by applying the Fair Allocation Procedure to resolve conflicts. For each unbalanced item $j \in U$, if $y_{ij} > \epsilon_1$ for some i then we allocate the item to each bin with conditional probability
$$\rho_j = \frac{1-\prod_i(1-y_{ij})}{\sum_i y_{ij}} \geq \frac{1-\frac{1-\epsilon_1}{e^{-\epsilon_1}}e^{-\sum_i y_{ij}}}{\sum_i y_{ij}} \geq 1-\frac{1}{e}\left(1-\frac{1}{2}\epsilon_1^2\right).$$
In case $\sum_i y_{ij} < 1-\epsilon_1$, we gain at least the same amount. Therefore, the expected value of our solution is at least
$$\begin{aligned}\sum_i\sum_j \rho_j y_{ij}v_{ij} &\geq \sum_i\sum_j\left(1-\frac{1}{e}\right)y_{ij}v_{ij} + \sum_i\sum_{j\in U}\frac{1}{2e}\epsilon_1^2 y_{ij}v_{ij} \\ &\geq \left(1-\frac{1}{e}\right)LP + \frac{\epsilon_1^5}{2e}\cdot LP.\end{aligned}$$

2. If the contribution of U is small, $\sum_i\sum_{j\in U} y_{ij}v_{ij} < \epsilon_1^3 \cdot LP$, then let's remove the unbalanced items. This incurs a factor of $(1-\epsilon_1^3)$ on the value of the fractional solution and in the following, we can assume that the fractional solution is ϵ_1-balanced.

The value of ϵ_1 that we use is determined by the subsequent stages (in fact, exceedingly small). We will choose $\epsilon_1 = 10^{-24}$, hence the improvement that we achieve in Case 1 is on the order of 10^{-120}. The improvement in Case 2 depends on what we can achieve in the following. We will prove that for any ϵ_1-balanced fractional solution, we can get a $(1-1/e+\epsilon_1^3)$-approximation. Since we lost a factor of $1-\epsilon_1^3$ by making the solution ϵ_1-balanced, the resulting approximation factor will be $(1-\epsilon_1^3)(1-1/e+\epsilon_1^3) = 1-1/e+\epsilon_1^3/e-\epsilon_1^6$, i.e. on the order of 10^{-72}.

6.2.3 Stage 2: Non-uniform item values

Here, we treat the case where item values vary significantly among different bins. We know that in any case, the Fair Allocation Procedure yields a factor of $1-1/e$. We show that if the item values are significantly non-uniform, the Greedy Allocation Procedure (assigning each item to the best bin) gains significantly compared to the Fair Allocation Procedure. We choose $\epsilon_2 = 2\epsilon_1$.

Lemma 6.6. *For each item j, let $W_j = \sum_i y_{ij}v_{ij}$ and $B_j = \{i : v_{ij} < (1-\epsilon_2)W_j\}$. Call the value of item j non-uniform if $\sum_{i\in B_j} y_{ij} > \epsilon_2$. For any non-uniform-value item, the Greedy Allocation Procedure retrieves value at least $(1-1/e+(\epsilon_2/e)^2)W_j$.*

We interpret the set B_j as "bad bins" for item j. The meaning of this lemma is that if many bins are bad for item j, then we gain by placing it in a good bin rather than a bad one.

Proof. Let j be an item of non-uniform value. Let's focus on the event that exactly one good bin and one bad bin compete for item j. Let $i \in B_j$ and $k \notin B_j$; the probability that the competing bins are exactly i and k is

$$\Pr[A_j = \{i,k\}] = y_{ij} y_{kj} \prod_{\ell \neq i,k} (1 - y_{\ell j}) \geq (1 - \epsilon_1) e^{-1} y_{ij} y_{kj},$$

using $y_{\ell j} < \epsilon_1$ and $\sum y_{\ell j} \leq 1$. In this case, the Fair Allocation Procedure assigns the item to bin i or k with probabilities $1/2 \pm O(\epsilon_1)$. On the other hand, the Greedy Allocation Procedure places the item always in bin k. Conditioned on i, k being the bins competing for item j, Greedy gains at least $(1/2 - O(\epsilon_1))(v_{kj} - v_{ij}) \geq (1/2 - O(\epsilon_1))(v_{kj} - (1-\epsilon_2)W_j)$ compared to the Fair Allocation Procedure. The total expected gain is

$$\begin{aligned}
\mathbf{E}[\text{gain}] &\geq \sum_{i \in B_j} \sum_{k \notin B_j} \Pr[A_j = \{i,k\}] \left(\frac{1}{2} - O(\epsilon_1)\right) (v_{kj} - (1-\epsilon_2)W_j) \\
&\geq \sum_{i \in B_j} \sum_{k \notin B_j} (1-\epsilon_1) e^{-1} y_{ij} y_{kj} \left(\frac{1}{2} - O(\epsilon_1)\right) (v_{kj} - (1-\epsilon_2)W_j) \\
&\geq e^{-2} \sum_{i \in B_j} y_{ij} \sum_{k \notin B_j} y_{kj} (v_{kj} - (1-\epsilon_2)W_j) \\
&\geq e^{-2} \epsilon_2 \sum_{k \notin B_j} y_{kj} (v_{kj} - (1-\epsilon_2)W_j).
\end{aligned}$$

using $\sum_{i \in B_j} y_{ij} > \epsilon_2$, since item j has non-uniform value. Also, this means

$$\begin{aligned}
W_j = \sum_k y_{kj} v_{kj} &= \sum_i y_{ij}(1-\epsilon_2)W_j + \sum_k y_{kj}(v_{kj} - (1-\epsilon_2)W_j) \\
&\leq (1-\epsilon_2)W_j + \sum_{k \notin B_j} y_{kj}(v_{kj} - (1-\epsilon_2)W_j)
\end{aligned}$$

since $\sum_i y_{ij} \leq 1$ and $(v_{kj} - (1-\epsilon_2)W_j) < 0$ for $k \in B_j$. We get

$$\sum_{k \notin B_j} y_{kj}(v_{kj} - (1-\epsilon_2)W_j) \geq \epsilon_2 W_j.$$

We conclude that $\mathbf{E}[\text{gain}] \geq e^{-2} \epsilon_2^2 W_j$. $\square$

If at least an ϵ_2-fraction of the LP value comes from non-uniform-value items, we gain a factor of $e^{-2}\epsilon_2^3 \geq \frac{1}{8}\epsilon_2^3 = \epsilon_1^3$ which was our goal. If the contribution of non-uniform-value items is smaller than ϵ_2, we define W_j and B_j as before, and

- remove all non-uniform-value items from all sets.
- redefine the value of item j to $v_j = (1 - \epsilon_2)W_j$.
- remove item j from all sets for the bins in B_j.

The first step removes at most $\epsilon_2 \cdot LP$ by assumption. The second step decreases the contribution of item j by $\epsilon_2 W_j$, losing at most $\epsilon_2 \cdot LP$ over all items. The third step decreases the contribution of item j further by $\sum_{i \in B_j} y_{ij} v_j \leq \epsilon_2 W_j$. Thus we lose at most $3\epsilon_2 \cdot LP$ in total. After this procedure, each item has the same value for each bin where it is used (and the new value is not higher than the original value, so any approximation result that we prove with the new values also holds with the old values). In the following, our goal is to achieve an approximation factor at least $1-1/e+3\epsilon_2$. If we achieve this, we obtain total value at least $(1-3\epsilon_2)(1-1/e+3\epsilon_2)LP \geq (1 - 1/e + \epsilon_2)LP$ for small $\epsilon_2 > 0$.

6.2.4 Stage 3: Precious sets

Now we can assume that the value of each item j is independent of the bin where it's placed. We denote that value of item j by v_j and the value of a set S by $v(S) = \sum_{j \in S} v_j$. Next, we consider the distribution of values among different sets for a given bin. The average value assigned to bin i is $V_i = \sum_S x_{i,S} v(S) = \sum_j y_{ij} v_j$. We prove that if many sets are much more valuable than this, we can gain by taking these sets with higher probability. We set $\epsilon_3 = 500\epsilon_2$.

Definition 6.7. *We call a set S "precious" for bin i if $v(S) > 10V_i$.*

Also, from now on we fix a preference order of all the pairs (i, S) where S is a set potentially allocated to bin i. We choose this order in such a way that all non-precious sets come before precious sets. Note that the rule to resolve contention does not change the expected profit here, since each item has the same value for all bins. However, the analysis is cleaner if we adopt the following convention.

The Preferential Allocation Procedure.

- Each bin i chooses a random tentative set S_i independently from its distribution.
- The selected pairs (i, S_i) are processed in their preferential order. Bin i gets all the items in S_i that have not been assigned to any preceding bin.

Before presenting our improvement, let us define another useful notion.

Definition 6.8. *We say that the priority of item j for a set S potentially allocated to bin i is*

$$p_{i,j,S} = \sum_{(i',S')<(i,S), j \in S'} x_{i',S'}$$

where $(i', S') < (i, S)$ means precedence in our preferential order.

Lemma 6.9. *Let $j \in S$ where S is a set potentially allocated to bin i. Assuming that the fractional solution is ϵ_1-balanced, the probability that item j is allocated to some pair (i', S') preceding (i, S) is $1 - e^{-p_{i,j,S}} + O(\epsilon_1)$.*

Proof. The probability that item j is *not* allocated to any pair (i', S') preceding (i, S) is

$$\prod_{i'} \left(1 - \sum_{S':(i',S')<(i,S), j \in S'} x_{i',S'} \right) \geq \exp\left(- \sum_{(i',S')<(i,S), j \in S'} \frac{x_{i',S'}}{1-\epsilon_1} \right) = e^{-\frac{p_{i,j,S}}{1-\epsilon_1}}$$

since $y_{i'j} = \sum_{S':j\in S'} x_{i',S'} \leq \epsilon_1$ for any bin i', and so $1 - y_{i'j} \geq e^{-y_{i'j}/(1-\epsilon_1)}$, due to the condition of ϵ_1-balancedness. Hence, the probability that item j is allocated to some preceding pair is at most $1 - e^{-p_{i,j,S}/(1-\epsilon_1)} \leq 1 - e^{-p_{i,j,S}} + O(\epsilon_1)$ for small $\epsilon_1 > 0$ and $p_{i,j,S} \in [0, 1]$. Similarly, it is easy to see that this probability is at least $1 - e^{-p_{i,j,S}}$. □

Lemma 6.10. *Assume that a β-fraction of the LP value comes from precious sets, $\beta \geq \epsilon_3$. Then there is an algorithm that achieves expected value at least $(1 - 1/e + \epsilon_3/100) \cdot LP$.*

Proof. We assume that the contribution of precious sets is $\beta \cdot LP$. We want to prove that in this situation, we gain by doubling the probabilities of precious sets. Denote the precious sets for bin i by $\mathcal{P}_i$. Denote the probability of sampling a precious set for this bin by $\gamma_i = \sum_{S \in \mathcal{P}_i} x_{i,S}$. Note that $\gamma_i \leq \frac{1}{10}$, because $V_i \geq \sum_{S \in \mathcal{P}_i} x_{i,S} v(S) \geq \gamma_i \cdot 10 V_i$. We boost the probability of each precious set to $x'_{i,S} = 2x_{i,S}$, while the probability of taking each non-precious set is reduced to $x'_{i,S} = \frac{1-2\gamma_i}{1-\gamma_i} x_{i,S}$. The total probability of choosing some non-precious set is modified from $1-\gamma_i$ to $1-2\gamma_i$. How does this affect the expected profit of the algorithm?

Observe that $x'_{i,S}$ is not necessarily a feasible solution but we can still consider its LP value and use it to generate a random allocation, which we compare to that generated by $x_{i,S}$. First, consider what we lose on non-precious sets. In terms of LP value, bin i loses at most $\gamma_i V_i$, since the average value of a non-precious set is at most the average value over all sets for bin i, which is V_i. The contribution of precious sets to bin i is at least $10\gamma_i V_i$, therefore the loss on non-precious sets is at most $\frac{1}{10}$ of the contribution of precious sets, i.e. $\frac{1}{10}\beta \cdot LP$ in total. In both allocation schemes, the order of preference when resolving conflicts is the same and non-precious sets come first in this order; hence, the priority $p_{i,j,S}$ of any item in a non-precious set can only decrease. Therefore, the total loss on non-precious sets cannot be greater than $\frac{1}{10}\beta \cdot LP$ even after resolving conflicts.

Secondly, consider the profit from precious sets. Rather than analyzing the profit per bin, we consider the overall probability that an item is allocated to a precious set. Consider an item j which appears in non-precious sets with total probability $\alpha_j = \sum_i \sum_{S \in \mathcal{F}_i \setminus \mathcal{P}_i} x_{i,S}$ and in precious sets with total probability $\beta_j = \sum_i \sum_{S \in \mathcal{P}_i} x_{i,S}$. This is the original (unmodified) LP solution. Due to Lemma 6.9, item j is still available for after the non-precious sets have been processed with probability $1 - e^{-\alpha_j} + O(\epsilon_1)$. On the other hand, it is available even after all precious sets have been processed with probability $1 - e^{-\alpha_j - \beta_j} + O(\epsilon_1)$. This means that it was actually allocated to a precious set with probability $e^{-\alpha_j} - e^{-\alpha_j - \beta_j} - O(\epsilon_1)$.

Now β_j has been boosted to $2\beta_j$, while α_j decreased to α'_j (which only helps). So in the new setting, the item is allocated to a precious set with probability at least $e^{-\alpha'_j} - e^{-\alpha'_j - 2\beta_j} - O(\epsilon_1)$. Ignoring $O(\epsilon_1)$ terms, the relative gain in probability is

$$\frac{e^{-\alpha'_j} - e^{-\alpha'_j - 2\beta_j}}{e^{-\alpha_j} - e^{-\alpha_j - \beta_j}} \geq \frac{1 - e^{-2\beta_j}}{1 - e^{-\beta_j}} = 1 + e^{-\beta_j} \geq 1 + e^{-1}.$$

Previously, precious sets contributed $\epsilon_3 \cdot LP$ and at least $\epsilon_3 e^{-1} LP$ was actually retrieved by the rounding technique (since any item for any set has priority at most 1 and hence is available for any set with probability at least e^{-1}). Now, we get at least $(1 + e^{-1})\epsilon_3 e^{-1} LP$ for precious sets; i.e. the gain for precious sets is at least $\epsilon_3 e^{-2} LP$. The net gain is at least

$$\left(\frac{\epsilon_3}{e^2} - \frac{\epsilon_3}{10}\right) LP \geq \frac{\epsilon_3}{100} LP.$$

□

Therefore, if Lemma 6.10 applies, we achieve a $(1-1/e+\frac{\epsilon_3}{100})$-approximation which is better than what we need (since $\epsilon_3 = 500\epsilon_2$). Otherwise, let us assume that the contribution of precious sets is less than $\epsilon_3 \cdot LP$. If that's the case, we actually remove precious sets from the fractional solution. If the larger part of a bin's value comes from precious sets, we remove the bin from the solution completely. For other bins, the expected value V_i might go down by a factor of 2; overall we lose at most $2\epsilon_3 \cdot LP$. In the following, any set possibly allocated to bin i has value at most $20V_i$ and we seek to achieve approximation factor at least $1-1/e+2\epsilon_3$. Again, this will produce a gain on the order of $\epsilon_3 \cdot LP$, even larger than we need.

6.2.5 Stage 4: Migrant items

We can now assume that the fractional solution is ϵ_1-balanced, it has no precious sets (of value more than $20V_i$ for bin i) and item values are uniform. Since any bin gives the same value v_j to item j, resolving conflicts does not make any difference in terms of value. However, item sizes still depend on the bin and this complication appears to be fundamental. Our final goal is to pack some additional items into the gaps created by conflicts between bins. It seems beneficial to try to merge these gaps and make them as large as possible. To this purpose, we use the following algorithm which treats the bins in a preferential order and creates most gaps in the bins towards the end of this order.

The Sequential Allocation Procedure.

- Recall the LP share of bin i, $V_i = \sum_j y_{ij} v_j$. Order bins according to these values: $V_1 \le V_2 \le V_3 \le \ldots$
- Bin i chooses a tentative set S_i with probability x_{i,S_i}.
- Each item goes to the first bin that requests it in its tentative set.

Denote by $U_k = \cup_{i<k} S_i$ the items occupied by bins preceding k. Every item outside of U_k is allocated to bin k with probability y_{kj}. This algorithm still achieves a factor of only $1-1/e$, since every item is left unclaimed at the end with probability $\prod_i (1-y_{ij}) \simeq 1/e$. However, we will modify this algorithm. We will prove that in many bins, there will be some space produced by items taken by preceding bins, and there will be also items still available to fill this space.

Definition 6.11. *Define the priority of item j for bin k as*

$$p_{kj} = \sum_{i=1}^{k-1} y_{ij}.$$

An item j is migrant *for bin k, if $p_{kj} \geq 1 - \epsilon_4$. We denote the set of items migrant for bin k by M_k.*

Choice of ϵ_4. We choose $\epsilon_4 = 0.01$; recall that $\epsilon_1 = 10^{-24}$, $\epsilon_2 = 2\epsilon_1$ and $\epsilon_3 = 500\epsilon_2 = 10^{-21}$.

The definition of priority here is analogous to Definition 6.8, now reflecting the fact that the procedure follows a fixed ordering of bins. The intuition behind "migrant items" is that they arrive at the end of the ordering, when it is likely that some other bin already claimed the same item. It is important to keep in mind that an item is migrant only with respect to certain bins; in fact, every item is migrant for some (typically small) fraction of bins. Due to an argument similar to Lemma 6.9, at the moment a migrant item is considered for bin k, it is still available with probability roughly $1/e$.

A non-negligible fraction of the LP value corresponds to migrant items, since $\sum_{i=1}^{n} y_{ij} > 1 - \epsilon_1$ (the condition of balancedness). Therefore, we can assume that the LP value in migrant items is at least $(\epsilon_4 - \epsilon_1)LP \geq \frac{3}{4}\epsilon_4 LP$. items. Consequently, at least a $\frac{1}{4}\epsilon_4$-fraction of bins (by LP contributions) have a $\frac{1}{2}\epsilon_4$-fraction of their value in migrant items. Let's call these bins "flexible bins". We will prove that each flexible bin k can gain a constant fraction of V_k in addition to the value that the Sequential Allocation Procedure allocates to it.

For each set S that could be assigned to bin k, let's define $M_k(S) = M_k \cap S$ to be the items in S which are migrant for bin k. We can assume that the value of these items is either $v(M_k(S)) \geq \frac{1}{4}\epsilon_4 V_k$ or $M_k(S) = \emptyset$. For sets with migrant items of value less than $\frac{1}{4}\epsilon_4 V_k$, we can set $M_k(S) = \emptyset$; this decreases the contribution of migrant items at most by a factor of 2. Let's call the sets with $v(M_k(S)) \geq \frac{1}{4}\epsilon_4 V_k$ "flexible sets", and the collection of all flexible sets for bin k by $\mathcal{M}_k$. The contribution of migrant items in flexible sets is now

$$\sum_{S \in \mathcal{M}_k} x_{k,S}\ v(M_k(S)) \geq \frac{1}{4}\epsilon_4 V_k.$$

Suppose that the total probability of sampling a flexible set for bin k is

$$\sum_{S \in \mathcal{M}_k} x_{k,S} = r_k.$$

Note that since $v(M_k(S)) \leq 20V_k$ for any flexible set S, we get $\frac{1}{4}\epsilon_4 V_k \leq 20V_k r_k$, i.e. $r_k \geq \frac{1}{80}\epsilon_4$. The probability of sampling each individual set is at most $\epsilon_1 << \epsilon_4$ which allows us to assume that we can split the collection of flexible sets into three (roughly) equal parts in terms of probability. Let's select a portion of flexible sets with the largest sizes of $M_k(S)$ and denote them by $\mathcal{A}_k$. For each set $A \in \mathcal{A}_k$, order the remaining flexible sets $S \in \mathcal{M}_k \backslash \mathcal{A}_k$ by the size of $M_k(S) \backslash M_k(A)$. Fix a threshold $\sigma_k(A)$ and partition the sets into two parts $\mathcal{B}_k(A), \mathcal{C}_k(A)$ and so that for any $B \in \mathcal{B}_k(S), C \in \mathcal{C}_k(S)$,

$$size_k(M_k(B) \setminus M_k(A)) \geq \sigma_k(A) \geq size_k(M_k(C) \setminus M_k(A)).$$

We make these groups of sets roughly equal in the sense that for any $A \in \mathcal{A}_k$,

$$\frac{1}{2} r_k \geq \sum_{S \in \mathcal{A}_k} x_{k,S} \geq \sum_{S \in \mathcal{B}_k(A)} x_{k,S} \geq \sum_{S \in \mathcal{C}_k(A)} x_{k,S} \geq \frac{1}{4} r_k.$$

The purpose of this partitioning is that we will be able to switch migrant items between sets: for any $A \in \mathcal{A}_k$ and $B \in \mathcal{B}_k(A)$, $M_k(B)$ fits into the space occupied by $M_k(A)$. Also, for any $B \in \mathcal{B}_k(A)$ and $C \in \mathcal{C}_k(A)$, $M_k(C) \backslash M_k(A)$ fits into the space occupied by $M_k(B) \setminus M_k(A)$.

Our plan of attack is the following: Since migrant items survive until their turn with probability $\simeq 1/e$, many of them will be gone by the time they are considered for allocation. We prove that with a certain probability, we can join the migrant items in $A \in \mathcal{A}_k$ and $B \in \mathcal{B}_k(S)$ and pack them together, where we would normally pack only A. This gives an advantage to some sets in $\mathcal{B}_k(S)$ - but we don't know which ones, and in fact this alone might not bring any profit since the additional set B might be effectively empty (allocated to preceding bins). However, we can use the fact that the migrant items in some sets $B \in \mathcal{B}_k(S)$ have increased their probability of being requested by bin k, and we can decrease this probability back to original, by giving up the migrant items in B voluntarily in the case where B is the primary tentative set selected by bin k. Instead, we can use this space later to pack some migrant items in $C \in \mathcal{C}_k(S)$. Now the probability of packing each set of migrant items in $\mathcal{C}_k(S)$ has increased *uniformly.* The final argument is that a constant fraction of migrant items in $\mathcal{C}_k(S)$ is *almost always* available, therefore we must gain something by this procedure.

First, consider the following variant of our algorithm.

The Modified Sequential Allocation Procedure.

- Process the bins in the order of $V_1 \leq V_2 \leq V_3 \leq \dots$

- For each k, let U_k be the items allocated to bins $1, \dots, k-1$.
- Let bin k sample two random sets S_k, S'_k from its distribution.
- Unless k is a flexible bin, allocate $S_k \setminus U_k$ to it and go to the next bin.
- If $S_k \in \mathcal{A}_k$, $S'_k \in \mathcal{B}_k(S_k)$ and $(S_k \cup M_k(S'_k)) \setminus U_k$ fits in bin k, then pack this set into bin k.
- If $S'_k \in \mathcal{A}_k$, $S_k \in \mathcal{B}_k(S'_k)$ and $(S'_k \cup M_k(S_k)) \setminus U_k$ fits in bin k (as in the previous case but with the roles of S_k and S'_k exchanged), then pack $(S_k \setminus (M_k(S_k) \setminus M_k(S'_k))) \setminus U_k$ into bin k.
- In all other cases, allocate $S_k \setminus U_k$ to bin k and go to the next bin.

This algorithm still achieves only a $(1-1/e)$-approximation. However, it has the advantage that some bins have provably some remaining space at the end.

Lemma 6.12. *In the Modified Sequential Allocation Procedure, every bin k still tentatively requests item j with probability y_{kj}, independently of preceding bins, and hence it achieves a $(1-1/e)$-approximation.*

Conditioned on a selected pair of sets $(S_k = B, S'_k = A)$ for a flexible bin k such that $A \in \mathcal{A}_k, B \in \mathcal{B}_k(A)$, the algorithm leaves available space at least $\sigma_k(A)$ in the bin with probability at least $1/5$.

Proof. Suppose we already processed the first $k-1$ bins and U_k are the items allocated so far. Consider two sets $A \in \mathcal{A}_k$ and $B \in \mathcal{B}_k(A)$ such that $(A \cup M_k(B)) \setminus U_k$ fits in bin k. When sampling the sets S_k, S'_k for bin k, we have two symmetric events of equal probability: $(S_k = A, S'_k = B)$ and $(S_k = B, S'_k = A)$. The difference between the two allocation procedures is that instead of choosing $S_k = A$ in the first event and $S_k = B$ in the second event, we choose $S_k \cup M_k(S'_k) = A \cup M_k(B) = (A \setminus M_k) \cup (M_k(A) \cup M_k(B))$ in the first event, and $S_k \setminus (M_k(S_k) \setminus M_k(S'_k)) = B \setminus (M_k(B) \setminus M_k(A)) = (B \setminus M_k) \cup (M_k(A) \cap M_k(B))$ in the second event. Both of these sets fit in bin k, and together they contain each element in $A \cap B$ twice and each element in $A \Delta B$ once. Therefore, the total probability of bin k requesting any given item j remains the same, y_{kj}, as in the Sequential Allocation Procedure. The probability of allocating item j is $1 - \prod_k (1 - y_{kj}) \geq (1-1/e) \sum_k y_{kj}$, so this is a $(1-1/e)$-approximation.

Now fix a pair $(S_k = B, S'_k = A)$ such that $A \in \mathcal{A}_k$ and $B \in \mathcal{B}_k(S'_k)$ and let us estimate the probability that $(A \cup M_k(B)) \setminus U_k$ fits in bin k. If this happens, we allocate only the set $(B \setminus (M_k(B) \setminus M_k(A))) \setminus U_k$ to bin k, and then the remaining space is at least $size_k(M_k(B) \setminus M_k(A)) \geq \sigma_k(A)$. Since any migrant item survives outside of U_k with probability at most $e^{-(1-\epsilon_4)} < \frac{2}{5}$, and the size of $M_k(B)$ is at most the size of $M_k(A)$, we get

$$\begin{aligned} \mathbf{E}_{U_k}[size_k((M_k(A) \cup M_k(B)) \setminus U_k)] &\leq \frac{2}{5} size_k(M_k(A) \cup M_k(B)) \\ &\leq \frac{4}{5} size_k(M_k(A)). \end{aligned}$$

By Markov's inequality, we get

$$\begin{aligned} &\Pr_{U_k}[(A \cup M_k(B)) \setminus U_k \text{ fits}] \\ \geq\ &\Pr_{U_k}[size_k((M_k(A) \cup M_k(B)) \setminus U_k) \leq size_k(M_k(A))] \geq \frac{1}{5}. \end{aligned}$$

□

Let's call a flexible bin k *available*, if after the procedure above, it still has available space at least $\sigma_k(A)$ for some flexible set $A \in \mathcal{A}_k$. We know that this happens with probability at least $1/5$ for any fixed pair of sets $A \in \mathcal{A}_k$, $B \in \mathcal{B}_k(A)$. Our final goal is to pack some additional items in the available space. For this purpose, we have the flexible sets $C \in \mathcal{C}_k(A)$ for each bin. We know that these sets have the property that $size_k(M_k(C) \setminus M_k(A)) \leq \sigma_k(A)$, and this is exactly the space that's available in bin k. The only remaining question is whether there is a suitable set in $\mathcal{C}_k(S)$ whose migrant items have not been allocated to other bins yet. However, since we have a choice from all the possible sets in $\mathcal{C}_k(S)$, we will prove that some of their items must be still available. This is our final algorithm.

The Extended Sequential Allocation Procedure.

- Run the Modified Sequential Allocation Procedure.
- After it finishes, process all the available bins k with remaining space at least $\sigma_k(S'_k)$, $S'_k \in \mathcal{A}_k$, in an arbitrary order. For bin k, sample a random set S''_k from its distribution.
- If $S''_k \in \mathcal{C}_k(S'_k)$, then allocate the available items in $M_k(S''_k) \setminus M_k(S'_k)$ to bin k.

Theorem 6.13. *For any balanced fractional solution where item values are uniform and there are no precious sets, the Extended Sequential Allocation Procedure with the choice of* $\epsilon_4 = 0.01$ *achieves expected value at least* $(1 - 1/e + 2 \cdot 10^{-21}) \cdot LP$.

Recall that our goal was to achieve a gain of $2\epsilon_3 LP = 2\cdot 10^{-21}LP$. All we need to show in order to prove Theorem 6.13 is that there are still enough migrant items available in $\mathcal{C}_k(S'_k)$ for most available bins, so that we can gain something by packing additional items from these sets. For this purpose, we prove the following concentration result.

Lemma 6.14. *Let X be a finite set with a weight function $w : X \to \mathbb{R}_+$, $w(X) = 1$; $\alpha, \lambda > 0$ and $0 < \epsilon < 1/2$. Let $S_1, S_2, \dots, S_q$ be independent random subsets of X where $\Pr[j \in S_i] = y_{ij}$ (but elements in S_i do not necessarily appear independently). Let*

- $\forall i; \forall j \in X; y_{ij} \le \epsilon$.
- $\forall j \in X; \sum_{i=1}^{q} y_{ij} \le 1$.
- *We always have $w(S_i) \le \alpha$.*

Then

$$\Pr\left[w\left(\bigcup_{i=1}^{q} S_i\right) > 1 - e^{-(1+\epsilon)} + \lambda\right] < \frac{\alpha + 2\epsilon}{\lambda^2}.$$

Proof. Let's define a random variable

$$X = w\left(\bigcup_{i=1}^{q} S_i\right).$$

We apply the second moment method to X. First, the expectation is

$$\begin{aligned} \mathbf{E}[X] &= \sum_{j\in X} w_j \Pr\left[j \in \bigcup_{i=1}^{q} S_i\right] = \sum_{j\in X} w_j \left(1 - \prod_{i=1}^{q}(1 - y_{ij})\right) \\ &\le \sum_{j\in X} w_j \left(1 - \prod_{i=1}^{q} e^{-(1+\epsilon)y_{ij}}\right) \le 1 - e^{-(1+\epsilon)} \end{aligned}$$

using $1 - y_{ij} \ge e^{-(1+\epsilon)y_{ij}}$ for $y_{ij} \le \epsilon < 1/2$, and $\sum_{j\in X} w_j = 1$. The second moment is:

$$\begin{aligned} \mathbf{E}[X^2] &= \sum_{j,k\in X} w_j w_k \Pr\left[j,k \in \bigcup_{i=1}^{q} S_i\right] = \sum_{j,k\in X} w_j w_k \Pr\left[\exists \ell, m; j \in S_\ell, k \in S_m\right] \\ &\le \sum_{\ell=1}^{q} \sum_{j,k\in X} w_j w_k \left(\Pr\left[j,k \in S_\ell\right] + \Pr\left[j \in S_\ell \setminus \bigcup_{i<\ell} S_i, k \in \bigcup_{m\neq\ell} S_m\right]\right). \end{aligned}$$

We estimate the first term using the condition $w(S_\ell) \leq \alpha$:

$$\begin{aligned} &\textstyle\sum_{\ell=1}^{q} \sum_{j,k \in X} w_j w_k \Pr[j,k \in S_\ell] \\ &\leq \textstyle\sum_{\ell=1}^{q} \sum_{j \in X} w_j \Pr[j \in S_\ell] \sum_{k \in X} w_k \Pr[k \in S_\ell \mid j \in S_\ell] \\ &= \textstyle\sum_{\ell=1}^{q} \sum_{j \in X} w_j y_{\ell j} \cdot \mathbf{E}[w(S_\ell) \mid j \in S_\ell] \leq \sum_{\ell=1}^{q} \sum_{j \in X} w_j y_{\ell j} \cdot \alpha \ \leq\ \alpha. \end{aligned}$$

The second term is:

$$\begin{aligned} &\textstyle\sum_{\ell=1}^{q} \sum_{j,k \in X} w_j w_k \Pr\left[j \in S_\ell \setminus \bigcup_{i<\ell} S_i, k \in \bigcup_{m \neq \ell} S_m\right] \\ &= \textstyle\sum_{\ell=1}^{q} \sum_{j \in X} w_j \Pr\left[j \in S_\ell \setminus \bigcup_{i<\ell} S_i\right] \sum_{k \in X} w_k \Pr\left[k \in \bigcup_{m \neq \ell} S_m \mid j \notin \bigcup_{i<\ell} S_i\right] \\ &= \textstyle\sum_{\ell=1}^{q} \sum_{j \in X} w_j \Pr\left[j \in S_\ell \setminus \bigcup_{i<\ell} S_i\right] \sum_{k \in X} w_k (1 - \prod_{m \neq \ell} \Pr\left[k \notin S_m \mid j \notin \bigcup_{i<\ell} S_i\right]). \end{aligned}$$

We use the fact that the event $k \in S_m$ is independent of S_i for any $i \neq m$ and since $\Pr[j \in S_m] = y_{mj}$, conditioning on $j \notin S_m$ can only increase the probability that $k \in S_m$ by a factor of $1/(1-y_{mj}) \leq 1/(1-\epsilon)$. Therefore, we get

$$\begin{aligned} &\textstyle\sum_{k \in X} w_k \left(1 - \prod_{m \neq \ell} \Pr\left[k \notin S_m \mid j \notin \bigcup_{i<\ell} S_i\right]\right) \\ &\leq \textstyle\sum_{k \in X} w_k \left(1 - \prod_{m=1}^{n} \left(1 - \frac{y_{mk}}{1-\epsilon}\right)\right) \\ &\leq \textstyle\sum_{k \in X} w_k \sum_{m=1}^{n} \frac{y_{mk}}{1-\epsilon} \leq (1+2\epsilon)\mathbf{E}[X]. \end{aligned}$$

Then, we can also simplify

$$\textstyle\sum_{\ell=1}^{q} \sum_{j \in X} w_j \Pr\left[j \in S_\ell \setminus \bigcup_{i<\ell} S_i\right] = \sum_{j \in X} w_j \Pr\left[j \in \bigcup_{\ell=1}^{q} S_\ell\right] = \mathbf{E}[X].$$

So the second moment can be bounded by

$$\mathbf{E}[X^2] \leq \alpha + (1+2\epsilon)\mathbf{E}[X]^2 \leq \mathbf{E}[X]^2 + \alpha + 2\epsilon.$$

This means that the variance of X is at most $\alpha + 2\epsilon$ and Chebyshev's inequality yields the lemma. □

Now we use this lemma to argue that there is enough value in $\mathcal{C}_k(S'_k)$ with high probability. Recall that for any available bin k, we sample a new set S''_k and whenever $S''_k \in \mathcal{C}_k(S'_k)$, we add all available items from $M_k(S''_k) \setminus M_k(S'_k)$ to bin k. Items could be unavailable because they were already allocated before this stage, or because they were allocated to preceding bins in this stage. To stay on the safe side, we only count the contribution of $M_k(S''_k) \setminus M_k(S'_k) \setminus U_{\neq k}$ where

$$U_{\neq k} = \bigcup_{i \neq k} (S_i \cup M_k(S'_i) \cup M_k(S''_i)),$$

the union of all items potentially allocated to other bins. The following lemma summarizes our final claim.

Lemma 6.15. *Consider a flexible bin k and fix $S'_k = A \in \mathcal{A}_k$. Call the bin "lucky" if $U_{\neq k}$ is such that the expected value of the set $M_k(S''_k) \setminus M_k(A) \setminus U_{\neq k}$ is*

$$\sum_{S'' \in \mathcal{C}_k(A)} x_{k,S''} v(M_k(S'') \setminus M_k(A) \setminus U_{\neq k}) \geq 10^{-8} V_k.$$

Then, for $\epsilon_4 = 0.01$, each available bin is lucky with probability at least $5/6$.

Proof. For a fixed $A \in \mathcal{A}_k$, define

$$y''_{kj} = \sum_{S'' \in \mathcal{C}_k(A): j \in M_k(S'')} x_{k,S''}.$$

Then we have

$$\sum_{S'' \in \mathcal{C}_k(A)} x_{k,S''} v(M_k(S'') \setminus M_k(A) \setminus U_{\neq k}) = \sum_{j \notin U_{\neq k} \cup M_k(A)} y''_{kj} v_j.$$

Our goal is to prove that this is at least $10^{-8} V_k$ with probability close to 1. First, we define a normalized weight function

$$w_j = \frac{y''_{kj} v_j}{V''_k}$$

where $V''_k = \sum_{S'' \in \mathcal{C}_k(A)} x_{k,S''} v(M_k(S'')) = \sum_j y''_{kj} v_j$, so that $\sum_j w_j = 1$. V''_k is the LP contribution of migrant items in $\mathcal{C}_k(A)$ to bin k. We know that this contribution is $V''_k \geq \frac{1}{4} r_k \cdot \frac{1}{4} \epsilon_4 V_k \geq \frac{1}{1280} \epsilon_4^2 V_k$. In the following, we prove that

$$w(U_{\neq k} \cup M_k(A)) \leq 0.85 \tag{6.1}$$

with probability $\geq 5/6$, which implies the statement of the lemma:

$$\sum_{j \notin U_{\neq k} \cup M_k(A)} y''_{kj} v_j = (1 - w(U_{\neq k} \cup M_k(A))) V''_k \geq 0.15 V''_k \geq 10^{-4} \epsilon_4^2 V_k = 10^{-8} V_k.$$

We proceed in three steps to estimate $w(U_{\neq k} \cup M_k(A))$.

1. For sets $S_1, S_2, \ldots, S_{k-1}$, we apply Lemma 6.14. We always have $y''_{kj} \leq y_{kj} \leq \epsilon_1$. We use $q = k - 1$. The set S_i chosen by any preceding bin $i < k$ has value at most $20 V_i \leq 20 V_k$ (here it's important that we ordered the bins by their values). Therefore, the weight of S_i can be at most $w(S_i) = \sum_{j \in S_i} y''_{kj} v_j / V''_k \leq \epsilon_1 v(S_i) / V''_k \leq 20 \epsilon_1 V_k / V''_k$. We know that the contribution of the flexible sets in $\mathcal{C}_k(A)$ is $V''_k \geq \frac{1}{1280} \epsilon_4^2 V_k$, i.e. $w(S_i) \leq 25600 \epsilon_1 / \epsilon_4^2 \leq 10^{-15}$, considering our values of $\epsilon_1 = 10^{-24}$ and

$\epsilon_4 = 10^{-2}$. Choosing $\epsilon = \epsilon_1$, $\alpha = 10^{-15}$ and $\lambda = 0.01$, Lemma 6.14 implies that

$$\Pr\left[w\left(\bigcup_{i=1}^{k-1} S_i\right) > 1 - e^{-(1+10^{-24})} + 0.01\right] \leq \frac{\alpha + 2\epsilon_1}{\lambda^2} \leq 10^{-10}.$$

2. Concerning the remaining items in $U_{\neq k}$, we only have to consider migrant items for each respective bin: in $M_k(S'_i)$ and $M_k(S''_i)$, all items are migrant by definition, and in $S_i, i > k$, the only relevant items are those migrant for bin k and hence also for $i > k$. The total probability that j is sampled as a migrant item by any bin is at most $3\epsilon_4$ (since in each of the collections of sets $\{S_i\}$, $\{S'_i\}$ or $\{S''_i\}$, an item appears as migrant with probability at most ϵ_4). Therefore

$$\mathbf{E}\left[w\left(\bigcup_{i>k} S_i \cup \bigcup_{i\neq k} M_k(S'_i) \cup \bigcup_{i\neq k} M_k(S''_i)\right)\right] \leq 3\epsilon_4 \sum_j w_j = 3\epsilon_4$$

and

$$\Pr\left[w\left(\bigcup_{i>k} S_i \cup \bigcup_{i\neq k} M_k(S'_i) \cup \bigcup_{i\neq k} M_k(S''_i)\right) > 0.2\right] \leq 15\epsilon_4 = 0.15.$$

3. Finally, the items occupied by $M_k(A)$ need to be considered separately, since A is not random here. However, by the same reasoning that we used in step 1,

$$w(M_k(A)) = \frac{1}{V''_k} \sum_{j\in M_k(A)} y''_{kj} v_j \leq \frac{1280}{\epsilon_4^2 V_k} \epsilon_1 v(M_k(A)) \leq 10^5 \frac{\epsilon_1}{\epsilon_4^2} = 10^{-15}.$$

By adding up the contributions of the three types of sets, we obtain that the probability that $w(U_{\neq k} \cup M_k(A)) > 1 - e^{-(1+10^{-24})} + 0.01 + 0.2 + 10^{-25} = 0.842...$ is at most $0.15 + 10^{-10} \leq 1/6$, which proves (6.1). $\square$

We summarize: For any flexible bin k, the probability that the first two selected sets are $S'_k \in \mathcal{A}_k$ and $S_k \in \mathcal{B}_k(S)$ is $\frac{1}{4} r_k^2 \geq \frac{1}{25600}\epsilon_4^2$. Conditioned on any such pair of sets $S'_k = A$ and $S_k = B$, bin k is has available space at least $\sigma_k(A)$ after the first stage with probability at least 1/5 (Lemma 6.12). Also, the bin is "lucky" with probability at least 5/6 (Lemma 6.15). Hence, with probability at least $1/5 + 5/6 - 1 = 1/30$, both events happen and the bin is available and lucky at the same time. Then, it gains an expected value at least $10^{-8} V_k$ (Lemma 6.15).

Thus the expected gain is at least $\frac{1}{25600}\epsilon_4^2 \cdot \frac{1}{30} \cdot 10^{-8} V_k \geq 10^{-14}\epsilon_4^2 V_k$, for each flexible bin k. These bins constitute a $\frac{1}{4}\epsilon_4$-fraction of the LP value, therefore the total gain is at least $\frac{1}{4} \cdot 10^{-14}\epsilon_4^3 LP = \frac{1}{4} \cdot 10^{-20} LP$.

List of Figures

Bibliography

[1] A. Ageev and M. Sviridenko. An 0.828 approximation algorithm for uncapacitated facility location problem, *Discrete Applied Mathematics* 93:2-3, (1999) 149–156.

[2] A. Ageev and M. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *J. of Combinatorial Optimization*, 8:307–328, 2004.

[3] P. Alimonti. Non-oblivious Local Search for MAX 2-CCSP with application to MAX DICUT, *Proc. of the 23rd International Workshop on Graph-theoretic Concepts in Computer Science* (1997).

[4] G. Calinescu, C. Chekuri, M. Pál and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint, *in IPCO 2007*, 182–196.

[5] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. *Proc. of APPROX*, Springer LNCS, 72–83, 2004.

[6] C. Chekuri and M. Pál. A recursive greedy algorithm for walks in directed graphs. *Proc. of IEEE FOCS*, 2005.

[7] C. Chekuri and S. Khanna. A PTAS for the multiple knapsack problem. *SIAM J. on Computing*, 35(3):713–728, 2004.

[8] V. Cherenin. Solving some combinatorial problems of optimal planning by the method of successive calculations, *Proc. of the Conference of Experiences and Perspectives of the Applications of Mathematical Methods and Electronic Computers in Planning* (in Russian), Mimeograph, Novosibirsk (1962).

[9] G. Cornuejols, M. Fischer and G. Nemhauser. Location of bank accounts to optimize float: an analytic study of exact and approximation algorithms, *Management Science*, 23 (1977), 789–810.

[10] G. Cornuejols, M. Fischer and G. Nemhauser. On the uncapacitated location problem, *Annals of Discrete Math* 1 (1977), 163–178.

[11] G. P. Cornuejols, G. L. Nemhauser and L. A. Wolsey. The uncapacitated facility location problem, *Discrete Location Theory* (1990), 119–171.

[12] I. Dinur, V. Guruswami, S. Khot and O. Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. *Proc. of ACM STOC*, 595–601, 2003.

[13] S. Dobzinski, N. Nisan and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. Proceedings of STOC 2005: 610–618.

[14] S. Dobzinski and M. Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. Proceedings of SODA 2006: 1064–1073.

[15] J. Edmonds. Matroids, submodular functions and certain polyhedra, *Combinatorial Structures and Their Applications* (1970), 69–87.

[16] U. Feige and M. X. Goemans. Approximating the value of two-prover systems, with applications to MAX-2SAT and MAX-DICUT, *Proc. of the 3rd Israel Symposium on Theory and Computing Systems*, Tel Aviv (1995), 182–189.

[17] U. Feige. A threshold of $\ln n$ for approximating Set Cover, *Journal of ACM* 45 (1998), 634–652.

[18] U. Feige. Maximizing social welfare when utility functions are subadditive, *Proc. of 38th STOC* (2006), 41–50.

[19] U. Feige and J. Vondrák. Approximation algorithms for combinatorial allocation problems: Improving the factor of $1-1/e$, *Proc. of 47th FOCS* (2006), 667–676.

[20] S. Fujishige. Canonical decompositions of symmetric submodular systems, *Discrete Applied Mathematics* 5 (1983), 175–190.

[21] L. Fleischer, S. Fujishige and S. Iwata. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions, *Journal of ACM* 48:4 (2001), 761-777.

[22] L. Fleischer, M.X. Goemans, V.S. Mirrokni and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. *Proc. of ACM-SIAM SODA*, 611–620, 2006.

[23] U. Feige, V. Mirrokni and J. Vondrák. Maximizing nonmonotone submodular functions, to appear in *FOCS 2007*.

[24] A. Frank. Matroids and submodular functions, *Annotated Biblographies in Combinatorial Optimization* (1997), 65-80.

[25] R. Gandhi, S. Khuller, S. Parthasarathy and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *JACM*, 53(3):324–360, 2006.

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

[27] B. Goldengorin, G. Sierksma, G. Tijsssen and M. Tso. The data correcting algorithm for the minimization of supermodular functions, *Management Science*, 45:11 (1999), 1539–1551.

[28] B. Goldengorin, G. Tijsssen and M. Tso. The maximization of submodular Functions: Old and new proofs for the correctness of the dichotomy algorithm, *SOM Report*, University of Groningen (1999).

[29] V. Guruswami. Inapproximability results for set splitting and satisfiability problems with no mixed clauses, *Algorithmica* 38 (2004), 451–469.

[30] V. Guruswami and Subhash Khot. Hardness of Max 3-SAT with no mixed clauses, *CCC 2005*.

[31] E. Halperin and U. Zwick. Combinatorial approximation algorithms for the maximum directed cut problem, *Proc. of 12th SODA* (2001), 1–7.

[32] J. Håstad. Some optimal inapproximability results, *Journal of ACM* 48 (2001): 798–869.

[33] R. Karp. Reducibility among combinatorial problems, *Plenum Press*, 1972, 85–103.

[34] V.R. Khachaturov. Mathematical methods of regional programming (in Russian), *Nauka, Moscow* (1989).

[35] S. Khot, G. Kindler. E. Mossel and R. O'Donnell. Optimal inapproximability results for MAX-CUT and other two-variable CSPs? *Proc. of 45th FOCS* (2004), 146–154.

[36] S. Khot, R. Lipton, E. Markakis and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions, in *Proceedings of WINE 2005.*

[37] S. Khot, R. O'Donnell. SDP gaps and UGC-hardness for MaxCutGain, in *FOCS 2006*, 217–226.

[38] B. Lehmann, D. J. Lehmann and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *ACM Conference on El. Commerce 2001*, 18–28.

[39] D. Livnat, M. Lewin, U. Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. *Proc. of 9th IPCO* (2002), 67–82.

[40] H. Lee, G. Nemhauser and Y. Wang. Maximizing a submodular function by integer programming: Polyhedral results for the quadratic case, *European Journal of Operational Research 94*, 154-166.

[41] L. Lovasz. Submodular functions and convexity. A. Bachem et al., editors, *Mathematical Programmming: The State of the Art*, 235–257.

[42] M. Minoux. Accelerated greedy algorithms for maximizing submodular functions, J. Stoer, ed., *Actes Congress IFIP, Springer Verlag, Berlin* (1977), 234–243.

[43] E. Mossel, R. O'Donnell and K. Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality, *Proc. of 46th FOCS* (2005), 21–30.

[44] N. Nisan and I. Segal. The communication requirements of efficient allocations and supporting prices, in *JET 2006.*

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

[46] G. L. Nemhauser, L. A. Wolsey and M. L. Fisher. An analysis of approximations for maximizing submodular set functions II, *Mathematical Programming Study* 8 (1978), 73–87.

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

[48] T. Robertazzi and S. Schwartz. An accelated sequential algorithm for producing D-optimal designs, *SIAM Journal on Scientific and Statistical Computing* 10, 341–359.

[49] M. Queyranne. A combinatorial algorithm for minimizing symmetric submodular functions, *Proc. of 6th SODA* (1995), 98–101.

[50] A. Schäfer, M. Yannakakis. Simple local search problems that are hard to solve, *SIAM J. Comput.* 20:1 (1991), 56–87.

[51] A. Schrijver. Combinatorial optimization - polyhedra and efficiency. Springer, 2003.

[52] D. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem, *Math. Programming* 62(3):461–474.

[53] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time, *Journal of Combinatorial Theory, Series B* 80 (2000), 346–355.

[54] A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. *Proc. of IEEE FOCS*, 588–597, 2001.

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

[56] L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982.

[57] L. Wolsey. Maximizing real-valued submodular functions: Primal and dual heuristics for location Problems. *Math. of Operations Research*, 7:410–425, 1982.