Thesis for the degree
Doctor of Philosophy

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

עבודת גמר (תזה) לתואר
דוקטור לפילוסופיה

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

By
Rani Izsak

מאת
רני איז'ק

התמודדות עם קושי של בעיות אופטימיזציה באמצעות המצאת מדדי סיבוכיות שימושיים

Coping with hardness of optimization problems by introducing useful complexity measures

Advisor: Prof. Uriel Feige

מנחה: פרופ' אוריאל פייגה

1/3/2017

ג' באדר, תשע"ז

Abstract

Suppose that we need to solve an $\mathcal{NP}$-hard optimization problem. Since we cannot generally solve such problems exactly and efficiently, possible approaches are to design an approximation algorithm or to design an algorithm that might be computationally inefficient. In this thesis, we parametrize $\mathcal{NP}$-hard optimization problems and design algorithms with running time or approximation guarantee proportional to the parameter.

In particular, we consider the welfare maximization problem. This $\mathcal{NP}$-hard problem is known to have a constant approximation guarantee for submodular set functions. We define the *supermodular degree*, a complexity measure for set functions, with value of 0 for submodular set functions and greater values for "less submodular" functions. Then we design an approximation algorithm for the welfare maximization problem with approximation guarantee that deteriorates linearly with the supermodular degree of the valuation set functions of the players, and in particular, constant for submodular set functions.

Other applications that we have for the supermodular degree include an approximation algorithm for a generalization of the welfare maximization problem, an algorithm (and a new model) for a secretary like problem that captures an online setting of welfare maximization, and a voting rule for committee selection that is based on the supermodular degree. Another related result is about an adversarial online welfare maximization problem. Our approximation guarantees deteriorate linearly with the supermodular degree in the offline settings and polynomially in the online settings.

Finally, we introduce another complexity measure – $\mathcal{MPH}$ (Maximum over Positive Hypergraphs), and design an approximation algorithm for the welfare maximization problem with an approximation guarantee that deteriorates linearly with the $\mathcal{MPH}$ level of the valuation set functions of the players.

Acknowledgments

I am extremely grateful to my Ph.D. advisor Uri Feige. Uri is great at explaining ideas in a simple and intuitive way. This ability that Uri has, together with his dedication, was priceless to me. Moreover, Uri is a great person, and I felt that he was always available for me, and was willing to help me with anything, any time.

I would like to thank my co-authors for the pleasure of working with them: Uri Feige, Michal Feldman, Moran Feldman, Nicole Immorlica, Brendan Lucier, Ola Svensson and Vasilis Syrgkanis. I am extremely grateful to Ola also for inviting me twice to work with him and with Moran at EPFL.

Additionally, I would like to thank my Ph.D. committee's members, Moni Naor and David Peleg, for their advice during my research.

I would also like to thank Chidambaram Annamalai, Moshe Babaioff, Irit Dinur, Shahar Dobzinski, Michal Feldman, Inbal Talgam-Cohen and Moshe Tennenholtz for interesting and useful discussions, and their useful suggestions on earlier versions of parts of my thesis.

Finally, I am very grateful for the funding that I have received during my Ph.D. period at Weizmann, thanks to the generosity of donors. I am also very grateful to ERC, the Israel Science Foundation and the I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation for supporting my research.

Declaration

Section 3 is a joint work with Uriel Feige and Moran Feldman. Preliminary versions appeared in [36, 41]. Section 4 is a joint work with Moran Feldman that appeared in [42]. Section 5 is a work that was accepted to AAMAS as an extended abstract. Section 6 is a joint work with Uriel Feige [37]. Section 7 is a work with Uriel Feige, Michal Feldman, Nicole Immorlica, Brendan Lucier and Vasilis Syrgkanis that appeared in [33].

Table of Contents

1 Introduction

Consider the following fundamental problem. We have a set of players and a set of indivisible items. Each player has a valuation set function, giving a value to every possible subset of the items. Our aim is to allocate the items to the players, while maximizing the sum of values of the items the players get, by their personal valuation functions. This problem is called THE WELFARE MAXIMIZATION PROBLEM (also known as "combinatorial auctions") and it has been researched extensively(see, *e.g.*, [73, 20, 27, 26, 1, 32]). Unfortunately, it is $\mathcal{NP}$-hard. Moreover, there are lower bounds excluding the possibility of having reasonable approximation guarantees for this problem. One possible approach to cope with this hardness is to restrict the input (*e.g.*, to submodular valuation functions [73]). For some restrictions, the welfare maximization problem is known to admit constant approximation guarantees. However, this approach has an obvious disadvantage; the problem is not promised to be solved with any approximation guarantee (or at least not with an acceptable one) when it does not obey the restriction. It might be most frustrating if an instance seems to be really close to obey the restriction, but however, slightly disobeys it. Another possible approach is to find approximation algorithms, without restricting the problem, but instead, to have approximation guarantees that are proportional to some *complexity measure* of the instances. Roughly speaking, this means to try having some "good" approximation guarantee in some restricted case, approximation guarantee slightly worse for instances that are close to belong to this restricted case, and generally, approximation guarantees with decreasing quality for instances of increased complexity. The latter approach is the one we study. In order to formalize it, the following fundamental questions should be answered: "What does it mean that an instance is "close" to another instance?" "What does it mean that an instance is "more complex" than another?" and more generally: "How can we measure the "complexity" of instances?" The latter (general) question is formalized by the notion of *complexity measures* of instances of optimization problems. Specifically, a complexity measure for an optimization problem P with a set of possible instances $\mathcal{I}(P)$ is a function $\mathcal{C} : \mathcal{I}(P) \to \mathbb{N}$. Indeed, there are typically infinitely many such functions (since there are typically infinitely many instances of an optimization problem), and it seems to not be necessarily true that each of the measures is meaningful for any optimization problem. But, we aim

to find complexity measures that are:

Natural: One can typically intuitively understand what is the meaning of a value given to an instance of P.

Useful: There exists an algorithm with approximation guarantees proportional to the value of the measure for each instance of P, which improves at least some of the currently known guarantees.

The research we have done includes introducing new complexity measures and designing specific algorithms for them for the welfare maximization problem, as well as for other optimization problems. Indeed, our measures apply to any set function, and are not specific for the welfare maximization problem.

1.1 Complexity measures that we introduced

The first complexity measure we introduced is the supermodular degree [36]. The supermodular degree measures the distance of a valuation function from being submodular. We briefly describe it. Recall that submodular functions admit non-increasing marginal values. That is, a valuation function $f : 2^M \to \mathbb{R}^+ \cup \{0\}$ is submodular, if for every item $j \in M$ and subsets of items $S \subseteq T \subseteq M$, we have that $f(j \mid S) \geq f(j \mid T)$, where $f(j \mid X)$ is the marginal value of j given X: $f(j \mid X) \stackrel{\text{def}}{=} f(\{j\} \cup X) - f(X)$. On the other hand, general valuation functions can admit increasing marginals. That is: $f(j \mid S) < f(j \mid S \cup \{j'\})$. Moreover, the latter is realistic in a welfare maximization setting (as an example, one can think of a battery charger of a phone, with respect to the phone). We see the latter phenomenon as synergy between items, and define the supermodular degree as the maximum number of items that a single item may have synergy with. In particular, this means that submodular valuation functions have supermodular degree of 0, and generally, a valuation function over a set of items M can have supermodular degree of up to $|M|-1$. Our applications for the supermodular degree include an approximation algorithm for the welfare maximization problem (Section 3), an approximation algorithm for a generalization of the welfare maximization problem (Section 3.8), an algorithm (and a new model) for a secretary like problem

that captures an online setting of welfare maximization (Section 4) and a voting rule for committee selection that is based on the supermodular degree (Section 5). Another related result is about an adversarial online welfare maximization problem [65]. Our approximation guarantees deteriorate linearly with the supermodular degree in the offline settings and polynomially in the online settings. We also studied non-monotone set functions (Section 6).

Another complexity measure that we introduced is $\mathcal{MPH}$(Maximum over Positive Hypergraphs) (Section 7). The definition of this measure relies on representing a valuation function by a hypergraph; see [1, 18, 20]. Given a hypergraph with a set of vertices V and a set of weighted hyperedges E, we can see it as a valuation function on the set of items (vertices) V, where the value of a subset $S \subseteq V$ is the sum of weights of hyperedges in the subgraph induced by S. A positive hypergraph valuation function is a valuation function with a hypergraph representation with only non-negative hyperedges. A k-positive hypergraph valuation function is a positive hypergraph valuation function with positive edges of rank at most k. We say that a valuation function f is in $\mathcal{MPH}-k$, if there exists a set of k-positive hypergraph valuation functions $\mathcal{F}$, such that for every subset of items S, $f(S) = \max_{f' \in \mathcal{F}} f'(S)$. $\mathcal{MPH}-1$ is actually the well known XOS class, and any set function over a set of items M is in $\mathcal{MPH}-|M|$. Our applications include an approximation algorithm for the welfare maximization problem with approximation guarantee that deteriorates linearly with $\mathcal{MPH}$.

2 Preliminaries

We recall the following definition (see for example [73]):

Definition 2.1. *Let M be a set, let $f : 2^M \to \mathbb{R}^+$ be a monotone set function and let $j \in M$. The* marginal valuation function *$f_j : 2^{M\setminus\{j\}} \to \mathbb{R}^+$ is a function mapping each subset $S \subseteq M \setminus \{j\}$ to the marginal value of j given S:*

$$f_j(S) \stackrel{def}{=} f(S \cup \{j\}) - f(S) \ .$$

We denote the marginal value $f_j(S)$ also by $f(j \mid S)$. For $S' = \{j_1, \ldots, j_{|S'|}\} \subseteq M$ and $S \subseteq M \setminus S'$ we also use either of the notations $f(j_1, \ldots, j_{|S'|} \mid S)$ or $f(S' \mid S)$ to indicate $f(S \cup S') - f(S)$.

2.1 Types of set functions without complementarities

We recall previously studied types of set functions without complementarities (see for example [73, 32]). Let S be a set and let $f : 2^S \to \mathbb{R}^+$ be a set function.

Definition 2.2. *We say that f is* submodular *if for any $S'' \subseteq S' \subseteq S$ and $x \in S \setminus S'$, $f(x \mid S') \leq f(x \mid S'')$.*

Definition 2.3. *We say that f is in $\mathcal{XOS}$, if for some $l \in \mathbb{N}$ there exist additive set functions $f_1, \cdots f_l$, such that for every $S \subseteq M$, we have that $f(S) = \max_{1\leq i\leq l} f_i(S)$.*

Definition 2.4. *We say that f is* fractionally subadditive *if for every subset $S' \subseteq S$, subsets $T_i \subseteq S'$ and every coefficients $0 < \alpha_i \leq 1$ such that for any $x \in S'$, $\sum_{i:x\in T_i} \alpha_i \geq 1$, it holds that $f(S') \leq \sum_i \alpha_i f(T_i)$.*

Definition 2.5. *We say that f is* subadditive *or* complement free *if for every $S_1, S_2 \subseteq S$, $f(S_1 \cup S_2) \leq f(S_1) + f(S_2)$.*

2.2 Representing set functions

In the welfare maximization problem, the domain of the valuation functions is exponential in the number of items. We recall two possible approaches to cope with this. The first is using an explicit representation model (specifically, we recall a hypergraph representation; see [21], [18], [1]) and the

second is using oracles, representing set functions by supporting queries with respect to them (see for example [12]).

2.2.1 A hypergraph representation

Every set function f can be represented in a unique way as a hypergraph in which the vertices are the items, vertices and hyperedges have weights associated with them, and the value $f(S)$ of a subset S of items equals the sum of all weights in the subgraph induced by the corresponding vertices.

A succinct representation Let $f : 2^M \to \mathbb{R}^+$ be a set function and let $m \stackrel{\text{def}}{=} |M|$. A succinct representation of f is any representation that takes space polynomial in m.

2.2.2 Query models

We recall the definitions of value and demand queries for an underlying set function. Let $f : 2^M \to \mathbb{R}^+$ be a set function.

Definition 2.6. *Value query is the following:*

Input: A subset $S' \subseteq M$.

Output: $f(S')$.

Definition 2.7. *Demand query is the following:*

Input: A cost function $c : M \to \mathbb{R}^+$.

Output: A subset $S' \subseteq M$ maximizing $f(S') - \sum_{j \in S'} c(j)$.

Note that demand queries are strictly stronger than value queries, in the sense that one can answer a value query by using a polynomial number of demand queries, but generally, not vice versa (see [13]).

Definition 2.8. *An oracle for a type of queries for a given set function can answer queries of the respective type with respect to the given set function.*

The notion of an oracle serves as an abstraction for a subroutine that computes an answer to

the respective type of queries for a given set function. When we say that an algorithm uses a certain type of oracle, the running time of the algorithm is computed as if each query takes unit time to answer, regardless of the true running time of the underlying subroutine. This abstraction is most justified if for the underlying set function, answers to the respective query can indeed be computed efficiently. We remark that given a succinct hypergraph representation for a set function, one can efficiently answer value queries, whereas answering demand queries might be $\mathcal{NP}$-hard (see for example Theorem 3.4).

2.3 Independence systems

Given a ground set M, a pair $(M, \mathcal{I})$ is called an independence system if $\mathcal{I} \subseteq 2^M$ is hereditary (that is, for every set $S \in \mathcal{I}$, every set $S' \subseteq S$ is also in $\mathcal{I}$). Independence systems are further divided into a few known classes. The probably most highly researched class of independence systems is the class of matroids.

Definition 2.9 (Matroid)**.** *An independence system is a matroid if for every two sets $S, T \in \mathcal{I}$ such that $|S| > |T|$, there exists an element $u \in S \setminus T$, such that $T + u \in \mathcal{I}$. This property is called the* augmentation property *of matroids.*

Two important types of matroids are uniform and partition matroids. In a uniform matroid a subset is independent if and only if its size is at most k, for some fixed k. In a partition matroid, the ground set M is partitioned into multiple subsets $M_1, M_2, \ldots, M_k$, and an independent set is allowed to contain at most a single element from each subset M_i. Note that the welfare maximization problem can be viewed as the problem of maximizing a set function subject to a partition matroid constraint, where the elements are all the possible pairs of an item and a player, and there is a subset M_i for every item j_i. This means that a subset is independent if and only if every item is allocated to at most a single player.

Some classes of independence systems are parametrized by a value $k \in \mathbb{N}$ ($k \geq 1$). The following is a simple example of such a class.

Definition 2.10 (k-intersection)**.** *An independence system $(M, \mathcal{I})$ is a **k-intersection** if there exist k matroids $(M, \mathcal{I}_1) \ldots (M, \mathcal{I}_k)$ such that a set $S \subseteq M$ is in $\mathcal{I}$ if and only if $S \in \bigcap_{i=1}^{k} \mathcal{I}_i$.*

The problem of k-dimensional matching can be represented as maximizing a linear function over a k-intersection independence system. In this problem, one looks for a maximum weight matching in a k-sided hypergraph, *i.e.*, an hypergraph where the nodes can be partitioned into k "sides" and each edge contains exactly one node of each side. The representation of this problem as the intersection of k partition matroids consists of one matroid per "side" of the hypergraph. The ground set of such a matroid is the set of edges, and a subset of edges is independent if and only if no two edges in it share a common vertex of the side in question.

The following definition, introduced by Mestre [76], describes a more general class of independence systems.

Definition 2.11 (k-extendible)**.** *An independence system $(M, \mathcal{I})$ is a* ***k*-extendible** *system if for every two subsets $T \subseteq S \in \mathcal{I}$ and element $u \notin T$ for which $T \cup \{u\} \in \mathcal{I}$, there exists a subset $Y \subseteq S \setminus T$ of cardinality at most k for which $S \setminus Y + u \in \mathcal{I}$.*

The problem of maximizing a linear function over a k-extendible system captures the problem of k-set packing.[1] In this problem, one is given a weighted collection of subsets of M, each of cardinality at most k, and seeks a maximum weight sub-collection of pairwise disjoint sets. The corresponding k-extendible system is as follows. The ground set contains the sets as elements. The independent subsets are all subsets of pairwise disjoint sets. Let us explain why this is a k-extendible system. Adding a set S of size k to an independent set I, while respecting disjointness, requires that every elements of S is not contained in any other set of I. On the other hand, since I is independent, each element is contained in at most one set of I. Therefore, in order to add S, while preserving disjointness, we need to remove up to k sets from I, as required by Definition 2.11.

The following (strict) inclusions can be shown to hold [15]:

$$\text{matroids} \subset k\text{-intersection} \subset k\text{-extendible systems} \ .$$

[1] k-set packing is, in fact, already captured by a smaller class called k-exchange, defined by [45].

2.4 Online algorithms

The performance of an online algorithm is measured by the competitive ratio which is the worst case ratio between the expected performance of the algorithm and the performance of an offline optimal algorithm. More formally, if $\mathcal{P}$ is the set of possible instances, $OPT(P)$ is the value of the optimal solution for an instance $P \in \mathcal{P}$ and $ALG(P)$ is the value of the algorithm's solution given the instance P. Then, the competitive ratio (for maximization problems) of ALG is given by:

$$\sup_{P \in \mathcal{P}} \frac{OPT(P)}{\mathbb{E}[ALG(P)]} \ ,$$

where the expectation is over the randomness of the algorithm and the arrival order of the input. Most works on online algorithms are interested in the competitive ratio that can be obtained given the information constraints of the online setting. Thus, an online algorithm is interesting even when its time complexity is exponential, as such an algorithm proves that sufficient information is available in the model to obtain results.

3 The Supermodular Degree

The welfare maximization problem (also known as "combinatorial auction") is the following. There is a set of players and a set of indivisible items. Each player has its own (monotone non-decreasing) valuation for any subset of items. The goal is to distribute the items to the players while maximizing social welfare - the sum of values of all players, by their personal valuations. The welfare maximization problem is $\mathcal{NP}$-hard to approximate with any reasonable guarantee. For this reason past research considered restrictions on the class of set functions that may serve as valuation functions for the players. Lehmann, Lehmann and Nisan [73] considered *complement free* functions, which essentially means that a value of a set of items cannot exceed the sum of values of its parts. They presented a hierarchy of classes of complement free functions, and established constant factor approximations for the welfare maximization problem in some cases.[2] Subsequent work established constant factor approximation for all classes of complement free functions. This made it clear that the poor approximation guarantees for the general case must come from complementarities (sets whose value is larger than that of the sum of their parts). Abraham, Babaioff, Dughmi and Roughgarden [1] considered a restricted class of set functions strictly based on complementarities (in particular, no set is valued less than the sum of its parts). Among other results, they presented an algorithm with an approximation guarantee that is linear in a certain parameter related to the extent of these complementarities.

In the current section, similarly to [1], we express the approximation guarantees as a function of some parameter associated with the underlying set functions. The smaller this parameter is, the better the approximation guarantee. However, we depart from the practice of considering restricted classes of set functions – our parametrization can be applied to any set function. It is most advantageous (in the sense that our approximation guarantees are not bad whereas previous work does not apply to these set functions) when the set functions are basically submodular (offer decreasing marginal value, which is the discrete analog of convexity), but exhibit a limited amount

[2]There was earlier work with related results that used different terminology. For example, Fisher, Nemhauser and Wolsey [48] studied "the m-box problem" which is essentially the welfare maximization problem with monotone submodular valuation functions. Among their results there was a greedy approximation algorithm for a generalized version of this problem with approximation guarantee 1/2.

of complementarities. As a simple example of how such functions may arise, consider shopping for shoes. Each additional pair of shoes may have decreasing marginal value, but within a pair of shoes, the left and right shoe are together worth more than the sum of values of each shoe on its own.

In this section, we introduce two new complexity measures of set functions. One is the *dependency degree*. Roughly speaking, this is the maximum number of items that may influence the marginal value of any item with respect to any possible subset of other items. The other is the *supermodular degree*. Roughly speaking, this is the dependency degree, taking into account only items that may *increase* the marginal value of an item. That is, "negative dependencies" do not increase the latter complexity measure. In particular, submodular functions might have arbitrary dependency degree, but their supermodular degree is 0. This measure can also be seen, in a sense, as the "degree of complementarity". We design two greedy approximation algorithms for the welfare maximization problem, each with approximation guarantee linear in the maximum of one of these measures over the set functions of the players. We further generalize our results to the problem of function maximization subject to a k-extendible system constraint, which extends in particular intersection of k-matroids (see definitions in Section 2).

3.1 The Welfare Maximization Problem

We define formally the welfare maximization problem.

Definition 3.1. *An instance $\mathcal{I}(P, M, v)$ of the welfare maximization problem is the following:*

- *P is a set of n players $1, \ldots, n$.*
- *M is a set of m items $j_1, \ldots, j_m$.*
- *v is a vector of n valuation functions (set functions) $v_1, \ldots, v_n$, where $v_p : 2^M \to \mathbb{R}^+$ is the valuation function associated with the player $p \in P$.[3] For any $p \in P$, v_p is restricted to be monotone non-decreasing and with value 0 for the empty set (and hence also non-negative).*

A feasible solution to $\mathcal{I}$ is a mapping $SOL : M \to P$, allocating each of the items to exactly one

[3] We use $\mathbb{R}^+$ to indicate the set of all non-negative real numbers (that is, 0 is included).

player. This mapping induces for each player $p \in P$ a set SOL_p of the items mapped to her. The utility/value of a player $p \in P$ is defined as $v_p(SOL_p)$. Our aim is to maximize the social welfare $v(SOL) \stackrel{def}{=} \sum_{p \in P} v_p(SOL_p)$.

3.2 Measuring dependencies

In this section we introduce complexity measures capturing dependencies of items in a ground set of a set function. For convenience, we treat the set functions as valuation functions of players of an instance of the welfare maximization problem (for notations only). Let M be a set, let $v_p : 2^M \to \mathbb{R}^+$ be a valuation function of player $p \in P$ and let $j \in M$.

Definition 3.2. *The dependency set of j by v_p is the set of all items j' in M such that there exists $S \subseteq M \setminus \{j, j'\}$, such that $v_p(j \mid S \cup \{j'\}) \neq v_p(j \mid S)$. For each such j', we say that j depends on j' by p and denote it by $j \to_p j'$, or by $j \stackrel{S}{\to}_p j'$ if we want to explicitly mention the set S. We denote the dependency set of j by v_p by $Dep_{v_p}(j)$ or simply $Dep_p(j)$. p or v_p may be omitted in any of the above, when it is clear from the context.*

The relation '$\to$' is symmetric (see Section 3.9), so we may also use the terminology "are dependent" and the notation '$\leftrightarrow$'.

Definition 3.3. *The supermodular dependency set of j by v_p is the set of all items j' in M such that there exists $S \subseteq M \setminus \{j, j'\}$, such that $v_p(j \mid S \cup \{j'\}) > v_p(j \mid S)$. Terminology and notations are the same as in Definition 3.2, but with the word "supermodular/ly" or with '+' as a superscript. Note that the relation '$\to^+$' is also symmetric (see Section 3.9).*

Definition 3.4. *The dependency degree of v_p is defined as $\mathcal{D}_{v_p} \stackrel{def}{=} \max_{j \in M} |Dep_p(j)|$. The supermodular dependency degree (or simply, the supermodular degree) of v_p is defined as $\mathcal{D}^+_{v_p} \stackrel{def}{=} \max_{j \in M} |Dep^+_p(j)|$. The (supermodular) dependency degree of an instance of the welfare maximization problem is the maximum (supermodular) dependency degree among all valuation functions of the instance.*

Note that any submodular set function has supermodular degree 0. Note also that $\mathcal{D}^+_f \leq \mathcal{D}_f$ for any set function f.

We also use the following definition:

Definition 3.5. *Let f be a set function. The (supermodular) dependency graph of f is the following. There is a vertex for each item and an undirected edge for each pair of (supermodularly) dependent items.*

3.3 Preliminary observations for the welfare maximization problem

3.3.1 $\mathcal{APX}$-hardness

Proposition 3.1. *The welfare maximization problem is $\mathcal{APX}$-hard even if there are only two players who all have the same valuation function f, with $\mathcal{D}_f^+ = 0$ and $\mathcal{D}_f = 3$.*

Proof. It is known that the question of whether a 3-regular graph has an independent set of a given size is $\mathcal{NP}$-hard, and that it is $\mathcal{APX}$-hard to maximize the size of an independent set []. Given a 3-regular graph G, consider the following instance of the welfare maximization problem with two players. The first player's valuation function has G as its hypergraph representation, with all vertices having value of 3 and all edges having value of -1. The second player's valuation function is additive with value of 2 for each of the items.

Clearly, the dependency degree of the valuation function of the second player is 0 (and hence also its supermodular degree). Additionally, the valuation function of player 1 is submodular and has dependency degree of 3, since its hypergraph is actually a 3-regular graph (hence the dependency degree is 3) with only negative edges (hence this function is submodular and has supermodular degree of 0).

Note that the value of any solution is at most three times the size of the maximum independent set among the vertices of player 1 plus at most 2 per each other item (regardless of which player gets it). Moreover, given a solution to this instance of the welfare maximization problem, one can compute in polynomial time a solution with the same value such that player 1 gets an independent set. To observe that, note that the induced hypergraph representation of the vertices representing the items that player 1 would get, must contain only independent vertices and pairs of vertices. One can arbitrarily choose one vertex of each of those pairs of vertices to obtain an independent set. The proof of Proposition 3.1 follows, since the size of the maximum independent set on a

3-regular graph with m vertices must be $\Omega(m)$, which is at least a constant fraction of the value of the solution to the aforementioned instance of the welfare maximization problem. □

3.3.2 Hardness as a function of the dependency degree

Recall that the maximum weighted k-set packing problem is the following:

Definition 3.6. *Let $G = (V, E, w)$ be a weighted k-uniform hypergraph (i.e., every hyperedge contains exactly k vertices) with set of vertices V, set of undirected edges E and edge weights function w. The maximum weighted **k**-set packing problem is to find a set of disjoint edges of maximum weight.*

This problem is known to be $\mathcal{NP}$-hard for any $k > 2$ and is $\mathcal{NP}$-hard for approximation within a ratio of $\frac{c \ln k}{k}$ for some $c > 0$ by a result of Hazan, Safra and Schwartz [60]. The best approximation guarantee known for it currently (as far as we know) is $2/(k+1)$, by an algorithm of Berman [8]. For completeness, we show the following (see [74]):

Proposition 3.2. *There exists an approximation preserving reduction of the maximum weighted k-set packing problem to the welfare maximization problem with dependency degree at most $k-1$.*

Proof. The reduction is Reduction 3.1.

Reduction 3.1 k-set packing to welfare maximization with dependency degree at most $k-1$

Input: An instance $\mathcal{I}_{SP}(V, E, w)$ of the maximum weighted k-set packing problem, with $V = \{v_1, \ldots, v_{|V|})\}$.

Output: An instance $\mathcal{I}(P, M, v)$ of the welfare maximization problem with dependency degree at most $k-1$.

1: For each vertex $v_i \in V$, create an item $i \in M$.
2: For each hyper-edge $e = \{v_{e_1}, \ldots, v_{e_k}\} \in E$, create a player $p_e \in P$ with $v_p(S) = w(e)$ for every S such that $\{v_{e_1}, \ldots, v_{e_k}\} \subseteq S$ and $v_p(S) = 0$ otherwise.

Clearly, the dependency degree of each of the players is exactly $k-1$, since all of them are single minded players that want only a bundle of k items. That is, if a player wants exactly the items $\{j_1, \ldots j_k\}$, then every $j, j' \in \{j_1, \ldots j_k\}$, $j \neq j'$ are dependent by setting in Definition 3.2 $S = \{j_1, \ldots, j_k\} \setminus \{j, j'\}$. Moreover, other items can never increase the marginal value of any other item, by the valuation function of this player.

It is easy to verify that any feasible solution for $\mathcal{I}_{SP}$ (the input of Reduction 3.1) has a corresponding feasible solution for $\mathcal{I}$ (the output of Reduction 3.1), with the same value, and vice versa, as desired. □

3.3.3 An exact algorithm for dependency degree at most 1

Proposition 3.3. *The welfare maximization problem with dependency degree at most 1 admits an exact polynomial time algorithm.*

The main idea is to use symmetry of the dependency relation in order to reduce an instance of the welfare maximization problem to an instance of maximum weighted matching. The full reduction appears in Section 3.11.

3.3.4 Demand queries and the dependency degree

Theorem 3.4. *Given a hypergraph representation of any set function with dependency degree at most 2, demand queries may be answered in polynomial time (in the size of the hypergraph representation). Given a hypergraph representation of a set function with dependency degree at least 3, demand queries are generally $\mathcal{APX}$-hard to answer (with respect to the size of the hypergraph representation).*

Proof. Let f be a set function. If $\mathcal{D}_f \leq 2$, then each item depends on at most two other items. The dependency graph of f is of maximum degree 2 and hence its connected components are either isolated vertices, isolated paths or isolated cycles. On each such component, demand queries may be answered using dynamic programming.

To show hardness for set functions f with $\mathcal{D}_f \geq 3$, we reduce from the problem of maximum independent set in 3-regular graphs. f is represented by the following hypergraph representation. Give each vertex value 3 and each edge value -1. The value of the answer to a demand query in which the cost of each vertex is 2 is equal to the size of the maximum independent set (it is worth taking a vertex only if it does not contribute an edge to the induced subgraph). Due to $\mathcal{APX}$-hardness of maximum independent set in 3-regular graphs, it is $\mathcal{APX}$-hard to answer demand

queries, as well. □

3.4 Our main algorithmic results

Theorem 3.5. *The welfare maximization problem with supermodular degree at most d admits a polynomial time greedy $\frac{1}{d+2}$-approximation algorithm. This algorithm requires for each valuation function a value oracle and a supermodular dependency graph.*

For the dependency degree we have a slightly better approximation guarantee, which is relevant only when the dependency degree and the supermodular degree are equal:

Theorem 3.6. *The welfare maximization problem with dependency degree at most d admits a greedy $\frac{1}{d+1}$ approximation algorithm. Its running time is polynomial in the number of players and items and in 2^d. This algorithm requires for each valuation function a value oracle and a dependency graph.*

Proposition 3.2 implies that an improvement of our results by a multiplicative factor of more than roughly 2 would improve the current approximation guarantee for weighted k-set packing. Additionally, by a hardness of approximation result of Blumrosen and Nisan [13] of $\Omega(\log m/m)$ for algorithms requiring only value oracles[4], Theorem 3.5 is tight up to a multiplicative factor of $O(\log m)$, in the sense that general set functions have dependency degree (and supermodular degree) of at most $m-1$.

3.4.1 More general results: k-extendible systems

Our most general [5] result is an algorithm for maximizing any monotone set function subject to a k-extendible system, with an approximation guarantee that degrades gracefully as the supermodular degree increases. Note that this algorithm matches the best known approximation guarantees also

[4] Our algorithms require also dependency graph / supermodular dependency graph. However, in the instance used by [13], it is trivial to construct both, since all the items supermodularly depend on each other (for infinitely many values of m). Alternatively, since the approximation hardness is $\Omega(\log m/m)$, we may simply modify the valuations functions so that every two items are supermodularly dependent, by adding a positive hyperedge that contains all items. Details omitted.

[5] the welfare maximization problem is a special case of a matroid constraint which corresponds to 1-extendible system.

for the more specific problems of welfare maximization (this thesis) and maximizing a monotone submodular function subject to a k-extendible system [48].

Theorem 3.7. *There exists a* $(1/(k(\mathcal{D}_f^+ + 1) + 1))$*-approximation algorithm of* $\texttt{Poly}(|M|, 2^{\mathcal{D}_f^+})$ *time complexity for the problem of maximizing a non-negative monotone set function* f *subject to a* k*-extendible system. The algorithm uses a value oracle and a supermodular dependency graph.*

Note that an exponential dependence in $\mathcal{D}_f^+$ is unavoidable, since, otherwise, we would get a polynomial time $(m+1)$-approximation algorithm for maximizing any monotone set function subject to a 1-extendible system.[6]

We show a similar result also for the dependency degree, providing a better approximation guarantee when $\mathcal{D}_f = \mathcal{D}_f^+$.

Theorem 3.8. *There exists a* $(1/(k(\mathcal{D}_f + 1)))$*-approximation algorithm of* $\texttt{Poly}(|M|, 2^{\mathcal{D}_f})$ *time complexity for the problem of maximizing a non-negative monotone set function* f *subject to a* k*-extendible system. The algorithm uses a value oracle and a dependency graph.*

On the other hand, we present examples showing that the bounds guaranteed by Theorems 3.7 and 3.8 are tight for the respective algorithms. Furthermore, we show the following hardness result via a reduction from k-dimensional matching (see Section 3.8.2).

Theorem 3.9. *It is* $\mathcal{NP}$*-hard to approximate non-negative monotone set function maximization subject to a* k*-intersection independence system constraint to within a factor* $\frac{c(\log k + \log \mathcal{D}_f)}{k\mathcal{D}_f}$*, for some constant* $c > 0$*.*

3.4.2 Discussion of results

Our results for the supermodular degree use a value oracle and a supermodular dependency graph, but not a demand oracle. Note that even for $\mathcal{XOS}$ valuations, one needs to use a demand oracle to achieve acceptable guarantees (*i.e.*, a value oracle does not suffice, see [26]). In this sense,

[6] To see that this cannot be done, consider the problem of maximizing the following family of set functions subject to a uniform $k = n/2$ matroid constraint. Each function in the family has a value of 1 for sets strictly larger than k and for a single set A of size k. For all other sets the function assigns the value of 0 (observe that $\mathcal{D}^+(u) = M - u$ for every element $u \in M$, hence, the supermodular dependency oracle is useless in this example). Given a random member of the above family, a deterministic algorithm using a polynomial number of value queries can determine A only with an exponentially diminishing probability, and thus, will also output a set of value 1 with such an exponentially diminishing probability. Using Yao's principle, this implies hardness also for randomized algorithms.

choosing submodular set functions as the simplest according to our measure is the best one can expect. Additionally, note that we measure distance from being submodular by the number of supermodular dependecies of items. This is in contrast of using, *e.g.*, rank of hyperedges in a graph, like [1] did. This enables us to use a greedy algorithm, and thus to refrain, again, from using a demand oracle, this time as a separation oracle for an LP (see also discussion about greedy in [1]).

3.5 Related work

The supermodular degree is a complexity measure for set functions, that ranges from 0 (for submodular set functions) to $m-1$ (where m is the number of items). One of our results is a greedy algorithm for the welfare maximization problem whose approximation guarantee increases linearly with the supermodular degree of the underlying set functions. Such a linear increase is to be expected, given the known reduction from set packing to the welfare maximization problem with single minded bidders, combined with the difficulty of approximating set packing ([60]). As far as we know, the notion of supermodular degree has not appeared in previous work. However, other related notions did appear, and in this section we discuss some of them.

Perhaps the simplest complexity measure for a set function is its support size, namely, the number of items that it depends on. This measure ranges from 1 to m. Moreover, simple greedy algorithms approximate the welfare maximization problem with guarantee equal to this measure, and known hardness results for the case of single minded bidders apply here as well. Hence the results that one can prove for the complexity measures of support size and supermodular degree are of a similar nature. However, as the supermodular degree of a function is not greater than its support size, and moreover, it is often much smaller (the gap being most dramatic for submodular functions), we view our results for supermodular degree as significantly more informative than the corresponding results for support size.

Lehmann, Lehmann and Nisan [73] proposed a hierarchy of classes of set functions based on notions of complement-freeness, and initiated a study of the complexity of the welfare maximization problem for these classes. For the lower classes (linear functions, and functions enjoying the *gross substitutes* property) the welfare maximization problem can be solved in polynomial time. For

submodular functions, a constant approximation guarantee is possible and value queries suffice for this [73, 85] (and somewhat better constants are achievable using demand queries [39]). For XOS (later referred to as *fractionally subadditive* in [35]) and *subadditive* set functions there are approximation algorithms with constant approximation guarantees [26, 27, 35], but they require demand queries (value queries do not suffice [26]). The algorithm given in [73] for submodular functions is greedy, and the greedy algorithm in the current paper can be viewed as an extension of the greedy algorithm of [73] to a setting that is not submodular. The classification of [73] does not distinguish between different classes of functions that are not subadditive, and hence unlike our supermodular degree measure, is applicable only to some classes of set functions, but not to set functions in general. Set functions not lying in the classification of [73] may have any supermodular degree between 1 and m and our algorithms for maximizing welfare distinguish among them in the approximation guarantees that they provide. Lehmann, Lehmann and Nisan [73] do suggest a way of extending their classification to additional functions, as follows. A function f can be called c-submodular, if for every (possibly empty) sets S and T and item x, the marginal value of x with respect to $S \cup T$ is at most c times larger that the marginal value of x with respect to S. (For submodular functions $c = 1$.) It is shown in [73] that the welfare maximization problem can be approximated with guarantee of $c + 1$ when the set functions are c-submodular. We note however that even functions on two items need not be c-submodular for any finite c (if one of the items has value 0 by itself but positive marginal value together with the other item). Another relevant measure is the submodularity ratio of Das and Kempe [23], which is defined, roughly speaking, as the ratio between adding items to a subset one by one and adding all of them together. Their applications are in the field of machine learning (specifically, *e.g.*, subset selection).

Submodular functions play a central role in the definition of the supermodular degree. However, other classes within the hierarchy of [73] have no special significance in this respect. The supermodular degree does not distinguish between linear functions and arbitrary submodular functions – they both have supermodular degree 0. Functions in the [73] hierarchy which are not submodular may have arbitrarily large supermodular degree.[7]

[7]For example, partition the set of items into two disjoint sets, A and B. Let f_A be a linear function that gives value 1 to each item in A and 0 to each item in B. Let f_B be a linear function that gives value 1 to each item in B and 0 to each item in A. Let f be defined as $f(S) = \max[f_A(S), f_B(S)]$. This is an XOS function (according to the

Conitzer, Sandholm and Santi [21] introduce the representation of set functions via hypergraphs with positive and negative hyperedges (presented in Section 2.2. See also the work of Chevaleyre, Endriss, Estivie and Maudet [18], who defined independently a similar concept). They showed that even if each hyperedge has at most two vertices, the welfare maximization problem is $\mathcal{NP}$-hard. They did not consider approximation algorithms. Abraham, Babaioff, Dughmi and Roughgarden [1] consider supermodular functions which have no negative hyperedges in their hypergraph representation. Among other results, they give an algorithm approximating the welfare maximization problem within a value equal to the maximum cardinality of any hyperedge (which may be smaller than the supermodular degree). They prove that obtaining similar approximation guarantees in the presence of negative hyperedges is $\mathcal{NP}$-hard. This serves as an explanation of why their model forbids negative hyperedges. Our results are to some extent in disagreement with this conclusion of [1]. Given a hypergraph representation of a set function with a given supermodular degree, adding negative hyperedges cannot increase the supermodular degree (in fact, it may cause the supermodular degree to decrease) and hence will not hurt our bounds on the approximation guarantees. This discrepancy between our results and those of [1] is explained by our requirement that set functions are nondecreasing, whereas the hardness of approximation results presented in [1] used set functions that are sometimes decreasing.

3.5.1 Independence systems

Extensive work has been conducted in recent years in the area of maximizing monotone submodular set functions subject to various constraints. We mention here the most relevant results. Historically, one of the very first problems examined was maximizing a monotone submodular set function subject to a matroid constraint. Several special cases of matroids and submodular functions were studied in [19, 57, 58, 67, 69], using the greedy approach. Recently, the general problem, with an arbitrary matroid and an arbitrary submodular set function, was given a tight approximation of $(1-1/e)$ by Calinescu et al. [15]. A matching lower bound is due to [77, 78].

The problem of maximizing a monotone submodular set function over the intersection of k

classification of [73]), but its supermodular degree is $\max[|A|, |B|] - 1$.

matroids was considered by Fisher et al. [48], who gave a greedy algorithm with an approximation guarantee of $1/(k+1)$, and stated that their proof extends to the more general class of k-systems using the outline of Jenkyns [67] (the extended proof is explicitly given by Calinescu et al. [15]). For k-intersection systems and k-exchange systems, this result was improved by Lee et al. [72] and Feldman et al. [45], respectively, to $1/(k+\varepsilon)$, for every constant $\varepsilon > 0$. The improvement is based on a local search approach that exploits exchange properties of the underlying combinatorial structure. Ward [87] further improved the approximation guarantee for k-exchange systems to $2/(k+3+\varepsilon)$ using a non-oblivious local search. However, for maximizing a monotone submodular set function over k-extendible independence systems (and the more general class of k-systems), the current best known approximation is still $1/(k+1)$ [48].

Other related lines of work deal with maximization of non-monotone submodular set functions (constrained or unconstrained) (see [14, 44, 86] for a few examples) and minimization of submodular set functions [53, 54, 62, 63].

The welfare maximization problem (or combinatorial auction) was studied in the context of many classes of utility (set) functions, including classes generalizing submodular set functions such as sub-additive [35] and fractionally sub-additive valuations [27]. For many of these classes a constant approximation algorithm is known [11, 26, 35, 39, 55] assuming access to a demand oracle, which given a vector of prices returns a set of elements maximizing the welfare of a player given these prices. However, when only a value oracle is available to the algorithm (*i.e.*, the only access the algorithm has to the utility functions is by evaluating them on a chosen set) one cannot get a better than a polynomial approximation guarantee, even for fractionally sub-additive valuations [27].

For maximization of set functions under general matroid constraints (as opposed to welfare maximization, that can be recast as maximizing a set function under a specific (partition) matroid constraint), we are not aware of previous work that addressed classes of set functions that are not submodular.

3.5.2 Subsequent work

Preliminary versions of the current section appeared in [36, 41]. We briefly discuss some subsequent work that further addresses the supermodular degree. Feldman and Izsak [42] introduced an online, secretary like, model based on supermodular dependencies. They designed online algorithms in this model for the secretary problem subject to a general matroid constraint or a cardinality constraint, with competitive ratios that depend on the supermodular degree of the input set function. These results appear in Section 4. Izsak and Svensson [65] considered the welfare maximization problem in an online adversarial model. Feldman, Friedler, Morgenstern and Reiner [40] introduced the hierarchy of Maximum over Positive-Supermodular-d valuation functions, where Positive-Supermodular-d valuation functions are valuation functions that are both positive hypergraph functions and have supermodular degree of at most d. They studied the efficiency of auctions for agents in the presence of complements, and upper bounded the price of anarchy as a function of the level in their hierarchy. Izsak [64] considered the problem of committee selection in the presence of synergies between the candidates. He used a notion of *joint supermodular degree* in the design of a voting rule for committee selection (see Section 5). Poularakis, Iosifidis, Smaragdakis and Tassiulas [80] studied the optimization of SDN upgrades. They showed that one of the problems they studied could be expressed as the maximization of a set function with a bounded supermodular degree. Poularakis et al. [80] evaluated the performance of their algorithms on real world data and showed improvements with respect to state-of-the-art methods.

A subsequent complexity measure, $\mathcal{MPH}$ (Maximum over Positive Hypergraphs), was introduced by Feige et al. [33]. The $\mathcal{MPH}$ measure uses $\mathcal{XOS}$ as the class of simplest functions, as well as the rank of hyperedges as the measure of distance from being $\mathcal{XOS}$. It turns out that any function that has a supermodular degree of d must be in $\mathcal{MPH} - (d+1)$. Feige et al. [33] designed an LP-based algorithm for the welfare maximization problem with approximation guarantees that are linear in $\mathcal{MPH}$. Their algorithm uses demand oracles (as opposed to value oracles), and therefore their results are incomparable to ours.

3.6 Approximation guarantee linear in supermodular degree

In this section we prove Theorem 3.5. Our result may be seen as an extension of a work of Lehmann, Lehmann and Nisan [73], who presented a greedy 2-approximation algorithm for submodular set functions.

3.6.1 The algorithm

The algorithm is greedy. In a given iteration, for every player p and item j, let $D_p^+(j)$ denote the set of items not yet allocated that have supermodular dependency with j with respect to v_p. The algorithm computes the player p and item j for which the marginal value for p (given the items that p already has) of the set $j \cup D_p^+(j)$ is maximized, and allocates $j \cup D_p^+(j)$ to p. For a full description of the algorithm, see Algorithm 3.2.

Algorithm 3.2 Greedily Approximate Welfare Maximization with Guarantee Linear in Supermodular Degree

Input:

- An instance $\mathcal{I}(P, M, v)$ of the welfare maximization problem.
- A value oracle and a supermodular dependency graph for each of the valuation functions.

Output: A solution with approximation guarantee $\frac{1}{d+2}$, where d is the supermodular degree of $\mathcal{I}$.

$Unallocated \leftarrow M$, $APX \leftarrow \emptyset$
while $Unallocated \neq \emptyset$ **do**
  $MaxMarginalUtility \leftarrow -1$
  for all $j \in Unallocated$, $p \in P$ **do**
    if $v_p(j, Dep_p^+(j) \cap Unallocated \mid \{j' \in M \mid (j' \mapsto p) \in APX\}) > MaxMarginalUtility$ **then**
      $MaxMarginalUtility \leftarrow v_p(j, Dep_p^+(j) \cap Unallocated \mid \{j' \in M \mid (j' \mapsto p) \in APX\})$
      $BestAllocation \leftarrow (\{j\} \cup (Dep_p^+(j) \cap Unallocated) \mapsto p)$
      $WinningPlayer \leftarrow p$, $AllocatedItem \leftarrow j$
    end if
  end for
  $APX \leftarrow APX \cup BestAllocation$
  $Unallocated \leftarrow Unallocated \setminus (AllocatedItem \cup Dep^+_{WinningPlayer}(AllocatedItem))$
end while

We show Algorithm 3.2 has approximation guarantee $1/(d+2)$, using a hybrid argument. This will prove Theorem 3.5.

of Theorem 3.5. Let OPT be an optimal solution with value opt and let APX be the output of Algorithm 3.2 with value $\approx$. For iteration i of the loop at line 2 of Algorithm 3.2, let APX_i be the allocations made at the first i iterations, let OPT_i be the allocations made by OPT for the items that have not yet been allocated and let $HYB_i = APX_i \cup OPT_i$ be a hybrid solution. Let HYB_i^p and OPT_i^p be the items allocated in HYB_i and OPT_i (respectively) to player $p \in P$. Note that $HYB_0 = OPT$ and $HYB_t = APX$, where t is the total number of iterations. We prove the following lemma:

Lemma 3.10. *Let i be an iteration of the loop at line 2 of Algorithm 3.2 and let p^* be the player to whom items are allocated at iteration i. Then,*

$$\begin{aligned}&(d+2)(v_{p^*}(APX_i^{p^*}) - v_{p^*}(APX_{i-1}^{p^*}))\\ &\geq \sum_{p=1}^{n} \left(v_p(OPT_{i-1}^p \mid APX_{i-1}^p) - v_p(OPT_i^p \mid APX_i^p)\right) \ .\end{aligned}$$

That is, the value lost by any iteration is bounded by $d+2$ times the value gained by the same iteration.

Proof. Let $x_i = v_{p^*}(APX_i^{p^*}) - v_{p^*}(APX_{i-1}^{p^*})$. Roughly speaking, we prove that:

1. For an item allocated to some $p' \neq p^*$ in OPT, the loss to the value of p' for not getting the item is at most x_i.

2. Having received items in the current iteration, the loss in marginal value of future items given to p^* is at most x_i.

The first "contributes" to the "damage" up to $(d+1)\cdot x_i$, since at most $d+1$ items are allocated at each iteration. The second "contributes" up to another x_i, and for any other player $HYB_{i-1} = HYB_i$. We prove the lemma formally. Let $j_1, \ldots j_{d'}$ be the items allocated at iteration i and let P' be the set of players $p' \neq p^*$ such that at least one of the items $j_1, \ldots, j_{d'}$ is allocated to p' in OPT_{i-1}. Let

$p' \in P'$ and let $\hat{j}_1, \ldots, \hat{j}_{d''} \in \{j_1, \ldots, j_{d'}\}$ be all the items of $j_1, \ldots, j_{d'}$, allocated to p' by OPT_{i-1}. Then,

$$
\begin{aligned}
& v_{p'}\left(OPT_{i-1}^{p'} \middle| APX_{i-1}^{p'}\right) - v_{p'}\left(OPT_{i}^{p'} \middle| APX_{i}^{p'}\right) \\
&= v_{p'}\left(\{\hat{j}_1, \ldots \hat{j}_{d''}\} \middle| OPT_{i}^{p'} \cup APX_{i-1}^{p'}\right) \\
&= \sum_{k=1}^{d''} v_{p'}\left(\hat{j}_k \middle| \bigcup_{k'=k+1}^{d''} \{\hat{j}_{k'}\} \cup OPT_{i}^{p'} \cup APX_{i-1}^{p'}\right) \\
&\leq d'' \cdot \max_{k=1}^{d''} v_{p'}\left(\hat{j}_k \middle| \bigcup_{k'=k+1}^{d''} \{\hat{j}_{k'}\} \cup OPT_{i}^{p'} \cup APX_{i-1}^{p'}\right) \\
&\leq d'' \cdot \max_{k=1}^{d''} v_{p'}\left(\hat{j}_k \middle| \left(\left(\bigcup_{k'=k+1}^{d''} \{\hat{j}_{k'}\} \cup OPT_{i}^{p'}\right) \cap Dep_{p'}^{+}(\hat{j}_k)\right) \cup APX_{i-1}^{p'}\right) \\
&\leq d'' \cdot \max_{k=1}^{d''} v_{p'}\left(\hat{j}_k \cup \left(\left(\bigcup_{k'=k+1}^{d''} \{\hat{j}_{k'}\} \cup OPT_{i}^{p'}\right) \cap Dep_{p'}^{+}(\hat{j}_k)\right) \middle| APX_{i-1}^{p'}\right) \\
&\leq d'' \cdot \max_{k=1}^{d''} v_{p'}\left(\hat{j}_k \cup \left(Dep_{p'}^{+}(\hat{j}_k) \setminus \bigcup_{p \in P} APX_{i-1}^{p}\right) \middle| APX_{i-1}^{p'}\right) \\
&\leq d'' \cdot v_{p^*}\left(j_1, \ldots, j_{d'} \middle| APX_{i-1}^{p^*}\right) \\
&= d'' \cdot \left(v_{p^*}\left(APX_{i}^{p^*}\right) - v_{p^*}\left(APX_{i-1}^{p^*}\right)\right)
\end{aligned}
$$

where, the first equality follows by definitions and by observing that for any player $p \neq p^*$, $APX_{i-1}^{p} = APX_{i}^{p}$; the second by definitions. The first inequality is trivial; the second follows by Definition 3.3; the third and fourth by monotonicity; the fifth by line 5 of Algorithm 3.2. The last equality follows by definitions. Since there are only $d' \leq d+1$ items allocated, and since for any player $p \notin P' \cup \{p^*\}$, $HYB_{i}^{p} = HYB_{i-1}^{p}$, we get,

$$
\begin{aligned}
& (d+1)(v_{p^*}(APX_{i}^{p^*}) - v_{p^*}(APX_{i-1}^{p^*})) \\
& \geq \sum_{p \in P \setminus \{p^*\}} \left(v_p(OPT_{i-1}^{p} \mid APX_{i-1}^{p}) - v_p(OPT_{i}^{p} \mid APX_{i}^{p})\right) . \qquad (1)
\end{aligned}
$$

For player p^*, we have by monotonicity $v_{p^*}(HYB_i^{p*}) \geq v_{p^*}(HYB_{i-1}^{p*})$. Hence,

$$v_{p^*}(OPT_{i-1}^{p^*} \mid APX_{i-1}^{p^*}) + v_{p^*}(APX_{i-1}^{p^*}) \leq v_{p^*}(OPT_i^{p^*} \mid APX_i^{p^*}) + v_{p^*}(APX_i^{p^*})$$

and then,

$$v_{p^*}(OPT_{i-1}^{p^*} \mid APX_{i-1}^{p^*}) - v_{p^*}(OPT_i^{p^*} \mid APX_i^{p^*}) \leq v_{p^*}(APX_i^{p^*}) - v_{p^*}(APX_{i-1}^{p^*}) \ .$$

This and (1) prove Lemma 3.10. □

We use Lemma 3.10 to complete the proof of Theorem 3.5.

$$\begin{aligned}
\mathsf{opt} &= \sum_{p=1}^{n} v_p(OPT_0^p \mid APX_0^p) \\
&= \sum_{p=1}^{n} \left(v_p(OPT_0^p \mid APX_0^p) - v_p(OPT_t^p \mid APX_t^p)\right) \\
&= \sum_{p=1}^{n} \sum_{i=0}^{t-1} \left(v_p(OPT_i^p \mid APX_i^p) - v_p(OPT_{i+1}^p \mid APX_{i+1}^p)\right) \\
&\leq (d+2) \cdot \sum_{i=1}^{t} x_i = (d+2) \cdot \approx \ ,
\end{aligned}$$

where the first three equalities follow by definition of hybrid solution and the inequality by Lemma 3.10.

This proves Theorem 3.5. □

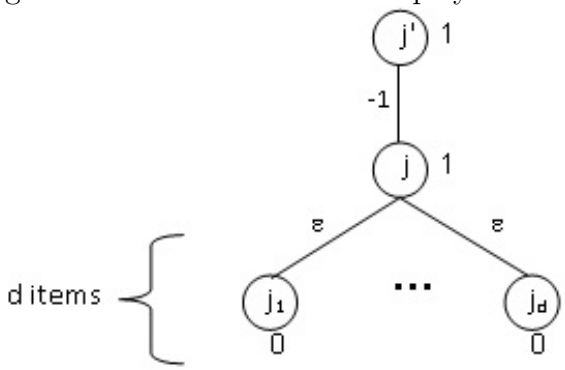
3.6.2 A tight example

The following example shows that Algorithm 3.2 may return a solution with value arbitrarily close to $\frac{1}{d+2}$, which matches the upper bound we proved in Theorem 3.5.

Example 3.11. *Let $\mathcal{I}(P, M, v)$ be an instance of the welfare maximization problem with players $P = \{1, 2\}$, items $M = \{j, j_1, \ldots, j_d, j'\}$ and valuation functions v_1 as in the hypergraph representation shown in Figure 1 below and $v_2(S) = |S \setminus \{j'\}|$, for any $S \subseteq M$. It is easy to observe the optimal solution gives all the items except j' to player 2 and j' to player 1. The value of this solution is*

$d+2$. On the other hand, Algorithm 3.2 gives all the items except j' to player 1, and gives j' to an arbitrary player. This solution has value of only $1+d\cdot\varepsilon$, which tends to 1 as ε decreases.

Figure 1: Valuation function of player 1



3.7 Approximation guarantee linear in dependency degree

We discuss two possible alternatives for proving Theorem 3.6. One is adapting Algorithm 3.2, as we briefly discuss in this section and the other is presented in Section 3.10. We sketch a possible adaptation of Algorithm 3.2. Recall that Algorithm 3.2 does the following in each iteration: for each player and item, it calculates the marginal value of the item and all its unallocated supermodular dependencies with respect to the allocated items of the player. A player and items with maximum value are selected. The modification here is twofold. Firstly, for each player and item, one considers not only the subset of *all* unallocated dependencies of the item, but *any possible subset* of unallocated dependencies. Secondly, one does not consider the marginal value of an item *together with its dependencies* with respect to the previously allocated items, but *only of the item*, with respect to the subset of dependencies under consideration and the previously allocated items, together. It can be shown that the "damage" for any player, caused by an allocation of a single item to another player, is bounded by the "profit" of the iteration "damaging" it, and also (unlike the case of supermodular dependency) that the player getting the allocation has no "damage" at all in future iterations.

The other alternative, fully presented in Section 3.10, is designing a different greedy algorithm. That algorithm has the somewhat surprising property that when considering which items to add

to a player, it completely ignores the items that the player already has (despite the fact that these items determine the marginal values for new items).

3.8 k-Extendible system

In this section we prove Theorems 3.7 and 3.9. The proof of Theorem 3.8 uses similar ideas and is deferred to Section 3.12 for readability.

3.8.1 Algorithm for k-extendible system (Proof of Theorem 3.7)

We consider in this section Algorithm 3.3, and prove it fulfills the guarantees of Theorem 3.7.

Algorithm 3.3 Extendible System Greedy$(f, \mathcal{I})$

Input:

- An instance $\mathcal{I}(M, f, C)$ of constrained function maximization, where C is a k-extendible system.
- A value queries oracle and a supermodular dependency graph for f.

Output: A solution with approximation guarantee $\frac{1}{k(d+1)+1}$, where d is the dependency degree of f.

1: Initialize: $APX_0 \leftarrow \varnothing$, $i \leftarrow 0$
2: **while** APX_i is not a base **do**
3: $\quad i \leftarrow i+1$
4: $\quad$ Let $u_i \in M \setminus APX_{i-1}$ and $D^+_{best}(u_i) \subseteq \mathcal{D}^+(u_i)$ be a pair of an element and a set maximizing $f(D^+_{best}(u_i) + u_i \mid APX_{i-1})$ among all pairs obeying $APX_{i-1} \cup D^+_{best}(u_i) + u_i \in \mathcal{I}$.
5: $\quad APX_i \leftarrow APX_{i-1} \cup D^+_{best}(u_i) + u_i$
6: $\quad$ Return APX_i
7: **end while**

First, let us give some intuition. Let APX be an approximate solution and let OPT be an arbitrary optimal solution. Originally, before the algorithm adds any elements to APX, it still can be that it chooses to add all of the elements of OPT (together) to APX, and get an optimal solution. At each iteration, when adding elements to APX, this possibility might get ruined for some elements of OPT. The reason for that is that given the elements that have already been added to APX, the independence constraint might exclude the possibility of adding some of the elements

of OPT. If we want to keep the invariant that all the elements of OPT can be added to APX, then we might have to discard some elements of OPT. This discard potentially decreases the value of OPT, and therefore, can be seen as the damage incurred by the iteration. Note that by the definition of a k-extendible system, we do not have to discard more than k elements for every element we add. That is, at every iteration, only up to $k(\mathcal{D}_f^+ + 1)$ elements must be discarded. Therefore, if we manage to upper bound the damage of discarding a single element by the benefit of the allocation at the same iteration, we get the desired bound.[8] Recall that the supermodular dependencies of an element are exactly the elements that may increase its marginal value. Therefore, when discarding an element from OPT, the maximum damage is bounded by the marginal value of this element with respect to its supermodular dependencies in OPT. But, as any subset of OPT can be added to APX, the greedy choice of Algorithm 3.3 explicitly takes into account the possibility of adding this element and its supermodular dependencies to APX. If another option is chosen, it must have at least the same immediate benefit.

We now give a formal proof for Theorem 3.7. Let us begin with the following observation.

Observation 3.12. *Whenever APX_i is not a base, there exists an element $u \in M \setminus APX_i$ for which $APX_i \cup \varnothing + u \in \mathcal{I}$ (note that $\varnothing \subseteq \mathcal{D}^+(u)$). Hence, Algorithm 3.3 always outputs a base.*

Throughout this section, we denote $d = \mathcal{D}_f^+$. Our proof is by a hybrid argument. That is, we have a sequence of hybrid solutions, one per iteration, where the first hybrid contains an optimal solution (and hence, has an optimal value), and the last hybrid is our approximate solution.[9] Roughly speaking, we show the following:

1. By adding each element to the approximate solution, we do not lose more than k elements of the iteration's hybrid (note that we add to our solution at most $d + 1$ elements at any given iteration). This is formalized in Lemma 3.13, and the proof is based on Definition 2.11 (k-extendible system).

2. The damage from losing an element of an iteration's hybrid is bounded by the profit the

[8] The other additive 1 in the denominator of the approximation guarantee comes from the fact that by OPT's value we actually mean its marginal contribution to APX. In this sense, addition of elements to APX also might reduce the value of OPT.

[9] Actually, the last hybrid is defined as *containing* our approximate solution, but, as our approximate solution is a base, the hybrid must be exactly equal to it.

algorithm gains at that iteration. This is formalized in Lemma 3.14, and the proof is based on Definition 3.3 (supermodular dependency set).

In conclusion, we show that when moving from one hybrid to the next, we lose no more than $k(d+1)$ times the profit at the respective iteration.

Let us formalize the above argument. Let ℓ be the number of iterations performed by Algorithm 3.3, *i.e.*, ℓ is the final value of i. We recursively define a series of $\ell+1$ hybrid solutions as follows.

- HYB_0 is a base containing OPT. By monotonicity, $f(HYB_0) = f(OPT)$.
- For every $1 \leq i \leq \ell$, HYB_i is a maximum size independent subset of $HYB_{i-1} \cup APX_i$ containing APX_i.

Lemma 3.13. *For every iteration* $1 \leq i \leq \ell$, $|HYB_{i-1} \setminus HYB_i| \leq k \cdot |APX_i \setminus HYB_{i-1}| \leq k(d+1)$.

Proof. Let us denote the elements of $APX_i \setminus HYB_{i-1}$ by $v_1, v_2, \ldots, v_r$. We prove by induction that there exists a collection of sets $Y_1, Y_2, \ldots, Y_r$, each of size at most k, such that: $Y_j \subseteq HYB_{i-1} \setminus (APX_{i-1} \cup \{v_h\}_{h=1}^{j-1})$ and $HYB_{i-1} \setminus (\cup_{h=1}^{j} Y_h) \cup \{v_h\}_{h=1}^{j} \in \mathcal{I}$ for every $0 \leq j \leq r$. For ease of notation, let us denote $Y_1^j = \cup_{h=1}^{j} Y_h$ and $v_1^j = \{v_h\}_{h=1}^{j}$. Using this notation, the claim we want to prove can be rephrased as follows: there exists a collection of sets $Y_1, Y_2, \ldots, Y_r$, each of size at most k, such that: $Y_j \subseteq HYB_{i-1} \setminus (APX_{i-1} \cup v_1^{j-1})$ and $(HYB_{i-1} \setminus Y_1^j) \cup v_1^j \in \mathcal{I}$ for every $0 \leq j \leq r$.

For $j = 0$ the claim is trivial since $HYB_{i-1} \in \mathcal{I}$. Thus, let us prove the claim for j assuming it holds for $j-1$. By the induction hypothesis, $(HYB_{i-1} \setminus Y_1^{j-1}) \cup v_1^{j-1} \in \mathcal{I}$. On the other hand, $APX_{i-1} \cup v_1^{j-1}$ is a subset of this set which is independent even if we add v_j to it. Since $(M, \mathcal{I})$ is a k-extendible system, this implies the existence of a set Y_j of size at most k such that:

$$Y_j \subseteq [(HYB_{i-1} \setminus Y_1^{j-1}) \cup v_1^{j-1}] \setminus [APX_{i-1} \cup v_1^{j-1}] \subseteq HYB_{i-1} \setminus (APX_{i-1} \cup v_1^{j-1}) \ ,$$

and:

$$[(HYB_{i-1} \setminus Y_1^{j-1}) \cup v_1^{j-1}] \setminus Y_j + v_j \in \mathcal{I} \Rightarrow (HYB_{i-1} \setminus Y_1^{j}) \cup v_1^{j} \in \mathcal{I} \ ,$$

which completes the induction step. Thus, $(HYB_{i-1} \setminus Y_1^r) \cup v_1^r \in \mathcal{I}$ is a subset of $HYB_{i-1} \cup APX_i$ which contains APX_i and has a size of at least: $|HYB_{i-1}| - rk + r$. On the other hand, HYB_i is a maximum size independent subset of $HYB_{i-1} \cup APX_i$, and thus: $|HYB_i| \geq |HYB_{i-1}| - rk + r$. Finally, all elements of HYB_i belong also to HYB_{i-1} except, maybe, the elements of $APX_i \setminus APX_{i-1}$. Hence,

$$|HYB_{i-1} \setminus HYB_i| \leq |HYB_{i-1}| - |HYB_i| + |APX_i \setminus APX_{i-1}| \leq |HYB_{i-1}| - (|HYB_{i-1}| - rk + r) + r = rk \ .$$

Lemma 3.13 now follows, since $r \leq d + 1$. □

The following lemma upper bounds the loss of moving from one hybrid to the next one.

Lemma 3.14. *Let i be an iteration of the loop at line 4 of Algorithm 3.3. Then,*

$$\begin{aligned} &(k(d+1)+1)(f(APX_i) - f(APX_{i-1})) \\ &\geq (f(OPT_{i-1} \mid APX_{i-1}) - f(OPT_i \mid APX_i)) \ . \end{aligned}$$

That is, the value lost by any iteration is bounded by $(k(d+1)+1)$ times the value gained by the same iteration.

Proof. The proof of Lemma 3.14 is similar to that of Lemma 3.10, but here, we do not have different players, when one can only gain value and the others can only lose value. This means, we need to take all into account in our single input valuation function. This means in particular that we generally do not have $APX_{i-1} = APX_i$ that we used in the proof of Lemma 3.10. Therefore, we first bound the loss $f(OPT_{i-1} \mid APX_{i-1}) - f(OPT_i \mid APX_{i-1})$ and then the loss $f(OPT_i \mid APX_{i-1} - f(OPT_i \mid APX_i)$. The first corresponds to the loss we prove for the player losing the items in Lemma 3.10 and the latter to the player getting the items.

Let $j_1, \ldots j_{d'}$ be the items allocated at iteration i and Let $\hat{j}_1, \ldots, \hat{j}_{d''}$ be all the items of $OPT_{i-1} \setminus$

OPT_i (which are exactly the items of $HYBi-1 \setminus HYB_i$, as well). Then,

$$
\begin{aligned}
& f\left(OPT_{i-1} | APX_{i-1}\right)-f\left(OPT_i | APX_{i-1}\right) \\
& =f\left(\{\hat{j}_1, \ldots \hat{j}_{d''}\} \middle| OPT_i \cup APX_{i-1}\right) \\
& =\sum_{k=1}^{d''} f\left(\hat{j}_k \middle| \bigcup_{k'=k+1}^{d''}\{\hat{j}_{k'}\} \cup OPT_i \cup APX_{i-1}\right) \\
& \leq d'' \cdot \max_{k=1}^{d''} f\left(\hat{j}_k \middle| \bigcup_{k'=k+1}^{d''}\{\hat{j}_{k'}\} \cup OPT_i \cup APX_{i-1}\right) \\
& \leq d'' \cdot \max_{k=1}^{d''} f\left(\hat{j}_k \middle| \left(\left(\bigcup_{k'=k+1}^{d''}\{\hat{j}_{k'}\} \cup OPT_i\right) \cap Dep_{p'}^{+}(\hat{j}_k)\right) \cup APX_{i-1}\right) \\
& \leq d'' \cdot \max_{k=1}^{d''} f\left(\hat{j}_k \cup \left(\left(\bigcup_{k'=k+1}^{d''}\{\hat{j}_{k'}\} \cup OPT_i\right) \cap Dep_{p'}^{+}(\hat{j}_k)\right) \middle| APX_{i-1}\right) \\
& \leq d'' \cdot \max_{k=1}^{d''} f\left(\hat{j}_k \cup \left(Dep_{p'}^{+}(\hat{j}_k) \setminus \bigcup_{p \in P} APX_{i-1}^{p}\right) \middle| APX_{i-1}\right) \\
& \leq d'' \cdot f\left(j_1, \ldots, j_{d'} | APX_{i-1}\right) \\
& =d'' \cdot\left(f\left(APX_i\right)-f\left(APX_{i-1}\right)\right)
\end{aligned}
$$

where, the first equality follows by definitions; the second by definitions. The first inequality is trivial; the second follows by Definition 3.3; the third and fourth by monotonicity; the fifth by line 4 of Algorithm 3.3. The last equality follows by definitions.

Combining it with Lemma 3.13, we get,

$$(k(d+1)) \cdot (f(APX_i) - f(APX_{i-1})) \geq f(OPT_{i-1} \mid APX_{i-1}) - f(OPT_i \mid APX_{i-1}) \ . \qquad (2)$$

We now bound $f(OPT_i \mid APX_{i-1})) - f(OPT_i \mid APX_i)$. By monotonicity, we have $f(OPT_i \cup APX_i) \geq f(OPT_i \cup APX_{i-1})$. Hence,

$$f(OPT_i \mid APX_{i-1}) + f(APX_{i-1}) \leq f(OPT_i \mid APX_i) + f(APX_i)$$

and then,

$$f(OPT_i \mid APX_{i-1}) - f(OPT_i \mid APX_i) \leq f(APX_i) - f(APX_{i-1}) \ .$$

This and (2) prove Lemma 3.14. □

We use Lemma 3.14 to complete the proof of Theorem 3.7.

$$\begin{aligned}
\mathsf{opt} &= f(OPT_0 \mid APX_0) \\
&= f(OPT_0 \mid APX_0) - f(OPT_t \mid APX_t) \\
&= \sum_{i=0}^{t-1} (f(OPT_i \mid APX_i) - f(OPT_{i+1} \mid APX_{i+1})) \\
&\leq (k(d+1)+1) \cdot \sum_{i=1}^{t} (f(APX_i) - f(APX_{i-1})) \\
&= (k(d+1)+1) \cdot \approx \ ,
\end{aligned}$$

where the first three equalities follow by definition of hybrid solution and the inequality by Lemma 3.14.

This proves Theorem 3.7.

A Tight Example for Algorithm 3.3

We present an example showing that our analysis of Algorithm 3.3 is tight even when the independence system $(M, \mathcal{I})$ belongs to k-intersection (recall that any independence system that is k-intersection is also k-extendible, but not vice versa).

Proposition 3.15. *For every $k \geq 1$, $d \geq 0$ and $\varepsilon > 0$, there exists a k-intersection independence system $(M, \mathcal{I})$ and a function $f : 2^M \to \mathbb{R}^+$ with $\mathcal{D}_f^+ = d$ for which Algorithm 3.3 produces a $(1+\varepsilon)/(k(d+1)+1)$ approximation.*

The rest of this section is devoted for constructing the independence system guaranteed by Proposition 3.15. Let $\mathcal{T}$ be the collection of all sets $T \subseteq \{1, 2, \ldots, k+1\} \times \{0, 1, \ldots, (d+1)(k+1)-1\}$ obeying the following properties:

- For every $1 \leq i \leq k+1$, there exists exactly one x such that T contains the pair (i, x).
- At least one pair (i, x) in T has $x \leq d$.
- Let x_{k+1} be such that $(k+1, x) \in T$. Then $x_{k+1} = 0$ or $x_{k+1} > d$.

Intuitively, the first requirement means that we can view a set $T \in \mathcal{T}$ as a point in a $(k+1)$-dimensional space. The other two requirements make some points illegal. For example, for $k = 1$ the space is a $2(d+1) \times 2(d+1)$ grid, and the legal points are the ones that are either in row 0 or in one of the rows $d+1$ to $2(d+1)-1$ and one of the columns 0 to d. Two examples of $\mathcal{T}$ can be seen in Figure 2.

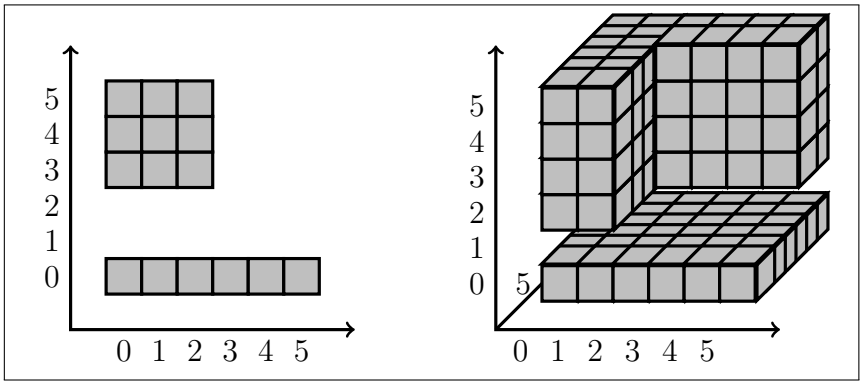


Figure 2: Graphical representations of $\mathcal{T}$ for two configurations: $k = 1, d = 2$ and $k = 2, d = 1$. In both cases the last coordinate corresponds to the top-down axis.

Let M be the ground set $\{u_T \mid T \in \mathcal{T}\}$. We define k matroids on this ground set as follows. For every $1 \leq i \leq k$, $\mathcal{M}_i = (M, \mathcal{I}_i)$, where a set $S \subseteq M$ belongs to $\mathcal{I}_i$ if and only if for every $0 \leq x < (d+1)(k+1)$, $|\{u_T \in S \mid (i, x) \in T\}| \leq 1$. One can easily verify that $\mathcal{M}_i$ is a partition matroid. The independence system we construct is the intersection of these matroids, *i.e.*, it is $(M, \mathcal{I})$, where $\mathcal{I} = \bigcap_{i=1}^{k} \mathcal{I}_i$. Next, we define the objective function $f : 2^M \to \mathbb{R}^+$ as follows.

$$f'(S) = \sum_{x=0}^{(d+1)(k+1)-1} \min\{1, |\{u_T \in S \mid (k+1, x) \in T\}|\} \ .$$

That is, for $k = 1$, f' gains a value of 1 for every row that was "hit" by an element. For every

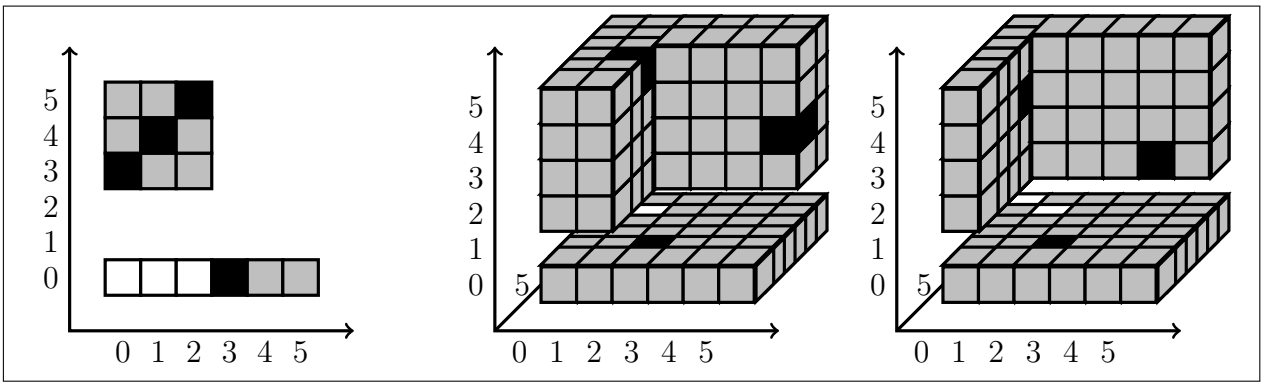


Figure 3: The solution produced by Algorithm 3.3 and the set S^* for the two examples presented in Figure 2. The second example is depicted twice, once with some of the elements removed to make more elements visible. The set S^* is denoted by black squares and the solution of Algorithm 3.3 is denoted by white squares. Note that in both solutions no two elements share a row or a depth (when there is a depth). In S^* no two points share a height, and thus, every element in S^* contributes 1 to the value. On the other hand, in the algorithm's solution all the elements share height, and thus its *overall value* is 1 by f' and $1+\varepsilon$ by f.

$0 \leq x \leq d$, let $\hat{T}(x) = \{(k+1,0)\} \cup \{(i,x)\}_{i=1}^{k}$ (note that $\hat{T}_x \in \mathcal{T}$).

$$f(S) = \begin{cases} f'(S) + \varepsilon & \text{if } \{u_{\hat{T}(x)}\}_{x=0}^{d} \subseteq S \\ f'(S) & \text{otherwise} \end{cases} .$$

One can check that f' is a non-negative monotone submodular function, and thus, $\mathcal{D}_f^+ = d$.

Claim 3.16 argues that Algorithm 3.3 outputs a poor solution for the above independence system and objective function. The discussion after the claim presents an independent set S^* of large value. Examples for both the solution of the algorithm and the set S^* can be found in Figure 3.

Claim 3.16. *Given the above constructed independence system $(M, \mathcal{I})$ and objective function f, Algorithm 3.3 outputs a solution of value $1+\varepsilon$.*

Proof. Consider the first iteration of Algorithm 3.3. Let $u_T \in M$. If $T \notin \{\hat{T}(x)\}_{x=0}^{d}$, then $f(u_T \mid S) = f'(u_T \mid S)$ for every set $S \subseteq M$, and thus, $\mathcal{D}^+(u_T) = \varnothing$ because f' is a submodular function. Hence, for every such u_T, we get: $f(\mathcal{D}^+(u_T) + u_T) = 1$. Consider now the case $T \in \{\hat{T}(x)\}_{x=0}^{d}$. In this case, clearly, $\mathcal{D}^+(u_T) = \{u_{\hat{T}(x)}\}_{x=0}^{d} - u_T$, and thus, $f(\mathcal{D}^+(u_T) + u_T) = 1+\varepsilon$. In conclusion, Algorithm 3.3 picks exactly the elements of $\{u_{\hat{T}(x)}\}_{x=0}^{d}$ to its solution at the first iteration.

To complete the proof, we show that Algorithm 3.3 cannot increase the value of its solution at the next iterations. Consider an arbitrary element $u_T \in M \setminus \{u_{\hat{T}(x)}\}_{x=0}^{d}$. By definition, T must contain a pair (i,x) such that $0 \leq x \leq d$. There are two cases:

- If $i \neq k+1$, then u_T cannot coexist in an independent set of $\mathcal{M}_i$ with $u_{\hat{T}(x)}$ because both correspond to sets containing the pair (i,x).
- If $i = k+1$, then $x = 0$ because $u_T \in M$.

From the above analysis, we get that all elements added to the solution after the first iteration contain the pair $(k+1,0)$ (and thus, no other pair of the form $(k+1,x)$). Hence, they do not increase the value of either f' or f. □

To prove Proposition 3.15, we still need to show that $(M,\mathcal{I})$ contains an independent set of a high value. Consider the set $S^* = \{u_{T^*(j)}\}_{j=0}^{k(d+1)}$, where $T^*(j) \stackrel{\text{def}}{=} \{(i,x) \mid 1 \leq i \leq k+1 \text{ and } x = (i(d+1)-j) \bmod (d+1)(k+1)\}$.

Claim 3.17. *$S^* \subseteq M$.*

Proof. We need to show that for every $0 \leq j \leq k(d+1)$, $u_{T^*(j)} \in M$. For $j = 0$, $(k+1,0) \in T^*(0)$, which completes the proof. Thus, we may assume from now on $1 \leq j \leq k(d+1)$, and let $i = \lceil j/(d+1) \rceil$. Clearly $1 \leq i \leq k$ and $T^*(j)$ contains the pair (i,x) for:

$$x = (i(d+1)-j) \bmod (d+1)(k+1) = (\lceil j/(d+1) \rceil \cdot (d+1) - j) \bmod (d+1)(k+1) \ .$$

To conclude Claim 3.17, we need to show that $0 \leq x \leq d$. This follows since $\lceil j/(d+1) \rceil \cdot (d+1) - j \geq (j/(d+1)) \cdot (d+1) - j = 0$ and $\lceil j/(d+1) \rceil \cdot (d+1) - j < [j/(d+1)+1] \cdot (d+1) - j = d+1$. □

Claim 3.18. *For every two values $0 \leq j_1 < j_2 \leq k(d+1)$, $T^*(j_1) \cap T^*(j_2) = \varnothing$. Hence $S^* \in \mathcal{I}$ and $f(S^*) \geq f'(S^*) = |S^*| = k(d+1)+1$.*

Proof. Assume towards contradiction that $(i,x) \in T^*(j_1) \cap T^*(j_2)$. Then, modulo $(d+1)(k+1)$,

the following equivalence must hold:

$$(i(d+1)-j_1) \equiv (i(d+1)-j_2) \Rightarrow j_1 \equiv j_2 \ ,$$

which is a contradiction since $j_1 \neq j_2$ and they are both in the range $[0, k(d+1)]$. □

3.8.2 Hardness (Proof of Theorem 3.9)

Before proving Theorem 3.9 let us state the hardness result of [60] given by Theorem 3.19. In the r-Dimensional Matching problem one is given an r-sided hypergraph $G = (\bigcup_{i=1}^{r} V_i, E)$, where every edge $e \in E$ contains exactly one vertex of each set V_i. The objective is to select a maximum size matching $M \subseteq E$, *i.e.*, a subset $M \subseteq E$ of edges which are pairwise disjoint.

Theorem 3.19 (Hazan et al. [60])**.** *It is $\mathcal{NP}$-hard to approximate r-Dimensional Matching to within a factor $\frac{c \log r}{r}$, for some constant $c > 0$ in polynomial time, even if r is a constant.*

Theorem 3.9 follows by combining Theorem 3.19 with the following lemma.

Lemma 3.20. *Any instance of r-Dimensional Matching can be represented as maximizing a monotone function f with $\mathcal{D}_f^+ = \mathcal{D}_f \leq d$ over a k-intersection set system for every $d \geq 0$ and $k \geq 1$ obeying $r \leq k(d+1)$.*

Proof. For simplicity, assume $r = k(d+1)$. Let $G = (\bigcup_{i=1}^{r} V_i, E)$ be the graph representing the r-Dimensional Matching instance. We first construct a new graph G' as follows. For every edge $e \in E$ and $1 \leq j \leq d+1$, let $e(j) = e \cap (\bigcup_{i=(j-1)k+1}^{jk} V_i)$, *i.e.*, $e(j)$ is the part of e hitting the vertex sets $V_{(j-1)k+1}, \ldots, V_{jk}$. The edges of the new graph $G' = (\bigcup_{i=1}^{r} V_i, E')$ are then defined as all edges that can be obtained this way. More formally:

$$E' = \{e(j) \mid e \in E \text{ and } 1 \leq j \leq d+1\} \ .$$

It is easy to see that the original instance of r-Dimensional Matching is equivalent to the problem

of finding a matching in G' maximizing the objective function $f : 2^{E'} \to \mathbb{R}^+$ defined as follows.

$$f(APX) = \sum_{e \in E} \left\lfloor \frac{|APX \cap \{e(j) \mid 1 \leq j \leq d+1\}|}{d+1} \right\rfloor \ .$$

Moreover, $\mathcal{D}_f = \mathcal{D}_f^+ = d$. Thus, to complete the proof we only need to show that the set of all legal matchings of G' can be represented as a k-intersection independence system.

Consider the following partition of the vertices of G'. For every $1 \leq j \leq k$, $V'_j = \bigcup_{i=0}^{d} V_{j+ki}$. Observe that each edge of G' contains exactly one vertex of V'_j. Hence, the constraint that no two edges intersect on a node of V'_j can be represented by the partition matroid $M_j = (E', \mathcal{I}_j)$ defined as following. A set $APX \subseteq E'$ is in $\mathcal{I}_j$ if and only if no two edges of APX intersect on a node of V'_j. The set of legal matchings of G' is, then, exactly $\bigcap_{j=1}^{k} \mathcal{I}_j$. □

3.9 Symmetry of dependency relations

Lemma 3.21 (Symmetry)**.** *Let $p \in P$ and let $j_1, j_2 \in M$ be such that $j_1 \to_p j_2$. Then $j_2 \to_p j_1$. In other words, the relation '$\to_p$' is symmetric.*

Note that the same is true for the relation $\to^+$ and that the proof is exactly the same.

Proof. Let S be such that $j_1 \xrightarrow{S} j_2$. We show that

$$j_2 \xrightarrow{S \setminus \{j_2\} \cup \{j_1\}} j_1 \ .$$

From Definition 2.1, on one hand,

$$v(\{j_1\} \cup S) = v(j_1 \mid S) + v(j_2 \mid S \setminus \{j_2\}) + v(S \setminus \{j_2\}) \ ,$$

and on the other hand,

$$v(\{j_1\} \cup S) = v(j_2 \mid S \setminus \{j_2\} \cup \{j_1\}) + v(j_1 \mid S \setminus \{j_2\}) + v(S \setminus \{j_2\}) \ .$$

By subtracting $v(S \setminus \{j_2\})$, we get:

$$v(j_1 \mid S) + v(j_2 \mid S \setminus \{j_2\}) = v(j_2 \mid S \setminus \{j_2\} \cup \{j_1\}) + v(j_1 \mid S \setminus \{j_2\}) \ .$$

Since $j_1 \overset{S}{\to} j_2$ means $v(j_1 \mid S) \neq v(j_1 \mid S \setminus \{j_2\})$, we get $v(j_2 \mid S \setminus \{j_2\}) \neq v(j_2 \mid S \setminus \{j_2\} \cup \{j_1\})$. The latter is exactly the definition of $j_2 \xrightarrow{S \setminus \{j_2\} \cup \{j_1\}} j_1$, as desired. □

3.10 A greedy $\frac{1}{d+1}$-approximation algorithm for dependency degree at most d

In this section we prove Theorem 3.6.

3.10.1 The algorithm

The algorithm is Algorithm 3.4.

Intuitively, Algorithm 3.4 promises at each iteration that the most possibly contributing item will have its full contribution in the approximated solution. Thus, any mislocated item in an optimal solution cannot "damage" it in more than the benefit of the iteration "damaging" it. We conclude the approximation guarantee by observing no more than $d+1$ items are allocated at each iteration.

Few remarks are in place.

Remark 3.1. *Algorithm 3.4 does not look at all at the value of already allocated items (neither at new inspected items relatively to them nor at the whole sub-allocation). Note that an approach looking* only *at the marginal value with respect to already allocated items without looking on "forward" dependencies does not work.*[10]

Remark 3.2. *Algorithm 3.4 discards unallocated dependencies (at line 13). This is to ensure any selected item will have its inspected marginal value. In other words, we ensure that the only dependencies of an item at the rest of the solution (i.e., the unallocated part) will be its optimal*

[10]For example, we may have two items and two players, where the first player has set function $f(S) = |S|$ and the second has set function giving ∞ to both items together and 0 otherwise. An algorithm that looks only "backward" will allocate both items to player 1 and gain value 2 where an optimal solution has value ∞.

Algorithm 3.4 Greedily Approximate Welfare Maximization with Guarantee Linear in Dependency Degree

Input:

- An instance $\mathcal{I}(P, M, v)$ of the welfare maximization problem.
- A value oracle and a supermodular dependency graph for each of the valuation functions of $\mathcal{I}$.

Output: A solution with approximation guarantee $\frac{1}{d+1}$, where d is the dependency degree of $\mathcal{I}$.

1: $Unallocated \leftarrow M$, $APX \leftarrow \emptyset$
2: **while** $Unallocated \neq \emptyset$ **do**
3: $\quad MaxMarginalUtility \leftarrow -1$
4: $\quad$ **for all** $j \in Unallocated$, $p \in P$, $S' \subseteq (Dep_p(j) \cap Unallocated)$ **do**
5: $\quad\quad$ **if** $v_p(j \mid S') > MaxMarginalUtility$ **then**
6: $\quad\quad\quad MaxMarginalUtility \leftarrow v_p(j \mid S')$
7: $\quad\quad\quad BestAllocation \leftarrow (\{j\} \cup S' \mapsto p)$
8: $\quad\quad\quad WinningPlayer \leftarrow p$, $AllocatedItem \leftarrow j$, $AllocatedOptimalDependencies \leftarrow S'$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: $\quad APX \leftarrow APX \cup BestAllocation$
12: $\quad Unallocated \leftarrow Unallocated \setminus (AllocatedItem \cup AllocatedOptimalDependencies)$
13: $\quad Unallocated \leftarrow Unallocated \setminus Dep_{WinningPlayer}(AllocatedItem)$ {Discard also unallocated dependencies of j}
14: **end while**

dependencies, as inspected at lines 4-10. Of course, because of monotonicity, we may add the "discarded" items to any player we wish (for example to the player we allocated to the rest or each item to its best possibility, in any order). The tight example we will show is tight also for any of these possibilities.

of Theorem 3.6. Let OPT be an optimal solution with value opt and let APX be an output of Algorithm 3.4 with value $\approx$. Let t be the number of iterations of the "while" loop at line 2 of the run created APX. We layer opt and $\approx$ by writing both of them by iterations of Algorithm 3.4.

$$\mathsf{opt} = \sum_{i=1}^{t} \sum_{k=1}^{d+1} v_{p_{opt}(i,k)}(j_{i,k} \mid OPT_{p_{opt}(i,k)} \setminus \bigcup_{i'=1}^{i-1} \bigcup_{k'=1}^{d+1} \{j_{i',k'}\} \setminus \bigcup_{k'=1}^{k} \{j_{i,k'}\})$$

$$\approx= \sum_{i=1}^{t}\sum_{k=1}^{d+1} v_{p_{app}(i)}(j_{i,k} \mid APX_{p_{app}(i)} \setminus \bigcup_{i'=1}^{i-1}\bigcup_{k'=1}^{d+1}\{j_{i',k'}\} \setminus \bigcup_{k'=1}^{k}\{j_{i,k'}\})$$

where:

- $j_{i,k}$ is the k^{th} item allocated at iteration i, where $j_{i,1}$ is the final item assigned at line 7 of this iteration, and the rest are ordered arbitrarily.
- $p_{opt}(i,k)$ is the player to whom the k^{th} item of iteration i is allocated in OPT.
- $p_{app}(i)$ is the player to whom all the items of iteration i are allocated in APX (all items of any iteration of Algorithm 3.4 are allocated to the same player).

Note that for simplicity, equations are written assuming exactly $d+1$ items are allocated at each iteration. The proof is correct also without this assumption.

Let $i \in [1..t]$ and let $Unallocated_i$ be the items of $Unallocated$ at line 4 of Algorithm 3.4 at iteration i. Then, since $M = \bigcup_{p=1}^{n}(OPT_p) = \bigcup_{p=1}^{n}(APX_p)$, we have:

$$Unallocated_i = \bigcup_{p=1}^{n}\left(OPT_p \setminus \bigcup_{i'=1}^{i-1}\bigcup_{k'=1}^{d+1}\{j_{i',k'}\}\right) = \bigcup_{p=1}^{n}\left(APX_p \setminus \bigcup_{i'=1}^{i-1}\bigcup_{k'=1}^{d+1}\{j_{i',k'}\}\right)$$

Therefore, for all $i \in [1..t]$, $k \in [1..d+1]$,

$$\bigcup_{p=1}^{n}\left(OPT_p \setminus \bigcup_{i'=1}^{i-1}\bigcup_{k'=1}^{d+1}\{j_{i',k'}\} \setminus \bigcup_{k'=1}^{k}\{j_{i,k'}\}\right) \subseteq Unallocated_i \ .$$

Then, by lines 4-10 and 13 of Algorithm 3.4, for all $i \in [1..t]$, $k \in [1..d+1]$, $v_{p_{app}(i)}(j_{i,1} \mid APX_{p_{app}(i)} \setminus \bigcup_{i'=1}^{i-1}\bigcup_{k'=1}^{d+1}\{j_{i',k'}\} \setminus \{j_{i,1}\}) \geq v_{p_{opt}(i,k)}(j_{i,k} \mid OPT_{p_{opt}(i,k)} \setminus \bigcup_{i'=1}^{i-1}\bigcup_{k'=1}^{d+1}\{j_{i',k'}\} \setminus \bigcup_{k'=1}^{k}\{j_{i,k'}\})$. Therefore

and by invoking (3) and (3) together with monotonicity,

$$
\begin{aligned}
&(d+1)\cdot \approx= \\
&(d+1)\cdot \sum_{i=1}^{t}\sum_{k}^{d+1} v_{p_{app}(i)}(j_{i,k} \mid APX_{p_{app}(i)} \setminus \bigcup_{i'=1}^{i-1}\bigcup_{k'=1}^{d+1}\{j_{i',k'}\} \setminus \bigcup_{k'=1}^{k}\{j_{i,k'}\}) \\
&\geq (d+1)\cdot \sum_{i=1}^{t} v_{p_{app}(i)}(j_{i,1} \mid APX_{p_{app}(i)} \setminus \bigcup_{i'=1}^{i-1}\bigcup_{k'=1}^{d+1}\{j_{i',k'}\} \setminus \{j_{i,1}\}) \\
&\geq \sum_{i=1}^{t}\sum_{k}^{d+1} v_{p_{opt}(i,k)}(j_{i,k} \mid OPT_{p_{opt}(i,k)} \setminus \bigcup_{i'=1}^{i-1}\bigcup_{k'=1}^{d+1}\{j_{i',k'}\} \setminus \bigcup_{k'=1}^{k}\{j_{i,k'}\}) \\
&= \mathsf{opt}
\end{aligned}
$$

It is easy to see the running time of Algorithm 3.4 is polynomial in $|M|, |P|$ and 2^c. This proves Theorem 3.6. □

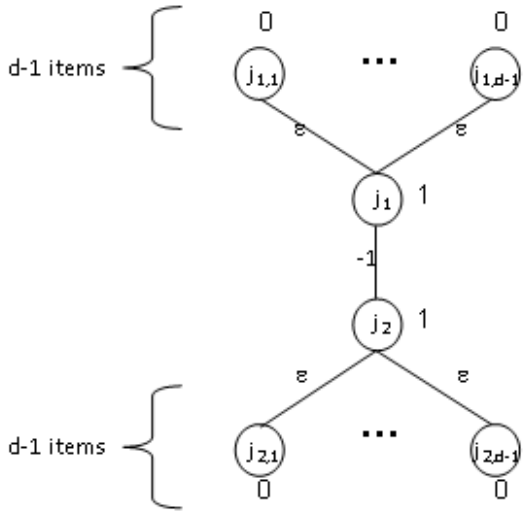
An example justifying "discarding" items We demonstrate the necessity of "discarding unused dependencies" as in line 13 of Algorithm 3.4. Note that another approach is to look also on already allocated items, as described in Section 3.7.

Example 3.22. *Let $|P| = 2$. Let the set functions be as follows: The valuation function of player 1 will be as in the hypergraph representation in Figure 4. The valuation function of player 2 will be $v_2(S) = |S|$.*

Intuitively, the idea is to cause Algorithm 3.4 to "ruin" a marginal value of an already allocated item, when trying to gain a maximal marginal value for another item. The middle edge with value -1 does so, without breaking monotonicity.

We analyze the approximation guarantee for this instance for Algorithm 3.4 with line 13 omitted (i.e., without "discarding unused dependencies"). On the first iteration the algorithm allocates either j_1 or j_2 to player 1 with their optimal dependencies. Assume without loss of generality it is j_1. The optimal dependencies of j_1 are $j_{1,1}, \ldots, j_{1,d-1}$ and the marginal value of j_1 with these dependencies is $1 + (d-1)\cdot\varepsilon$. On the second iteration, the algorithm allocates j_2 to player 1, together with its optimal dependencies (that has not been allocated yet) $j_{2,1}, \ldots, j_{2,d-1}$. The marginal value of j_2

Figure 4: Valuation function of player 1



with respect to these dependencies is also $1 + (d-1) \cdot \varepsilon$. *But now, the marginal value of* j_1 *with respect to the rest of the items allocated to player* 1 *(which had not been allocated yet when it was allocated) is only* $(d-1) \cdot \varepsilon$. *The algorithm terminates after the second iteration with social welfare* $1 + 2(d-1) \cdot \varepsilon$. *On the other hand, allocating all the items to player* 2 *results in a social welfare of* $2d$. *For small enough* ε *the approximation guarantee is arbitrarily close to* $1/2d$ *which is much worse than the approximation guarantee of* $1/(d+1)$ *of Algorithm 3.4.*

Note that Algorithm 3.4 (the unmodified version) does much better for this instance. It also chooses firstly j_1 *(without loss of generality) with its optimal dependencies, but then does not inspect anymore any of its dependencies, including* j_2. *Therefore, assuming* ε *is small, it allocates* $j_{2,1}, \ldots, j_{2,d-1}$ *to player* 2 *and gains for them a value of* $d-1$ *in addition to the marginal value it indeed gained for* j_1. *Thus, the approximated solution's total value Algorithm 3.4 gains for this input is* $1 + (d-1) \cdot \varepsilon + (d-1) = d + (d-1) \cdot \varepsilon$, *which expresses an approximation guarantee of slightly more than* $\frac{1}{2}$. *Note that allocating* j_2 *to any of the players will do no harm (and allocating it to player* 2 *will even slightly help); just reinspecting it does the harm.*

3.10.2 A tight example

We now show the analysis of Algorithm 3.4 is tight.

Example 3.23. *Let $|P| = 2$ and let $|M| = m' \cdot (d+1)$ for some $m' \in \mathbb{N}$. We set an arbitrary ordering on the items and define m' subsets of items $S_1, \ldots S_{m'}$; the first set will be the first $d+1$ items, the second one will be the next $d+1$ items and so on. Let v be the following set functions for any $S \subseteq M$:*

- $v_1(S) = \sum_{\substack{i \in [1..m'], \\ S_i \subseteq S}} (1+\varepsilon)$

  (meaning, $(1+\varepsilon)$ times the number of subsets S_i that are subsets of S).

- $v_2(S) = |S|$.

It is easy to see the marginal value of any item is maximized, when it is given to player 1 with all its dependencies. Moreover, since at this way only whole subsets are allocated, this is the situation at any iteration of Algorithm 3.4. Therefore, algorithm 3.4 allocates all the items to player 1 and gains total value of $m' \cdot (1+\varepsilon)$. In contrast, the optimal solution is to allocate all the items to player 2 and to gain total value of m. Thus, the approximation guarantee for this instance is close as we with to $m/m' = 1/(d+1)$, and the analysis of Theorem 3.6 is indeed tight.

3.11 An exact algorithm for dependency degree at most 1

In this section, we present a full reduction (Reduction 3.5) of the welfare maximization problem with dependency degree at most 1 to the maximum weighted matching problem.

The following observation is straightforward:

Observation 3.24.

- *Reduction 3.5 is polynomial time computable.*
- *Every feasible solution for $\mathcal{I}_M$ induces a feasible solution for $\mathcal{I}$ with the same value, that may be computed in polynomial time.*
- *Every feasible solution for $\mathcal{I}$, induces a feasible solution for $\mathcal{I}_M$ with at least the same value.*

Reduction 3.5

Input: An instance $\mathcal{I}(P, M, v)$ of the welfare maximization problem.
Output: An instance $\mathcal{I}_M(V, E, w)$ of the maximum weighted matching problem, with set of vertices V, set of undirected edges E and edge weights function w, such that each item $j \in M$ is represented by a corresponding vertex $v_j \in V$.
Reduction: For each item j, we have a vertex u_j. For each player p, we have the following. For each item j with $Dep_p(j) = \emptyset$, we have an auxiliary vertex u_j^p and an edge (u_j, u_j^p) with weight $v_p(\{j\})$, representing the possibility of allocating j to p. For each pair of items j, j', such that $j \leftrightarrow_p j'$, we have a single auxiliary vertex $u_{j,j'}^p$ and three edges: $(u_j, u_{j,j'}^p)$ with weight $v_p(\{j\})$, representing the possibility of allocating j to p without allocating j' to p; $(u_{j'}, u_{j,j'}^p)$ with weight $v_p(\{j'\})$, representing the possibility of allocating j' to p without allocating j to p; $(u_j, u_{j'})$ with weight $v_p(\{j, j'\})$, representing the possibility of allocating both j and j' to p. Note that both j and j' have no other dependencies, since the dependency degree is at most 1 and by Lemma 3.21. Note also that multiedges may be resolved, by choosing, without loss of generality, one edge with maximum weight for each pair of vertices.

Therefore, we may use Reduction 3.5 together with any exact polynomial time maximum weighted matching algorithm (see for example [29, 49, 50, 51]) to derive an exact polynomial time algorithm for the welfare maximization problem with dependency degree at most 1.

Remark 3.3. *Note that Reduction 3.5 does not work for $c > 1$. This is since we may have a player p with items j_1, j_2, j_3, such that $j_1 \leftrightarrow j_2 \leftrightarrow j_3$ (i.e., a "chain"). At this case, j_1, j_2 and j_3 may be allocated together to p, although this allocation does not induce a feasible matching.*

3.12 A greedy $\frac{1}{k(d+1)}$-approximation algorithm for dependency degree at most d for k-extendible system

In this section, we consider Algorithm 3.6, and prove it fulfills all the guarantees of Theorem 3.8.

Reduction 3.6 Extendible System Greedy - Dependency Degree$(f, \mathcal{I})$

1: Initialize: $S_0 \leftarrow \varnothing$, $i \leftarrow 0$
2: **while** S_i is not a base **do**
3: $\quad i \leftarrow i + 1$
4: $\quad$ Let $u_i \in M \setminus S_{i-1}$ and $D_{best}(u_i) \subseteq \mathcal{D}(u_i)$ be a pair of an element and a set maximizing $f(u_i \mid D_{best}(u_i) \cup S_{i-1})$ among all pairs obeying $S_{i-1} \cup D_{best}(u_i) + u_i \in \mathcal{I}$
5: $\quad S_i \leftarrow S_{i-1} \cup D_{best}(u_i) + u_i$
6: **end while**
7: Return S_i

For the analysis of Algorithm 3.6 we use the same notation introduced in Section 3.8, except that we set $d = \mathcal{D}_f$. Observe that both Observation 3.12 and Lemma 3.13 (and their proofs) apply also to Algorithm 3.6. The following lemma is a counterpart of Lemma 3.14.

Lemma 3.25. *For every iteration* $1 \leq i \leq \ell$, $f(H_{i-1}) - f(H_i) \leq [(k(d+1)-1] \cdot f(u_i \mid D_{best}(u_i) \cup S_{i-1})$, *where* u_i *and* $D^+_{best}(u_i)$ *are the greedy choices made by Algorithm 3.3 at iteration* i.

Proof. Order the elements of $H_{i-1} \setminus H_i$ in an arbitrary order $v_1, v_2, \ldots v_r$, and let $\bar{H}_j = H_{i-1} \setminus \{v_h \mid 1 \leq h \leq j\}$. By Definition 3.2 (dependency set), for every $1 \leq j \leq r$,

$$f(v_j \mid (\mathcal{D}(v_j) \cap \bar{H}_j) \cup S_{i-1}) = f(v_j \mid \bar{H}_j \cup S_{i-1}) \ .$$

Since $\bar{H}_j \cup S_{i-1} = \bar{H}_{j-1} \cup S_{i-1} - v_j$, we get:

$$\begin{aligned} \sum_{j=1}^{r} f(v_j \mid (\mathcal{D}(v_j) \cap \bar{H}_j) \cup S_{i-1}) &= \sum_{j=1}^{r} f(v_j \mid \bar{H}_j \cup S_{i-1}) \\ &= f(\bar{H}_0 \cup S_{i-1}) - f(\bar{H}_r \cup S_{i-1}) = f(H_{i-1}) - f(\bar{H}_r) \ , \end{aligned} \tag{3}$$

where the last equality holds since $\bar{H}_0 = H_{i-1} \supseteq S_{i-1}$ and $\bar{H}_r = H_{i-1} \cap H_i \supseteq S_{i-1}$. We upper bound $f(\bar{H}_r)$ by recalling that $\bar{H}_r \subseteq H_i$, which gives by monotonicity $f(\bar{H}_r) \leq f(H_i)$ and then, by (3), we have:

$$\sum_{j=1}^{r} f(v_j \mid (\mathcal{D}(v_j) \cap \bar{H}_j) \cup S_{i-1}) \geq f(H_{i-1}) - f(H_i) \ .$$

Note that the pair $(v_j, \left(\mathcal{D}(v_j) \cap \bar{H}_j\right) \setminus S_{i-1})$ is a candidate pair that Algorithm 3.6 can choose at Line 4 for every element $v_j \in H_{i-1} \setminus H_i$. This implies the lemma, unless $r = k(d+1)$ (recall that $r \leq k(d+1)$ by Lemma 3.13).

Thus, we may assume from now on that $r = k(d+1)$, which implies by Lemma 3.13 that $|S_i \setminus H_{i-1}| = d+1$. In other words, the algorithm adds u_i and all of $\mathcal{D}(u_i)$ in the i^{th} iteration. This means that the marginal contribution of u_i is maximized when all of $\mathcal{D}(u_i)$ is in the set, and thus, u_i contributes to the hybrid solution the same value it contributes to the final solution. Formally,

$|S_i \setminus H_{i-1}|$ implies $D_{best}(u_i) = \mathcal{D}(u_i)$ and $(\mathcal{D}(u_i) + u_i) \cap H_{i-1} = \varnothing$. Hence, since $\mathcal{D}(u_i) + u_i \subseteq H_i$:

$$\begin{aligned} f(H_i) =& f(\mathcal{D}(u_i) + u_i \mid \bar{H}_r) + f(\bar{H}_r) \\ &\geq f(u_i \mid \mathcal{D}(u_i) \cup \bar{H}_r) + f(\bar{H}_r) = f(u_i \mid D_{best}(u_i) \cup S_{i-1}) + f(\bar{H}_r) \ , \end{aligned}$$

where the inequality follows by monotonicity and the second equality by Definition 3.2 (dependency set) together with $D_{best}(u_i) = \mathcal{D}(u_i)$. Combining with (3), we get:

$$\sum_{j=1}^{k(d+1)} f(v_j \mid (\mathcal{D}(v_j) \cap \bar{H}_j) \cup S_{i-1}) - f(u_i \mid D_{best}(u_i) \cup S_{i-1}) \geq f(H_{i-1}) - f(H_i) \ ,$$

which implies the lemma. □

Corollary 3.26. *Algorithm 3.6 is a $(1/(k(d+1)))$-approximation algorithm.*

Proof. We have

$$\begin{aligned} [k(d+1)-1] \cdot [f(S_\ell) - f(S_0)] &= [k(d+1)-1] \cdot \sum_{i=1}^{\ell} f(D_{best}(u_i) + u_i \mid S_{i-1}) \\ &\geq [k(d+1)-1] \cdot \sum_{i=1}^{\ell} f(u_i \mid D_{best}(u_i) \cup S_{i-1}) \\ &\geq \sum_{i=1}^{\ell} [f(H_{i-1}) - f(H_i)] = f(H_0) - f(H_\ell) \ , \end{aligned} \tag{4}$$

where the first inequality follows by monotonicity and the second by adding up Lemma 3.25 over $1 \leq i \leq \ell$. Note that $H_\ell = S_\ell$ because S_ℓ is a base, and therefore, every independent set containing S_ℓ must be S_ℓ itself. Recall also that $f(H_0) = f(OPT)$ and $f(S_0) \geq 0$. Plugging these observations into (4) gives:

$$[k(d+1)-1] \cdot f(S_\ell) \geq f(OPT) - f(S_\ell) \Rightarrow f(S_\ell) \geq \frac{f(OPT)}{k(d+1)} \ . \qquad \square$$

3.12.1 A tight example

In this section we present an example showing that our analysis of Algorithm 3.6 is tight even when the independence system $(M, \mathcal{I})$ belongs to k-intersection (recall that any independence system in k-intersection is also k-extendible, but not vice versa).

Proposition 3.27. *For every $k \geq 1$, $d \geq 0$ and $\varepsilon > 0$, there exists a k-intersection independence system $(M, \mathcal{I})$ and a function $f : 2^M \to \mathbb{R}^+$ with $\mathcal{D}_f = d$ for which Algorithm 3.6 produces a $(1+\varepsilon)/(k(d+1))$ approximation.*

The rest of this section is devoted for constructing the independence system guaranteed by Proposition 3.27. Let $\mathcal{T}$ be the collection of all sets $T \subseteq \{1, 2, \ldots, k\} \times \{0, 1, \ldots, k(d+1) - 1\}$ obeying the following properties:

- For every $1 \leq i \leq k+1$, there exists exactly one x such that T contains the pair (i, x).
- At least one pair (i, x) in T has $x \leq d$.

Let M be the ground set $\{u_T \mid T \in \mathcal{T}\} \cup \{v_x\}_{x=0}^{k(d+1)-1}$. We define k matroids on this ground set as follows. For every $1 \leq i \leq k$, $\mathcal{M}_i = (M, \mathcal{I}_i)$, where a set $S \subseteq M$ belongs to $\mathcal{I}_i$ if and only if for every $0 \leq x < k(d+1)$, $|S \cap \{v_x\}| + |\{u_T \in S \mid (i, x) \in T\}| \leq 1$. One can easily verify that $\mathcal{M}_i$ is a partition matroid. The independence system we construct is the intersection of these matroids, *i.e.*, it is $(M, \mathcal{I})$, where $\mathcal{I} = \bigcap_{i=1}^{k} \mathcal{I}_i$. Next, we define the objective function $f : 2^M \to \mathbb{R}^+$, as follows. We first define the following function f'.

$$f'(S) = |\{u_T \in S \mid T \in \mathcal{T}\}| \ .$$

Let $\hat{T} = \{(i, 0)\}_{i=1}^{k}$ (note that $\hat{T} \in \mathcal{T}$). Then,

$$f(S) = \begin{cases} f'(S) + \varepsilon & \text{if } u_{\hat{T}} \in S \text{ and } \{v_i\}_{i=1}^{d} \subseteq S \ , \\ f'(S) & \text{otherwise} \ . \end{cases}$$

Since $f'(S)$ is a linear function, $\mathcal{D}_f = d$.

Claim 3.28. *Given the above constructed independence system $(M, \mathcal{I})$ and objective function f, Algorithm 3.6 outputs a solution of value $1 + \varepsilon$.*

Proof. At the first iteration, it is clear that Algorithm 3.6 picks exactly the elements of $\{v_i\}_{i=1}^{d} + u_{\hat{T}}$, since $\{v_i\}_{i=1}^{d}$ is the dependency set of $u_{\hat{T}}$, and the marginal contribution of any other element is at most 1, given any subset of M.

To complete the proof, we show that Algorithm 3.6 cannot increase the value of its solution at the next iterations. Consider an arbitrary element $u \in M \setminus (\{v_i\}_{i=1}^{d} + u_{\hat{T}})$. If $u = v_x$ for some $0 \leq x < k(d+1)$, then the addition of v_x does not affect the value of f. On the other hand, if $u = u_T$ for some $T \in \mathcal{T}$, then T must contain a pair (i, x) such that $0 \leq x \leq d$. There are two cases:

- If $x \neq 0$, then u_T cannot coexist in an independent set of $\mathcal{M}_i$ with v_x.
- If $x = 0$, then u_T cannot coexist in an independent set of $\mathcal{M}_i$ with $u_{\hat{T}}$ because both correspond to sets containing the pair $(i, 0)$. □

To prove Proposition 3.27, we still need to show that $(M, \mathcal{I})$ contains an independent set of a high value. Consider the set $S^* = \{u_{T^*(j)}\}_{j=1}^{k(d+1)}$, where $T^*(j) = \{(i, x) \mid 1 \leq i \leq k \text{ and } x = (i(d+1) - j) \bmod k(d+1)\}$.

Claim 3.29. *$S^* \subseteq M$, hence, $f(S^*) = |S^*| = k(d+1)$, because $S^* \cap \{v_x\}_{x=0}^{k(d+1)-1} = \varnothing$.*

Proof. We need to show that for every $1 \leq j \leq k(d+1)$, $u_{T^*(j)} \in M$. Let $i = \lceil j/(d+1) \rceil$. Clearly $1 \leq i \leq k$ and $T^*(j)$ contains the pair (i, x) for:

$$x = (i(d+1) - j) \bmod k(d+1) = (\lceil j/(d+1) \rceil \cdot (d+1) - j) \bmod k(d+1) \ .$$

To prove the claim, we need to show that $0 \leq x \leq d$. This follows since $\lceil j/(d+1) \rceil \cdot (d+1) - j \geq (j/(d+1)) \cdot (d+1) - j = 0$ and $\lceil j/(d+1) \rceil \cdot (d+1) - j < [j/(d+1) + 1] \cdot (d+1) - j = d + 1$. □

Claim 3.30. *For every two values $1 \leq j_1 < j_2 \leq k(d+1)$, $T^*(j_1) \cap T^*(j_2) = \varnothing$. Hence $S^* \in \mathcal{I}$.*

Proof. Assume towards contradiction that $(i, x) \in T^*(j_1) \cap T^*(j_2)$. Then, modulo $k(d+1)$, the following equivalence must hold:

$$(i(d+1) - j_1) \equiv (i(d+1) - j_2) \Rightarrow j_1 \equiv j_2 \ ,$$

which is a contradiction since $j_1 \neq j_2$ and they are both in the range $[1, k(d+1)]$. □

4 Building a Good Team: Secretary Problems and the Supermodular Degree

This section is based on a paper with Moran Feldman [42].

In the (classical) Secretary Problem, one has to hire a worker from a pool of n candidates. The candidates arrive to an interview at a uniformly random order, and the algorithm must decide immediately and irrevocably, after interviewing a candidate, whether to hire him or continue interviewing. The objective is to hire the best candidate. It is well-known that the best candidate can be hired with a probability of $1/e$, and that this is asymptotically optimal [28].

Recently, there has been an increased interest in variants of the secretary problem where more than a single candidate can be selected, subject to some constraint (e.g., a matroid constraint). Such variants have important applications in mechanism design (see, e.g., [3, 4, 5, 68] and the references therein). When more than one candidate can be selected, there is a meaning to the values of *subsets* of candidates. If one allows these values to be determined by an arbitrary non-negative monotone set function, then only exponential competitive ratios (in the number of candidates) can be achieved, even subject to a simple cardinality constraint.[11]

In light of the above hardness, previous works have concentrated on restricted families of objective functions, such as linear and submodular functions (see, e.g., [6, 7, 17, 24, 46, 56, 71] and a more thorough discussion in Section 4.2.2). However, for many applications, the desired set function might admit complements, i.e., a group of candidates might exhibit synergy and contribute more as a group than the sum of the candidates' personal contributions (see also Woolley et al. [88] for a research about collective intelligence). Such complements cannot be modeled by submodular (or linear) objectives. Dealing with complements in general results in unacceptable guarantees, as discussed above. However, what if one has a function which is submodular, except for a pair of candidates which are better to hire together? Can we guarantee anything for this case?

We give a strong affirmative answer to this question. Specifically, we give algorithms for sec-

[11] Intuitively, the bad example consists of a cardinality constraint allowing us to select only k candidates, and an objective function assigning a strictly positive value only to sets containing k specific candidates or more than k candidates. In this case the algorithm has no room for mistakes, which leads to a very poor performance.

retary problems with *arbitrary* non-negative monotone objective functions, whose guarantees are proportional to the distance of the objective function from being submodular, as measured by the *supermodular degree* (see Section 3). Back to the above example, the pair of synergistic candidates results in an objective function with a supermodular degree of 1, and for such an objective our algorithms provide a constant competitive ratio for the problem of hiring a team of a given size. Intuitively, the supermodular degree can be seen as measuring the number of candidates that any single candidate can have synergy with. Our algorithms handle both the case of a cardinality constraint (demonstrated above) and the more general case of a matroid constraint. For a cardinality constraint, we obtain a constant competitive ratio when the supermodular degree is constant. For a (general) matroid constraint, our competitive ratios depend logarithmically on the rank of the matroid.[12] To the best of our knowledge, these are the first algorithms for the secretary problem with an arbitrary non-negative monotone objective set function.

4.1 Techniques

Most algorithms for secretary problems start with a learning phase in which they reject all elements, and later, after accumulating some information about the input, they move to a phase in which they may accept elements. When the value of an element might positively depend on other d elements, there might be a set of $d+1$ elements such that every reasonable solution must contain this set. In this case, any reasonably good algorithm must terminate the learning phase, with a significant probability, before any element of this set arrives.

This means that the learning phases of our algorithms consists of only about $1/d$ of the input, and thus, they rarely see all the dependencies of an element in the learning phase. Thus, by the end of the learning phase the algorithm cannot calculate an optimal solution for the sub-problem represented by the part of the input seen so far. However, we show that it is possible to *estimate* the value of the optimal solution based on the learning phase, and this estimation is crucial for the performance of our algorithms.

[12] Note that, till a very short while ago, this was the case even for the state of the art algorithm for a *submodular* objective function (which corresponds to a supermodular degree of 0) [56]. An improved algorithm with a competitive ratio of $O(\log \log k)$ (where k is the rank of the matroid) has been recently given by [47].

4.2 Model and results

Consider the following scenario. A client enters a store, and wants to buy herself a new phone. However, the client's main motive to buy this phone is a novel accessory not supported by her old phone. Thus, the client asks the salesman to buy the phone together with the accessory. Unfortunately, the accessory is not available at that time, because supply does not meet the overwhelming demand. If the client insists on buying the phone only together with the accessory, the salesman can offer her to buy the phone now and get the accessory next week when a new supply shipment arrives.

On the other hand, consider a slightly different scenario. Here the client tells the salesman she wants the phone together with some accessory, but does not tell which accessory it is. The client then offers the following deal: the salesman will give her the phone now, and the client will pay when the unspecified accessory becomes available. Clearly no salesman can accept such an offer. Our model (below) assumes the more realistic first scenario. That is, in case of complementarity, the "bidder" is required to announce the future elements she needs in order to get the maximum value from the current "product".

The above example is a simplified real life scenario demonstrating the naturalness of using some future information. Our model is designed to handle not only complementarities of the form presented by this example (*i.e.*, single minded bidders), but also substantially more expressive complementarities, as can be captured by a monotone objective function with a given supermodular degree. Formally, an instance of the monotone matroid secretary problem consists of a ground set $\mathcal{N}$ of size n, a non-negative monotone set function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ and a matroid $M = (\mathcal{N}, \mathcal{I})$. The execution of an algorithm for this problem consists of n steps (also referred to as times). In each step the following occurs:

- One element of $\mathcal{N}$ is revealed (arrives), at a uniformly random order (without repetitions).
- The algorithm must decide whether to include the element in its output (irrevocably).

The objective of the algorithm is to select an independent subset maximizing f. When making decisions, the algorithm has access to the value of n, the supermodular degree of f and the following

oracles.

The first oracle is the independence oracle given above, which gives information about the constraint. The second oracle gives information about the objective function. This oracle is the counterpart of the value oracle defined above.

Definition 4.1. *Online marginal oracle of a set function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ is the following:*
Input: An element $u \in \mathcal{N}$ that has already been revealed *and a subset $S \subseteq \mathcal{N}$.*
Output: $f(u \mid S)$.

Note that our algorithms do not use the full power of the online marginal oracle. In particular, they only use it to calculate marginals with respect to already observed elements and their supermodular dependencies. This use is consistent with the above motivation of our model. The last oracle of our model returns the dependency sets of already revealed elements.

Definition 4.2. *Online supermodular oracle of a set function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ is the following:*
Input: An element $u \in \mathcal{N}$ that has already been revealed.
Output: $\mathcal{D}_f^+(u)$.

The last oracle can, in fact, be implemented using the online marginal oracle, albeit using an exponential time in n. However, this oracle is a natural online variant of the supermodular oracle emphasizing the relation between our model and previous work on the supermodular oracle.

On the necessity of receiving information about future elements. We briefly show that with no information about future elements one can have only exponentially decreasing approximation guarantees with respect to the supermodular degree of the objective set function. Let $A \subseteq \mathcal{N}$ and let $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ be a set function giving a value of 1 to every $S \supseteq A$ and 0 to every other subset of $\mathcal{N}$. Note that $\mathcal{D}_f^+ = |A| - 1$. Consider the maximization of f with respect to a uniform matroid constraint of rank $|A|$. Clearly, the only feasible solution with a positive value for this problem is A itself. Thus, an algorithm allowed to query only values of subsets of already revealed elements gets no information about A till all its elements are revealed. That is, up to this point such an algorithm has to guess correctly whether each of the revealed elements is in A. The probability of a successful guess is exponentially decreasing in $|A|$, and thus, also in $\mathcal{D}_f^+$.

Additional Notation. In the rest of this paper we use the notation $\mathcal{N}_u$ to denote the set of elements revealed up to the point in which a given element $u \in \mathcal{N}$ is revealed (*i.e.*, $\mathcal{N}_u$ contains u and every other element revealed before u).

4.2.1 Our results

In this section we formally state our results for the model introduced above. Our first result is for instances where the matroid M is of rank $k \leq \mathcal{D}_f^+ + 1$. This setting is interesting for two reasons: it is closely related to the classical secretary problem, and our algorithm for it is used as a building block in our algorithms for other settings.

Theorem 4.1. *There is an $O(k \log k) = O(\mathcal{D}_f^+ \log \mathcal{D}_f^+)$-competitive algorithm for the monotone matroid secretary problem when the rank of the matroid constraint M is $k \leq \mathcal{D}_f^+ + 1$.*

The time complexity of the above algorithm (and all our other algorithms) is $\texttt{Poly}(n, 2^{\mathcal{D}_f^+})$. The exponential dependence of the time complexity on $\mathcal{D}_f^+$ is unavoidable even for *offline* algorithms and a uniform matroid constraint (see Section 3.4.1. Our main result is given by the next theorem.

Theorem 4.2. *There is an $O({\mathcal{D}_f^+}^3 \log \mathcal{D}_f^+ + {\mathcal{D}_f^+}^2 \log k)$-competitive algorithm for the monotone matroid secretary problem.*

Interestingly, the above competitive ratio matches the, until recently, state of the art ratio of $O(\log k)$ by Gupta et al. [56] for the case where f is a submodular function, and extends it to any constant supermodular degree. For uniform matroids we have the following improved guarantee.

Theorem 4.3. *There is an $O({\mathcal{D}_f^+}^3 \log \mathcal{D}_f^+)$-competitive algorithm for the monotone matroid secretary problem when the matroid M is uniform.*

Note that the guarantee of Theorem 4.3 has no dependence on k, and thus, it yields a constant competitive ratio for a constant supermodular degree.

It is handy to assume that f is normalized (*i.e.*, $f(\varnothing) = 0$). Reduction 4.21 in Appendix 4.7 shows that this assumption is without loss of generality, and thus, we implicitly assume it in all our proofs.

Lower Bounds. Note that even the *offline* version of maximizing a function f with respect to matroid constraint is $\mathcal{NP}$-hard to approximate within a guarantee of $\Omega(\ln \mathcal{D}_f^+/\mathcal{D}_f^+)$ (see, *e.g.*, [60] and Section 3). Moreover, for a uniform matroid constraint, it is $\mathcal{SSE}$-hard to achieve any constant[13] approximation guarantee (even) in the offline setting [41].

4.2.2 Related results

Secretary problems Many variants of the secretary problem have been considered throughout the years, and we mention here only those most relevant to this work. Under a cardinality constraint of k, Babaioff et al. [4] and Kleinberg [68] achieve two incomparable competitive ratios of e and $1/[1 - O(1/\sqrt{k})]$, respectively, for linear objective functions. For submodular objective functions, the best algorithms have a competitive ratio of $8e^2 \approx 59$ for the general case [7] and a competitive ratio of $(e^2 + e)/(e - 1) \approx 5.88$ when the objective is also monotone [43].

The matroid secretary problem considers a linear objective and a general matroid constraint. This variant was introduced by Babaioff et al. [6], who described an $O(\log k)$-competitive algorithm for it (where k is the rank of the matroid) and conjectured the existence of an $O(1)$-competitive algorithm. Motivated by this conjecture, $O(1)$-competitive algorithms have been obtained for a wide variety of special classes of matroids including graphic matroids [6, 70], transversal matroids [6, 24, 70], co-graphic matroids [84], laminar matroids [61, 66, 75] and regular matroids [25]. However, progress on the general case has been much slower. An $O(\sqrt{\log k})$-competitive algorithm was described by Chakraborty and Lachish [17], and recently two $O(\log \log k)$-competitive algorithms were given by Lachish [71] and Feldman et al. [46].

The submodular variant of the matroid secretary problem was also considered. For general matroids [56] found an $O(\log k)$-competitive algorithm, and $O(1)$-competitive algorithms were described for special classes of matroids including partition matriods [7, 43, 56] and transversal and laminar matroids [75]. A recent work [47] shows that any algorithm for the linear variant can be translated, with a limited loss in the competitive ratio, into an algorithm for the submodular variant. This implies an $O(1)$-competitive algorithm for the submodular variant under any class

[13] That is, a guarantee that does not depend on $\mathcal{D}_f^+$.

of matroids admitting such an algorithm for linear objectives, and an $O(\log \log k)$-competitive algorithm for general matroids. A secretary problem with an even more general family of objective functions was considered by Bateni et al. [7] who proved a hardness result for subadditive objective functions. Finally, variants of the matroid secretary problem which use a different arrival process or a non-adversarial assignment of element values were also considered [52, 66, 84].

Complements in online settings A different online setting exhibiting complements can be found in the work of Emek et al. [31].

4.3 Small rank matroids (Theorem 4.1)

We begin this section by describing the main intuitive ideas behind the proof of Theorem 4.1. For simplicity, we assume in this intuitive description that M is a uniform matroid of rank $d+1$. The formal proof of the theorem, which does not make these simplifying assumptions, can be found in Section 4.3.1.

A natural generalization of the algorithm for the classical secretary problem is the following algorithm. First, during the learning phase, reject the first $O(n/d)$ elements. From the remaining elements, take the first one whose marginal contribution with respect to some of its future supermodular dependencies (*i.e.*, the elements that may increase its marginal contribution and the algorithm can still choose to take) is better than any such contribution inspected thus far.

It is not difficult to argue that the best marginal contribution seen by the above algorithm is always at least $f(OPT)/(d+1)$. However, to get a competitive ratio guarantee, we need to show that the algorithm manages to pick this best contribution with a significant probability. One approach to proving this claim is by generalizing the analysis of the classical secretary algorithm to this more general algorithm. Such a generalization requires lower bounding the probability that the following two events occur (at the same time).

- The element with best marginal contribution arrives after the learning phase.
- The second best marginal contribution, up to the point where we see the best contribution,

is seen during the learning phase.

Unfortunately, it is difficult to bound the above probability due to the following phenomenon. The earlier an element arrives, the more future supermodular dependencies it has, and thus, the higher its corresponding marginal contribution. Hence, elements in the learning phase tend to have larger marginal contributions in comparison to elements appearing after the learning phase.

To overcome this issue, we modify the algorithm. Specifically, instead of comparing the marginal contribution of the current element to the marginal contributions seen thus far, we compare it to the marginal contributions that the elements seen thus far could have if they would have arrived at this time (instead of the time in which they have really arrived). A similar idea has been previously used by a work on the case of a submodular objective function [43].

Additionally, to get the exact approximation ratio guaranteed by Theorem 4.1, the algorithm has to use a random threshold from a logarithmic scale. This allows the analysis to assume that (with a significant probability) the learning phase takes about half of the time up to the point when the best marginal is observed by the algorithm.

4.3.1 Formal Proof of Theorem 4.1

We need some additional notation. The max-marginal of an element u at time i is the largest marginal value that u can contribute to a subset of the elements that arrive after time i (while keeping the subset independent). More formally, let $\mathcal{N}_i$ be the (random) set of the first i elements that arrived, then the max-marginal of an element u at time i is:

$$\mathsf{m}_{\max}(u,i) = \max_{\substack{S \subseteq \mathcal{N} \setminus \mathcal{N}_i \\ S+u \in \mathcal{I}}} f(u \mid S) \ .$$

We also use $\mathsf{S}_{\max}(u,i)$ to denote an arbitrary set for which the maximum is obtained. Note that one can calculate both $\mathsf{m}_{\max}(u,i)$ and $\mathsf{S}_{\max}(u,i)$ in $O(2^d)$ time. We begin the proof with the following simple claim.

Claim 4.4. *We may assume that n is divisible by any quantity h whose value is polynomial in k.*

Proof. Let n' be the least multiple of h which is at least as large as n. Note that n' is polynomial in n (since $k \leq n$). Let $\mathcal{N}'$ be a ground set containing the elements of $\mathcal{N}$ and a set D of $n' - n$ dummy elements. We extend f and M to $\mathcal{N}'$ as follows:

- The function $f' : 2^{\mathcal{N}'} \to \mathbb{R}^+$ is defined as: $f'(S) = f(S \setminus D)$ for every set $S \subseteq \mathcal{N}'$. Note that f' is non-negative, monotone and has a supermodular degree of d. Additionally, $\mathcal{D}^+_{f'}(u) = \varnothing$ for the dummy elements of D, and $\mathcal{D}^+_{f'}(u) = \mathcal{D}^+_f(u)$ for every other element.
- The matroid $M' = (\mathcal{N}', \mathcal{I}')$ is defined by the following rule. A set $S \subseteq \mathcal{N}'$ is in $\mathcal{I}'$ if and only if $S \setminus D \in \mathcal{I}$ and $|S| \leq k$. Note that this rule defines a matroid of rank k which is uniform whenever M is.

One can observe that the problems (f, M) and (f', M') are equivalent in the sense that: any solution for (f, M) is also a solution for (f', M') of the same value, and removing the dummy elements of any solution for (f', M') results in a solution for (f, M) of the same value. Moreover, given access to the oracles corresponding to (f, M), one can efficiently implement the oracles for (f', M'). Thus, given algorithm ALG that is r-competitive for ground sets obeying the requirements of the reduction, one can construct an r-competitive algorithm for general ground sets as follows:

1. Apply ALG to the instance (f', M').
2. Accept every element of $\mathcal{N} = \mathcal{N}' \setminus D$ that ALG accepts.

□

In the rest of this section we make two assumptions. First, we assume that n is divisible by $10k$, which is justified by Claim 4.4. Second, we assume $k \geq 2$ (if $k = 1$, then the classical secretary algorithm can be used to get an $O(1)$-competitive algorithm). Our objective is to show that Algorithm 4.1 obeys the requirements of Theorem 4.1 given these assumptions.

For the purpose of analyzing Algorithm 4.1, it is helpful to think about the input as created backwards by the following process. The set $\mathcal{N}_n$ is simply the entire ground set $\mathcal{N}$. Then, the last element of the input u_n is selected uniformly at random from $\mathcal{N}_n$, and the set $\mathcal{N}_{n-1}$ becomes

Reduction 4.1 Small Rank Matroid

1: Select an arbitrary order $\prec$ over the elements of the ground set $\mathcal{N}$.
2: Let p be a uniformly random integer from the set $\{0, 1, \ldots, \lceil \log_2 k \rceil\}$.
3: Reject the first $t = 2^p \cdot \frac{n}{2k}$ elements.
4: **for** $i = t + 1$ to n **do**
5: Let u_i be the element arriving at time i.
6: **if** for every element $u \in \mathcal{N}_{i-1}$ either $\mathsf{m}_{\max}(u_i, i) > \mathsf{m}_{\max}(u, i)$ or $(\mathsf{m}_{\max}(u_i, i) = \mathsf{m}_{\max}(u, i)$ and $u_i \succ u)$ **then**
7: Terminate the "for" loop and accept the elements of $\mathsf{S}_{\max}(u_i, i) + u_i$ when they arrive.
8: **end if**
9: **end for**

$\mathcal{N}_n - u_n$. On the next step, the $(n-1)$-th element u_{n-1} of the input is selected uniformly at random from $\mathcal{N}_{n-1}$ and we set $\mathcal{N}_{n-2} = \mathcal{N}_{n-1} - u_{n-1}$. The process then continues in the same way, *i.e.*, when it is time to determine the i-th element of the input, this element is selected uniformly at random from $\mathcal{N}_i$, and we set $\mathcal{N}_{i-1} = \mathcal{N}_i - u_i$.

We say that an element u is the *top* element of a set $\mathcal{N}_i$ if for every other element $u' \in \mathcal{N}_i \setminus u$ either $\mathsf{m}_{\max}(u', i) < \mathsf{m}_{\max}(u, i)$ or $\mathsf{m}_{\max}(u', i) = \mathsf{m}_{\max}(u, i)$ and $u' \prec u$. Note that Line 6 of Algorithm 4.1 in fact checks whether u_i is the top element of $\mathcal{N}_i$. Additionally, we say that an input is *well-behaved* with respect to the value t and order $\prec$ chosen by Algorithm 4.1 if it has the following properties:

(A1) There exists a time $i > n/k$ such that for some element $u \in \mathcal{N}_i$, $\mathsf{m}_{\max}(u, i) \geq f(OPT)/k$, where OPT is an independent set maximizing f. We denote the first such time by ℓ_1.

(A2) There exists exactly a single time $t < i \leq \ell_1$ such that u_i is the top element of $\mathcal{N}_i$.[14] We denote this time by ℓ_2.

The analysis of Algorithm 4.1 consists of two parts. First we show that it produces a good output for well-behaved inputs, and then we show that the input is well-behaved with a significant probability.

Lemma 4.5. *Algorithm 4.1 outputs a solution of value at least $f(OPT)/k$ when its input is well-behaved with respect to t and $\prec$.*

[14] Note that in some cases we might have $\ell_1 \leq t$. In these cases the input is not well-behaved with respect to t and $\prec$.

Proof. The definition of the algorithm and Property A2 guarantees that the algorithm outputs $\mathsf{S}_{\max}(u_{\ell_2}, \ell_2) + u_{\ell_2}$. The value of this solution is:

$$\begin{aligned} f(\mathsf{S}_{\max}(u_{\ell_2}, \ell_2) + u_{\ell_2}) &\geq f(u_{\ell_2} \mid \mathsf{S}_{\max}(u_{\ell_2}, \ell_2)) \\ &= \mathsf{m}_{\max}(u_{\ell_2}, \ell_2) \ . \end{aligned}$$

Hence, we are only left to lower bound $\mathsf{m}_{\max}(u_{\ell_2}, \ell_2)$. Observe that by Property A1, there exists an element $u'_{\ell_1} \in \mathcal{N}_{\ell_1}$ such that $\mathsf{m}_{\max}(u'_{\ell_1}, \ell_1) \geq f(OPT)/k$. Let us prove by a backward induction that this is true for every $\ell_2 \leq i \leq \ell_1$, *i.e.*, that for every such time there exists an element $u'_i \in \mathcal{N}_i$ such that $\mathsf{m}_{\max}(u'_i, i) \geq f(OPT)/k$.

Assume the claim holds for a given $\ell_2 < i \leq \ell_1$, and let us prove it for $i-1$. Observe that we can assume without loss of generality that u'_i is the top element of $\mathcal{N}_i$. Thus, by Property A2 $u'_i \neq u_i$, which implies: $u'_i \in \mathcal{N}_{i-1}$. By definition $\mathsf{m}_{\max}(u, i)$ is a non-increasing function of i, hence, $\mathsf{m}_{\max}(u'_i, i-1) \geq \mathsf{m}_{\max}(u'_i, i) \geq f(OPT)/k$, which complete the induction step.

The claim we proved by induction implies that: $\mathsf{m}_{\max}(u'_{\ell_2}, \ell_2) \geq f(OPT)/k$. The lemma now follows by observing that Property A2 guarantees that u_{ℓ_2} is the top element of $\mathcal{N}_{\ell_2}$, and thus, $\mathsf{m}_{\max}(u_{\ell_2}, \ell_2) \geq \mathsf{m}_{\max}(u'_{\ell_2}, \ell_2)$. □

Lemma 4.6. *Property A1 holds with a probability of at least* 0.2.

Proof. Given an element $u \in \mathcal{N}$, let i_u denote the time when it arrives. Then,

$$\begin{aligned} \sum_{u \in OPT} \mathsf{m}_{\max}(u, i_u) &\geq \sum_{u \in OPT} f(u \mid OPT \setminus \mathcal{N}_u) \\ &= f(OPT) \ , \end{aligned}$$

where the inequality follows from the definition of $\mathsf{m}_{\max}$. Since $|OPT| \leq k$, we get by averaging that for some element $u \in OPT$ there must be $\mathsf{m}_{\max}(u, i_u) \geq f(OPT)/k$. Hence, Property A1 is guaranteed to hold when all the elements of OPT appear after time $\hat{t} = n/k$. The last event occurs

with a probability of at least:

$$\begin{aligned}\frac{\left(\hat{t}! \cdot \binom{n-k}{\hat{t}}\right) \cdot (n-\hat{t})!}{n!} &= \frac{(n-k)! \cdot (n-\hat{t})!}{n! \cdot (n-k-\hat{t})!} \\ &= \prod_{i=0}^{\hat{t}-1} \frac{n-k-i}{n-i} \geq \left(\frac{n-k-\hat{t}}{n-\hat{t}}\right)^{\hat{t}} \\ &= \left(1-\frac{k}{n-\hat{t}}\right)^{\hat{t}} \geq \left(1-\frac{k}{0.9n}\right)^{n/k} \\ &\geq e^{-1/0.9} \cdot \left(1-\frac{k}{0.9^2 \cdot n}\right) \\ &\geq e^{-10/9} \cdot \left(1-\frac{1}{8.1}\right) \geq 0.2 \ .\end{aligned}$$

□

Lemma 4.7. *Given that Property A1 holds, with a probability of at least* $(\log_2 k+2)^{-1}/4$ *Property A2 holds as well.*

Proof. First, let us consider the event E_1 that there exists a time $\ell_1/2 < \ell_2' \leq \ell_1$ such that $u_{\ell_2'}$ is the top element of $\mathcal{N}_{\ell_2'}$ and for every time $\ell_2' < i \leq \ell_1$, u_i is not the top element of $\mathcal{N}_i$. For this event not to occur, a non-top element u_i must be selected from $\mathcal{N}_i$ for every time $\ell_1/2 < i \leq \ell_1$, which happens with probability:

$$\prod_{i=\lfloor \ell_1/2+1 \rfloor}^{\ell_1} \frac{i-1}{i} = \frac{\lfloor \ell_1/2 \rfloor}{\ell_1} \leq \frac{1}{2} \ .$$

Hence, E_1 occurs with the complement probability, which is at least $1/2$. Next, given that E_1 occurred, we are interested in the event that $\ell_2'/2 \leq t < \ell_2'$, which we denote by E_2. It is important to notice that E_1 is independent of the choice of t by the algorithm, and thus, the distribution of t is unaffected by conditioning on E_1. Additionally, notice that:

$$2^0 \cdot \frac{n}{2k} = \frac{n}{2k} \leq \frac{\ell_1}{2} < \ell_2' \quad \text{and} \quad \frac{\ell_2'}{2} \leq \frac{n}{2} \leq 2^{\lceil \log_2 k \rceil} \cdot \frac{n}{2k} \ .$$

Hence, one of the possible values of t obeys the requirement $\ell_2'/2 \leq t < \ell_2'$. Since t takes at most

$\log_2 k + 2$ different values, and it takes them with equal probabilities, we get that E_2 occurs with a probability of at least $(\log_2 k + 2)^{-1}$ given E_1.

Given that E_1 and E_2 both occur, for Property A2 to hold with need the additional event that in the range (t, ℓ'_2) no element u_i is the top element of $\mathcal{N}_i$. Note that the order of the elements of $\mathcal{N}_{\ell'_2} - u_{\ell'_2}$ is independent of E_1 and E_2. Hence, the probability of this event is at least:

$$\prod_{i=t+1}^{\ell'_2-1} \frac{i-1}{i} = \frac{t}{\ell'_2 - 1} \geq \frac{\ell'_2/2}{\ell'_2 - 1} > 1/2 \ .$$

□

We are now ready to prove Theorem 4.1.

Proof of Theorem 4.1. Lemmata 4.6 and 4.7 imply that the input is well-behaved with respect to t and $\prec$ with probability at least $(\log_2 k + 2)^{-1}/20$. By Lemma 4.5, when the input is well-behaved with respect to t and $\prec$, Algorithm 4.1 outputs a solution of value at least $f(OPT)/k$. Hence, the competitive ratio of Algorithm 4.1 is at least:

$$20k(\log_2 k + 2) = O(k \log k) \ .$$

□

4.4 Estimation aided algorithms

We say that a value opt_α is an α-estimation of an optimum solution OPT if it obeys $f(OPT)/\alpha \leq \mathsf{opt}_\alpha \leq f(OPT)$. We say that an algorithm is *α-aided* if it assumes getting an α-estimation of the optimum as part of its input. In this section we describe an aided algorithm for the case of a general matroid constraint, and an improved aided algorithm for the special case of a uniform matroid constraint. In the next section we explain how to convert our aided algorithms into non-aided ones. Note that our aided algorithms work even under a model where the arrival order is determined by an adversary. However, the randomness of the input is required for converting them

into non-aided algorithms.

4.4.1 Estimation aided algorithm for a general matroid constraint

The properties of the first algorithm we describe are given by the following theorem.

Theorem 4.8. *For every $\alpha \geq 1$, there exists an α-aided $O(d^2 \log(\alpha k))$-competitive algorithm for the monotone matroid secretary problem.*

The algorithm we use to prove Theorem 4.8 is Algorithm 4.2. A few of the ideas used in Algorithm 4.2 and its analysis can be traced back to [6].

Reduction 4.2 α-Aided Algorithm for General Matroids

1: Let p be a uniformly random integer from the set $\{-\lceil \log_2 k \rceil - 3, -\lceil \log_2 k \rceil - 2, \ldots, \lceil \log_2 \alpha \rceil\}$.
2: Let $\tau \leftarrow 2^p \cdot \frac{\mathsf{opt}_\alpha}{2}$ and $S \leftarrow \varnothing$.
3: **for** every arriving element u **do**
4: **if** there exists a set $D^*(u) \subseteq \mathcal{D}^+(u) \setminus \mathcal{N}_u$ such that $f(u \mid D^*(u) \cup S) \geq \tau$ and $S \cup D^*(u) + u \in \mathcal{I}$ **then**
5: Add $D^*(u) + u$ to S.
6: **end if**
7: **end for**
8: **return** S.

We define a weight $w(u)$ for every element $u \in OPT$ as follows: $w(u) = f(u \mid OPT \setminus \mathcal{N}_u)$, and extend w to subsets of OPT in the natural way. Additionally, let $p_1 = -\lceil \log_2 k \rceil$ and $p_2 = \lceil \log_2 \alpha \rceil$. For every integer $p_1 \leq p \leq p_2$, we define a set (bucket) $OPT_p = \{u \in OPT \mid 2^p \cdot \frac{\mathsf{opt}_\alpha}{2} \leq w(u) \leq 2^p \cdot \mathsf{opt}_\alpha\}$. Intuitively, the following lemma shows that, taken together, the buckets contain significant value. The lemma holds since any element belonging to no bucket must be of a very low weight.

Lemma 4.9. $w\left(\bigcup_{p=p_1}^{p_2} OPT_p\right) \geq \frac{f(OPT)}{2}$.

Proof. Clearly $w(OPT) = f(OPT)$. Moreover, by definition, $2^{p_1} \cdot \mathsf{opt}_\alpha \leq f(OPT)/k$. Hence:

$$f(OPT) - w\left(\bigcup_{p=p_1}^{p_2} OPT_p\right) = \sum_{\substack{w \in OPT \\ w(u) < 2^{p_1} \cdot \mathsf{opt}_\alpha / 2}} w(u)$$
$$\leq k \cdot (2^{p_1} \cdot \mathsf{opt}_\alpha / 2) \leq \frac{f(OPT)}{2} \quad .$$

□

Our next objective is to show that if Algorithm 4.2 selects a value p, then its gain is proportional to $w(OPT_{p+3})$. Whenever S appears below it denotes the output of the algorithm.

Lemma 4.10. *If Algorithm 4.2 selects a value p and $|S| \geq |OPT_{p+3}|/[2(d+1)]$, then $f(S) \geq w(OPT_{p+3})/[32(d+1)^2]$.*

Intuitively, Lemma 4.10 holds since a large $|S|$ means that the algorithm adds elements to S in many iterations, and each iteration increases $f(S)$ by at least τ.

Proof of Lemma 4.10. For an element $u \in \mathcal{N}$, let S_u be the set S immediately before u is processed by Algorithm 4.2. Note that each time that Algorithm 4.2 adds elements to S, it adds up to $d+1$ elements and $f(S)$ increases by at least τ since, by monotonicity:

$$f(D^*(u) + u \mid S_u) \geq f(u \mid D^*(u) \cup S_u) \ .$$

Hence, we can lower bound $f(S)$ by:

$$\begin{aligned} f(S) \geq \left\lceil \frac{|S|}{d+1} \right\rceil \cdot \tau &\geq \frac{|OPT_{p+3}|}{2(d+1)^2} \cdot \left[\frac{1}{16} \cdot \max_{u \in OPT_{p+3}} w(u) \right] \\ &\geq \frac{w(OPT_{p+3})}{32(d+1)^2} \ . \end{aligned}$$

□

Lemma 4.11. *If Algorithm 4.2 selects a value p and $|S| < |OPT_{p+3}|/[2(d+1)]$, then $f(S) \geq w(OPT_{p+3})/8$.*

The main idea behind the proof of Lemma 4.11 is as follows. Since $|S|$ is small, many elements of $OPT_{p+3} \setminus S$ could be added to it, together with their dependencies, without violating independence. The reason these elements were not added must be that they did not pass the threshold, which can only happen when $f(S)$ is large enough.

Proof of Lemma 4.11. Observe that OPT_{p+3} is a subset of OPT, and thus, independent. Hence, by the matroid properties, there exists a set $O' \subseteq OPT_{p+3} \setminus S$ of size at least $|OPT_{p+3}| - |S|$ such that

$O' \cup S \in \mathcal{I}$. Every element $u \in OPT_{p+3} \setminus O'$ can belong to the dependence set of at most d other elements of OPT_{p+3}. Thus, the number of elements $u \in OPT_{p+3}$ having $\mathcal{D}^+(u) \cap OPT + u \not\subseteq O'$ is upper bounded by:

$$(d+1) \cdot |S| < \frac{|OPT_{p+3}|}{2} \ .$$

In other words, there exists a set $O'' \subseteq OPT_{p+3}$ of size at least $|OPT_{p+3}|/2$ such that $\mathcal{D}^+(u) \cap OPT + u \subseteq O'$ for every $u \in O''$. Observe that by monotonicity:

$$\begin{aligned} f(O') &\geq \sum_{u \in O''} f(u \mid O' \setminus \mathcal{N}_u) \\ &\geq \sum_{u \in O''} f(u \mid OPT \setminus \mathcal{N}_u) = w(O'') \\ &\geq \frac{|OPT_{p+3}|}{2} \cdot \min_{u \in OPT_{p+3}} w(u) \geq \frac{w(OPT_{p+3})}{4} \ , \end{aligned}$$

where the second inequality holds since O' already contains all the elements of $\mathcal{D}^+(u) \cap OPT$.

Every element $u \in O'$ must have been rejected upon arrival by Algorithm 4.2 due to the threshold. Moreover, for every such element u we have $([\mathcal{D}^+(u) \cap (O' \cup S)] \setminus \mathcal{N}_u + u) \cup S_u \subseteq O' \cup S \in \mathcal{I}$ (where S_u is, again, the set S immediately before u is processed by Algorithm 4.2). Hence:

$$\begin{aligned} f(u \mid (O' \setminus \mathcal{N}_u) \cup S) & \\ &\leq f(u \mid [\mathcal{D}^+(u) \cap (O' \cup S)] \setminus \mathcal{N}_u \cup S_u) \\ &< \tau \ , \end{aligned}$$

where the first inequality holds by the definition of $\mathcal{D}^+(u)$. Adding the last inequality over all

elements $u \in O'$ gives:

$$\begin{aligned}\frac{w(OPT_{p+3})}{4} &\leq f(O') \leq f(O' \cup S) \\ &= f(S) + \sum_{u \in O'} f(u \mid (O' \setminus \mathcal{N}_u) \cup S) \\ &< f(S) + |OPT_{p+3}| \cdot \tau \\ &\leq f(S) + |OPT_{p+3}| \cdot \frac{\min_{u \in OPT_{p+3}} w(u)}{8} \\ &\leq f(S) + \frac{w(OPT_{p+3})}{8} \ .\end{aligned}$$

The lemma now follows by rearranging the last inequality. □

Corollary 4.12. *If Algorithm 4.2 selects a value p, then $f(S) \geq w(OPT_{p+3})/[32(d+1)^2]$.*

We are now ready to prove Theorem 4.8.

Proof of Theorem 4.8. Recall that every value p is selected by Algorithm 4.2 with probability at least $(\log_2 \alpha + \log_2 k + 6)^{-1} = (\log_2(\alpha k) + 6)^{-1}$. Hence, by Corollary 4.12, the expected value of the output of Algorithm 4.2 is at least:

$$\begin{aligned}&\frac{1}{\log_2(\alpha k) + 6} \cdot \sum_{p=p_1}^{p_2} \frac{w(OPT_{p+3})}{32(d+1)^2} \\ &\qquad = \frac{w\left(\bigcup_{p=p_1}^{p_2} OPT_p\right)}{32(d+1)^2 \cdot [\log_2(\alpha k) + 6]} \\ &\qquad \geq \frac{f(OPT)}{64(d+1)^2 \cdot [\log_2(\alpha k) + 6]} \ ,\end{aligned}$$

where the last inequality is due to Lemma 4.9. □

4.5 Estimation aided algorithm for a uniform matroid constraint

In this section we prove the following theorem.

Theorem 4.13. *For every $\alpha \geq 1$, there exists an α-aided $O(d \log \alpha)$-competitive algorithm for the monotone matroid secretary problem when the matroid M is uniform.*

Before proving the existence of an α-aided algorithms for any $\alpha \geq 1$, let us begin with a 2-aided algorithm.

Proposition 4.14. *There is a 2-aided $O(d)$-competitive algorithm for the monotone matroid secretary problem when the matroid M is uniform.*

The algorithm we use to prove Proposition 4.14 is Algorithm 4.3.

Reduction 4.3 2-Aided Cardinality

Let $\tau \leftarrow \frac{\mathsf{opt}_2}{2k}$.
Let $S \leftarrow \varnothing$.
for every arriving element u **do**
  if there exists a set $D^*(u) \subseteq \mathcal{D}^+(u) \setminus \mathcal{N}_u$ such that $f(u \mid D^*(u) \cup S) \geq \tau$ and $|S| + |D^*(u)| + 1 \leq k$ **then**
    Add $D^*(u) + u$ to S.
  end if
end for
return S.

Let S_u be the set S before the element u is processed. When S appears below without a subscript it denotes the output of the algorithm.

Lemma 4.15. *If $|S| \leq \max\{0, k - d - 1\}$, then $f(S) \geq f(OPT)/2$.*

Proof. Assume, towards a contradiction, that $|S| \leq \max\{0, k - d - 1\}$ and still $f(S) < f(OPT)/2$. Since $|S| \leq \max\{0, k - d - 1\}$, for every element $u \in OPT \setminus S$ Algorithm 4.3 could add the set $[\mathcal{D}^+(u) \cap (OPT \cup S)] \setminus \mathcal{N}_u + u$ to S. From the fact that the algorithm did not add this set (or any other set containing u) to S, we learn that:

$$\begin{aligned} &f(u \mid (OPT \setminus \mathcal{N}_u) \cup S) \\ &\quad \leq f(u \mid [\mathcal{D}^+(u) \cap (OPT \cup S)] \setminus \mathcal{N}_u \cup S_u) < \tau \ , \end{aligned}$$

where the first inequality holds by the definition of $\mathcal{D}^+(u)$. Adding the last inequality over all

elements $u \in OPT \setminus S$ gives:

$$\begin{aligned} f(OPT) &\leq f(OPT \cup S) \\ &= f(S) + \sum_{u \in OPT} f(u \mid (OPT \setminus \mathcal{N}_u) \cup S) \\ &< f(S) + k\tau \ . \end{aligned}$$

Plugging the assumption that $f(S) < f(OPT)/2$ and the definition of τ into the last inequality gives an immediate contradiction. □

Lemma 4.16. *If* $|S| \geq \max\{1, k-d\}$, *then* $f(S) \geq f(OPT)/[8(d+1)]$.

Proof. Note that each time that Algorithm 4.3 adds elements to S, it adds up to $d+1$ elements and $f(S)$ increases by at least τ since, by monotonicity:

$$f(D^*(u) + u \mid S_u) \geq f(u \mid D^*(u) \cup S_u) \ .$$

Hence, we can lower bound $f(S)$ by:

$$\begin{aligned} f(S) &\geq \left\lceil \frac{|S|}{d+1} \right\rceil \cdot \tau \geq \left\lceil \frac{\max\{1, k-d\}}{d+1} \right\rceil \cdot \frac{\mathsf{opt}_2}{2k} \\ &\geq \frac{k}{2(d+1)} \cdot \frac{f(OPT)}{4k} = \frac{f(OPT)}{8(d+1)} \ . \end{aligned}$$

□

Proposition 4.14 follows immediately from the last two lemmata. Theorem 4.13 generalizes Proposition 4.14 to general α-aided algorithms. The algorithm we use to prove Theorem 4.13 is Algorithm 4.4.

Reduction 4.4 α-Aided Cardinality

Let p be a uniformly random integer from the set $\{0, 1, \ldots, \lceil \log_2 \alpha \rceil\}$.
Apply Algorithm 4.3 with $\mathsf{opt}_2 = 2^p \cdot \mathsf{opt}_\alpha$.

Proof of Theorem 4.13. The largest value Algorithm 4.4 can assign to opt_2 is:

$$2^{\lceil \log \alpha \rceil} \cdot \mathsf{opt}_\alpha \geq \alpha \cdot (f(OPT)/\alpha) = f(OPT) \ .$$

On the other hand, the smallest value Algorithm 4.4 can assign to opt_2 is: $2^0 \cdot \mathsf{opt}_\alpha \leq f(OPT)$. Hence, for some p Algorithm 4.4 is guaranteed to produce a value opt_2 obeying $f(OPT)/2 \leq \mathsf{opt}_2 \leq f(OPT)$, in which case Algorithm 4.3 is $O(d)$-competitive by Proposition 4.14. Since every value of p occurs with a probability of at least $1/(\log_2 \alpha + 2)$, we get that the competitive ratio of Algorithm 4.4 is at most $O(d \log \alpha)$. □

4.6 Estimating the optimum: from aided to non-aided algorithms

In this section, we show how to convert aided algorithms into non-aided ones. Together with our aided algorithms, the following theorem implies the results stated in Theorems 4.2 and 4.3.

Theorem 4.17. *If there exists a $(80(d+2)^2)$-aided β-competitive algorithm ALG for the monotone matroid secretary problem with supermodular degree d, under a class $\mathcal{C}$ of matroid constraints closed under restriction, then there also exists a non-aided $O(d^3 \log d + \beta)$-competitive algorithm for the same problem.*

Recall that the truncation of a matroid $M = (\mathcal{N}, \mathcal{I})$ to rank k' is a matroid $M' = (\mathcal{N}, \mathcal{I}')$ where a set $S \subseteq \mathcal{N}$ is independent in M' if and only if $S \in \mathcal{I}$ and $|S| \leq k'$. The algorithm we use to prove Theorem 4.17 is Algorithm 4.5.

Algorithm 4.5 consists of two parts, each executed with probability 1/2. In order to prove Theorem 4.17 we show that for any instance of the monotone matroid secretary problem, one of the following cases is true:

- The algorithm guaranteed by Theorem 4.1 produces an $O(d^3 \log d)$-competitive solution for the non-truncated problem.
- The second part of Algorithm 4.5 achieves a competitive ratio of $O(\beta)$.

To determine which of the above cases is true for every given instance we need some notation.

Reduction 4.5 Multiple Elements Estimation

With probability of 1/2:
  Apply the algorithm guaranteed by Theorem 4.1 to the problem after truncating the matroid to rank $\min\{k, d+1\}$ (where k is the rank of the original matroid).
Otherwise:
  Choose X according to the binomial distribution $B(n, (d+2)^{-1})$, and let T the set of the first X elements revealed.
  Let $A \leftarrow \varnothing$ and $W \leftarrow 0$.
  while there exist $u \in T \setminus A$ and $\mathcal{D}_u \subseteq \mathcal{D}^+(u)$ s.t. $A \cup \mathcal{D}_u + u \in \mathcal{I}$ **do**
    Find such a pair maximizing $f(u \mid A \cup \mathcal{D}_u)$.
    Increase $W \leftarrow W + f(u \mid A \cup \mathcal{D}_u)$ and update $A \leftarrow A \cup \mathcal{D}_u + u$.
  end while Apply ALG to the remaining elements with $\mathsf{opt}_{80(d+2)^2} = W/10$.

Let $u^* \in \mathcal{N}$ be an element maximizing

$$\max_{\substack{S \subseteq \mathcal{D}^+(u^*) \\ S+u^* \in \mathcal{I}}} f(u^* \mid S) \ ,$$

and let m^* and S^* denote the value of this maximum and an arbitrary corresponding set S, respectively. It can be shown quite easily that the first case above holds when $m^* \geq f(OPT)/(256(d+1)^2)$.

Lemma 4.18. *If $m^* \geq f(OPT)/[256(d+1)^2]$, then Algorithm 4.5 is $O(d^3 \log d)$-competitive.*

Proof. Note that $S^* + u^*$ is an independent set in the matroid even after it is truncated to rank $\min\{k, d+1\}$. Hence, when Algorithm 4.5 applies the algorithm guaranteed by Theorem 4.1 (which happens with probability 1/2), the produced set has an expected value of at least:

$$\begin{aligned} \frac{f(S^* + u^*)}{O(d \log d)} \geq \frac{f(u^* \mid S^*)}{O(d \log d)} &= \frac{m^*}{O(d \log d)} \\ &\geq \frac{f(OPT)/[256(d+1)^2]}{O(d \log d)} \ . \end{aligned}$$

□

We now sketch the more interesting part of the analysis, which is to show that the second above case holds when $m^* \leq f(OPT)/(256(d+1)^2)$ (a full proof can be found in Section 4.8). The main thing we need to show is that $\mathsf{opt}_{80(d+2)^2}$ is, with constant probability, an $80(d+2)^2$-estimation for $f(OPT)$. For that purpose, we relate the expected value of the estimate $\mathsf{opt}_{80(d+2)^2}$ to $f(OPT)$,

and then bound the variance of this estimate to show that it is close enough to its expected value with constant probability.

In the rest of this section we assume Algorithm 4.5 executes its second part. Let W_ℓ and A_ℓ be W and A, respectively, when Algorithm 4.5 exits its loop. We bound W_ℓ, which immediately implies bounds for $\mathsf{opt}_{80(d+2)^2}$. In order to achieve that, we switch our attention from Algorithm 4.5 to an offline algorithm (Algorithm 4.6) producing exactly the same distribution for W_ℓ. We explain intuitively why the distributions of W_ℓ in both algorithms are the same. Let us choose a set T_{pre} ahead, exactly as T is chosen by the online algorithm. Next, we modify the offline algorithm so that whenever it chooses an element from its set T, instead of randomly deciding whether to keep it in T, it queries membership in T_{pre}. Clearly, since there are no repetitions in these queries, this does not affect the behavior of the offline algorithm. However, one can verify that the output of the offline algorithm is now identical to the output of the online algorithm when it selects $T = T_{\text{pre}}$.

Reduction 4.6 Offline W Calculation

Let $A \leftarrow \varnothing$, $W \leftarrow 0$ and $T \leftarrow \mathcal{N}$.
while There exist $u \in T \setminus A$ and $\mathcal{D}_u \subseteq \mathcal{D}^+(u)$ s.t. $A \cup \mathcal{D}_u + u \in \mathcal{I}$ **do**
    Find such a pair maximizing $f(u \mid A \cup \mathcal{D}_u)$.
    With probability of $(d+2)^{-1}$:
        Increase $W \leftarrow W + f(u \mid A \cup \mathcal{D}_u)$ and update $A \leftarrow A \cup \mathcal{D}_u + u$.
    Otherwise:
        Update $T \leftarrow T - u$.
end while

Let L_ℓ be the sum of $f(u \mid A \cup \mathcal{D}_u)$ for all the iterations done by Algorithm 4.6, regardless of the random choice made by the algorithm. We show how to lower bound the expectation of L_ℓ with respect to $f(OPT)$, and then get a concentration result for W_ℓ using a bound on its variance.

Lemma 4.19. $(d+1) \cdot L_\ell \geq f(A_\ell)$.

Brief sketch of proof. The proof is by induction on the number of iterations. Assume the lemma is true for $i-1$ iterations, and let us prove it for iteration i. Trivially, if Algorithm 4.6 randomly chooses not to add elements to A, then the lemma is true for iteration i as well. Now, assume the random choice made is to add the elements to A. Note that only up to $d+1$ elements are added to A, and therefore, it is sufficient to show that the marginal value of each is upper bounded by

the increase in the value of L. Let u be the element chosen by the algorithm. Any dependency of u that appears in T could also be chosen instead of u, so its marginal is upper bounded by the increase in the value of L (since the algorithm uses a greedy choice). On the other hand, any dependency $u' \notin T$ must have been removed when chosen by the algorithm in a previous iteration. Therefore, its marginal value, computed with respect to its optimal dependencies in this previous iteration, is already counted by L_ℓ. Note that the last marginal value must be at least as large as the marginal value of u' with respect to the optimal dependencies in the current iteration (by definition of supermodular dependencies). □

Lemma 4.20. $f(A_\ell) + (d+1) \cdot L_\ell \geq f(OPT)$.

Brief sketch of proof. We consider a hybrid solution starting as OPT and ending as A_ℓ. Using the matroid augmentation property we observe that, when a new element of A is added to this hybrid solution, no more than $d+1$ non A elements have to be removed to restore independence. Then, we bound the "damage" resulting from the removal of these elements using the greedy choice of the algorithm and the definition of supermodular dependencies. Finally, we observe that the value of A_ℓ cannot be increased by adding elements that are still in T, since, otherwise, the algorithm would have done this. Thus, A_ℓ itself is as good as the final hybrid (which contains it). □

An immediate corollary of the last two lemmata is a lower bound of $f(OPT)/(2(d+1))$ on L_ℓ. This bound, together with a concentration result we prove in Section 4.8, shows that $W_\ell \geq f(OPT)/O(d^2)$ with constant probability. The last inequality implies that with constant probability, when $m^* \leq f(OPT)/(256(d+1)^2)$, $\mathsf{opt}_{80(d+2)^2} = W/10$ is indeed a $(80(d+2)^2)$-estimation for the optimum of the part of the input that was not read by the estimation algorithm (*i.e.*, the input for the aided algorithm). The competitive ratio of the second part of Algorithm 4.5 then follows from the competitive ratio of the aided algorithm.

4.7 Assuming our set functions are normalized is without loss of generality

Reduction 4.21. *If ALG is an α-competitive algorithm for the monotone matroid secretary problem under the assumption that f is normalized, then ALG is an α-competitive algorithm also without this assumption.*

Proof. Let $g\colon 2^{\mathcal{N}} \to \mathbb{R}^+$ be the function $g(S) = f(S) - f(\varnothing)$. Notice that g is a non-negative monotone function and $\mathcal{D}_f^+ = \mathcal{D}_g^+$. Moreover, all the oracles that an algorithm for the monotone matroid secretary problem has access to return the same answers for both f and g, and thus, the algorithm produces a random set S with the same distribution when given either f or g as input. Since g is normalized, by the definition of ALG:

$$\mathbb{E}[g(S)] \geq \frac{g(OPT)}{\alpha} \ ,$$

where OPT is a set maximizing g (and f). Thus:

$$\begin{aligned}\mathbb{E}[f(S)] &= \mathbb{E}[g(S)] + f(\varnothing) \\ &\geq \frac{g(OPT)}{\alpha} + f(\varnothing) \geq \frac{f(OPT)}{\alpha} \ .\end{aligned}$$

□

4.8 Full proof for $m^* \leq f(OPT)/(256(d+1)^2)$

In this section we analyze Algorithm 4.5 in the case of a small m^*. We begin with a concentration result proved in Section 4.8.1. The analysis of Algorithm 4.5 appears in Section 4.8.2.

4.8.1 Concentration result

In this section we study a stochastic process consisting of rounds. In each round $i \geq 1$, a positive value $X_i \in (0, B]$ (for some parameter $B > 0$) arrives, and is flagged "accepted" with a probability

$p \in [0,1]$, independently, and "rejected" otherwise. The value X_i itself might depend on the way previous values have been flagged, but not on the way X_i itself is flagged. More formally, let A be the set of indexes corresponding to accepted values, then X_i is a function of the set $A \cap \{1, 2, \ldots, i-1\}$. The process terminates after $T \geq 1$ rounds, where T itself might depend on the way values have been flagged. However, it is guaranteed that T is upper bounded by a finite integer $\bar{T}$ and:

$$\sum_{i=1}^{T} X_i \geq L$$

for some parameter $L \geq 0$.

Let $\Sigma_A = \sum_{i \in A} X_i$ be the random sum of the accepted values. Our objective is to show a concentration bound for Σ_A. Let us first prove such a bound for the case when we make an additional assumption.

Assumption 4.22. *Each value X_i is equal to $B/2^j$ for some value $j \geq 0$.*

Using the above assumption, we can now define some additional notation. Let δ be the smallest number such that some value X_i has a positive probability to take the value δ. Observe that δ is well defined since the above process has only finitely many possible outcomes. By Assumption 4.22, every value X_i is a multiple of δ.

It is helpful to think of the values X_i as intervals placed one after the other on the axis of real numbers, starting from 0. In other words, for every value X_i we have an interval starting at $\sum_{j=1}^{i-1} X_j$ and ending at $\sum_{j=1}^{i} X_j$. Taking this point of view, every interval X_i can be partitioned into X_i/δ ranges of size δ. Let us associate a random variable with each one of these ranges. More formally, for every $i \geq 1$, let Y_i be a random variable taking the value 1 when the range $(\delta(i-1), \delta i)$ is contained within an accepted interval X_j, and the value 0 in all other cases.

Lemma 4.23. *Under Assumption 4.22, it must hold that $\Sigma_A = \delta \cdot \sum_{i=1}^{\bar{T} \cdot B/\delta} Y_i$.*

Proof. Fix a realization of the above process. Recall that Y_i is zero whenever the range $(\delta(i-1), \delta i)$ is not contained within an accepted interval. On the other hand, consider an arbitrary accepted interval X_i. The interval X_i contains X_i/δ ranges of size δ. Moreover, all these ranges end at the point $\sum_{j=1}^{T} X_j \leq \bar{T} \cdot B$ or earlier, and thus, their variables appear in the sum on the right hand

side of the equality we want to prove. Hence, in conclusion, the contribution of X_i to that sum is exactly X_i/δ. The observation now follows since the intervals $\{X_i\}_{i=1}^{T}$ are disjoint, and thus, so are their contributions to the sum. □

Let I be the minimal integer such that $\delta I \geq L$.

Observation 4.24. *Under Assumption 4.22, $\Sigma_A \geq \delta \cdot \sum_{i=1}^{I} Y_i$.*

Proof. Notice that $\bar{T} \cdot B$ is an upper bound on the sum $\sum_{j=1}^{T} X_j$. On the other hand, L is a lower bound on this sum, and thus, we get: $\bar{T} \cdot B \geq L$. Hence, $\delta(\bar{T} \cdot B/\delta) \geq L$. Since the term $\bar{T} \cdot B/\delta$ is an integer, the minimality of I implies $I \leq \bar{T} \cdot B/\delta$. Using Lemma 4.23 and the fact that the variables Y_i are non-negative, we get:

$$\Sigma_A = \delta \cdot \sum_{i=1}^{\bar{T}\cdot B/\delta} Y_i \geq \delta \cdot \sum_{i=1}^{I} Y_i \ .$$

□

Observation 4.24 shows that it is enough for our purpose to prove a concentration bound for $\sum_{i=1}^{I} Y_i$. The following observation gives another useful property of I.

Observation 4.25. *Under Assumption 4.22, for every $1 \leq i \leq I$, the range $(\delta(i-1), \delta i)$ is always contained within some interval X_j.*

Proof. Assume towards a contradiction that there is some realization of the process under which the range $(\delta(i-1), \delta i)$ is not contained within some interval X_j. This implies:

$$L \leq \sum_{j=1}^{T} X_j \leq \delta(i-1) \leq \delta(I-1) \ ,$$

contradicting the definition of I. □

Let us now study the distribution of the variables $\{Y_i\}_{i=1}^{I}$.

Lemma 4.26. *Under Assumption 4.22,* $\Pr[Y_i = 1] = p$ *for every* $1 \leq i \leq I$*. Hence, by linearity of expectation:*

$$\mathbb{E}\left[\sum_{i=1}^{I} Y_i\right] = \sum_{i=1}^{I} \mathbb{E}\left[Y_i\right] = pI \ .$$

Proof. For every $j \geq 1$, let $\mathcal{E}_j$ be the event that $j \leq T$ and $(\delta(i-1), \delta i)$ is included in the interval X_j. Observe that $\mathcal{E}_j$ depends only the acceptance of intervals $X_{j'}$ for $j' < j$. Moreover, given that X_j exists it is accepted with probability p, independently of the acceptance of previous intervals. Hence, we get:

$$\Pr[Y_i = 1 \mid \mathcal{E}_j] = p \ .$$

The observation now follows by the law of total probability since Observation 4.25 guarantees that $(\delta(i-1), \delta i)$ is included in some interval, and thus, the event $\mathcal{E}_j$ happens for exactly a single value of j. □

For every $1 \leq h \leq B/\delta$, let us define $V_h = \{1 \leq i \leq I \mid i \equiv h \pmod{B/\delta}\}$. Observe that the sets $\{V_h\}_{h=1}^{B/\delta}$ form a disjoint partition of the indexes from 1 to I.

Observation 4.27. *Under Assumption 4.22, for every* $1 \leq h \leq B/\delta$ *and two different indexes* $i, i' \in V_h$*, the ranges* $(\delta(i-1), \delta i)$ *and* $(\delta(i'-1), \delta i')$ *cannot be contained in one interval* X_j*.*

Proof. Assume without loss of generality that $i < i'$. The definition of V_h guarantees that $i + B/\delta \leq i'$. Hence,

$$\delta i' - \delta(i-1) = \delta(i'-i) + \delta \geq B + \delta \ .$$

Hence, any interval X_j containing both ranges $(\delta(i-1), \delta i)$ and $(\delta(i'-1), \delta i')$ must be of length at least $B + \delta$, which contradicts the definition of the process. □

Lemma 4.28. *Under Assumption 4.22, for every* $1 \leq h \leq B/\delta$*, the variables of* $\{Y_i \mid i \in V_h\}$ *are independent.*

Proof. For every $i \in V_h$, let $V_h^{<i}$ denote the intersection $V_h \cap \{1, 2, \ldots, i-1\}$. To prove the lemma it is enough to show that for every $i \in V_h$ the variable Y_i takes the value 1 with probability p conditioned on any assignment to the variables of $\{Y_j \mid j \in V_h^{<i}\}$. Let $\mathcal{A}$ denote an arbitrary such

assignment having a non-zero probability. For every $1 \leq \ell \leq \bar{T}$, let $\mathcal{E}_\ell$ be the event that the interval X_ℓ exists and contains the range $(\delta(i-1), \delta i)$.

Assume $\mathcal{E}_\ell$ happens. By Observation 4.27 the variables of $\{Y_j \mid j \in V_h^{<i}\}$ correspond to ranges contained in intervals before X_ℓ. Thus, the values of these variables only imply information about the acceptance of these intervals. Since the acceptance of X_ℓ is independent of the acceptance of previous intervals, we get that every $1 \leq \ell \leq \bar{T}$ obeying $\Pr[\mathcal{E}_\ell \mid \mathcal{A}] > 0$ must also obey:

$$\Pr[Y_i = 1 \mid \mathcal{E}_\ell, \mathcal{A}] = p \ .$$

Clearly the events $\{\mathcal{E}_\ell\}_{\ell=1}^{\bar{T}}$ are disjoint. By Observation 4.25 we also know that one of them must happen. Hence, we get:

$$\begin{aligned}\Pr[Y_i \mid \mathcal{A}] &= \sum_{\substack{1 \leq \ell \leq \bar{T} \\ \Pr[\mathcal{E}_\ell \mid \mathcal{A}] > 0}} \Pr[\mathcal{E}_\ell \mid \mathcal{A}] \cdot \Pr[Y_i = 1 \mid \mathcal{E}_\ell, \mathcal{A}] \\ &= p \cdot \sum_{\ell=1}^{\bar{T}} \Pr[\mathcal{E}_\ell \mid \mathcal{A}] = p \ .\end{aligned}$$

□

Corollary 4.29. *Under Assumption 4.22, for every* $1 \leq h \leq B/\delta$, $\mathrm{Var}\left[\sum_{i \in V_h} Y_i\right] \leq p(L/B + 2)$.

Proof. By Lemma 4.26, for every $i \in V_h$, $\mathrm{Var}[Y_i] = p(1-p) \leq p$. Thus, by Lemma 4.28,

$$\mathrm{Var}\left[\sum_{i \in V_h} Y_i\right] = \sum_{i \in V_h} \mathrm{Var}\left[Y_i\right] \leq \sum_{i \in V_h} p = p \cdot |V_h| \ .$$

The definition of V_h guarantees that its size is at most:

$$|V_h| \leq \left\lceil \frac{I}{B/\delta} \right\rceil \leq \left\lceil \frac{L/\delta + 1}{B/\delta} \right\rceil \leq \frac{L+\delta}{B} + 1 \leq \frac{L}{B} + 2 \ .$$

□

To bound the variance of the sum $\sum_{i=1}^{I} Y_i$, we need the following simple technical lemma.

Lemma 4.30. *For a set of random variables $Z_1, Z_2, \ldots, Z_\ell$, each having a finite variance,*

$$\mathrm{Var}\left[\sum_{i=1}^{\ell} Z_i\right] \leq \left(\sum_{i=1}^{\ell} \sqrt{\mathrm{Var}[Z_i]}\right)^2 .$$

Proof.

$$\begin{aligned}
\mathrm{Var}\left[\sum_{i=1}^{\ell} Z_i\right] &= \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} \mathrm{Cov}[Z_i, Z_j] \\
&\leq \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} \sqrt{\mathrm{Var}[Z_i] \cdot \mathrm{Var}[Z_j]} \\
&= \left(\sum_{i=1}^{\ell} \sqrt{\mathrm{Var}[Z_i]}\right)^2 .
\end{aligned}$$

□

Corollary 4.31. *Under Assumption 4.22, it holds that* $\mathrm{Var}\left[\sum_{i=1}^{I} Y_i\right] \leq pB\delta^{-2}(L+2B)$.

Proof. Observe that:

$$\sum_{i=1}^{I} Y_i = \sum_{h=1}^{B/\delta} \sum_{i \in V_h} Y_i .$$

Hence, by Corollary 4.29 and Lemma 4.30:

$$\begin{aligned}
\mathrm{Var}\left[\sum_{i=1}^{I} Y_i\right] &\leq \left(\frac{B}{\delta} \cdot \sqrt{p\left(\frac{L}{B}+2\right)}\right)^2 \\
&= \frac{pB}{\delta^2}(L+2B) .
\end{aligned}$$

□

We are now ready to prove the promised concentration bound for Σ_A.

Lemma 4.32. *Under Assumption 4.22, for every $t > 0$,* $\Pr[\Sigma_A < pL - t] \leq pB(L+2B)/t^2$.

Proof. By Lemma 4.24,

$$\begin{aligned}\Pr[\Sigma_A < pL - t] &\leq \Pr\left[\delta \cdot \sum_{i=1}^{I} Y_i < pL - t\right] \\ &\leq \Pr\left[\delta \cdot \sum_{i=1}^{I} Y_i < \delta \cdot pI - t\right] \\ &\leq \Pr\left[\left|\delta \cdot \sum_{i=1}^{I} Y_i - \delta \cdot pI\right| > t\right] \quad .\end{aligned}$$

Since the expected value of $\delta \cdot \sum_{i=1}^{I} Y_i$ is $\delta \cdot pI$ by Lemma 4.26, we get by Chebyshev's inequality:

$$\begin{aligned}\Pr\left[\left|\delta \cdot \sum_{i=1}^{I} Y_i - \delta \cdot pI\right| > t\right] &\leq \frac{\mathrm{Var}\left[\delta \cdot \sum_{i=1}^{I} Y_i\right]}{t^2} \\ &= \frac{\delta^2 \cdot \mathrm{Var}\left[\sum_{i=1}^{I} Y_i\right]}{t^2} \\ &\leq \frac{pB(L+2B)}{t^2} \quad ,\end{aligned}$$

where the last inequality holds by Corollary 4.31. □

Finally, we would like to get a version of Lemma 4.32 that holds without Assumption 4.22.

Corollary 4.33. *For every $t > 0$, $\Pr[\Sigma_A < pL/2 - t] \leq pB(L+2B)/(4t^2)$*

Proof. For every value X_i, let us define a value X'_i as follows:

$$X'_i = B/2^{\lfloor \log_2(B/X_i) \rfloor} \quad .$$

Intuitively, X'_i is the smallest value allowed by Assumption 4.22 that is at least as large as X_i. We say that X'_i is accepted if and only if X_i is. One can verify that the following holds:

- The values $X'_1, X'_2, \ldots, X'_T$ define a legal process with the same parameters p, B and L as the original process, and this process obeys Assumption 4.22. Let Σ'_A be the sum of the accepted values in this process.

- For every value X_i, $X'_i \leq 2X_i$, hence, $\Sigma'_A \leq 2 \cdot \Sigma_A$.

Thus, by Lemma 4.32:

$$\begin{aligned}\Pr[\Sigma_A < pL/2 - t] &\leq \Pr[\Sigma'_A < pL - 2t] \\ &\leq \frac{pB(L+2B)}{(2t)^2} \\ &= \frac{pB(L+2B)}{4t^2} \ .\end{aligned}$$

□

4.8.2 Proof for $m^* \leq f(OPT)/(256(d+1)^2)$

In this section we prove that Algorithm 4.5 is $O(\beta)$-competitive when $m^* \leq f(OPT)/(256(d+1)^2)$. Recall that Algorithm 4.5 consists of two parts. Let us say that Algorithm 4.5 "applies the second option" when it executes the second part (the one that involves the aided algorithm). Notice that it is enough to show that the algorithm is $O(\beta)$-competitive when it applies the second option, since this option is applied with probability 1/2.

We need some additional notation. First, we denote by $\ell+1$ the number of iterations performed by the loop on Line 6 of the algorithm (*i.e.*, the $\ell+1$ iteration is the iteration at which the algorithm decides to leave the loop, and does not change A). Additionally, for every $1 \leq i \leq \ell$, let A_i and W_i denote the set A and the value W, respectively, immediately after the i-th iteration of this loop. For consistency, we also denote by A_0 and W_0 these set and value before the first iteration. Finally, for every $1 \leq i \leq \ell$, let u_i denote the element u chosen at iteration i of the loop. We begin the analysis of the small m^* case by proving an upper bound on W_ℓ (the final value of W).

Observation 4.34. *When Algorithm 4.5 applies the second option, $A_i \in \mathcal{I}$ for every $0 \leq i \leq \ell$.*

Proof. The observation holds since Algorithm 4.5 chooses at every iteration an element u and a set $\mathcal{D}_u$ whose addition to A does not violate independence. □

Lemma 4.35. *When Algorithm 4.5 applies the second option, $W_\ell \leq f(OPT)$.*

Proof. We prove by induction on i the claim that the inequality $W_i \leq f(A_i)$ holds for every $0 \leq i \leq \ell$. Notice that the observation follows from this claim since, by Observation 4.34, A_ℓ is independent, and thus, $f(A_\ell) \leq f(OPT)$.

For $i = 0$ the claim is trivial since $W_0 = 0 = f(\varnothing) = f(A_0)$. For $i > 0$, assume the claim holds for $i - 1$, and let us prove it for i.

$$\begin{aligned} W_i &= W_{i-1} + f(u_i \mid A_{i-1} \cup \mathcal{D}_{u_i}) \\ &\leq f(A_{i-1}) + f(u_i \mid A_{i-1} \cup \mathcal{D}_{u_i}) \\ &\leq f(A_{i-1} \cup \mathcal{D}_{u_i}) + f(u_i \mid A_{i-1} \cup \mathcal{D}_{u_i}) = f(A_i) \ , \end{aligned}$$

where the first inequality holds by the induction hypothesis, and the second inequality follows from the monotonicity of f. □

Our next objective is to prove a lower bound on W_ℓ that holds with a constant probability. For that purpose, let us consider an offline algorithm which calculates a value W having the same distribution as the value W calculated by Algorithm 4.5.

Reduction 4.7 Offline W Calculation

1: Let $A \leftarrow \varnothing$, $W \leftarrow 0$ and $T \leftarrow \mathcal{N}$.
2: **while** there exist $u \in T \setminus A$ and $\mathcal{D}_u \subseteq \mathcal{D}^+(u)$ s.t. $A \cup \mathcal{D}_u + u \in \mathcal{I}$ **do**
3: Find such a pair maximizing $f(u \mid A \cup \mathcal{D}_u)$.
4: With probability of $(d+2)^{-1}$:
5: Increase $W \leftarrow W + f(u \mid A \cup \mathcal{D}_u)$ and update $A \leftarrow A \cup \mathcal{D}_u + u$.
6: Otherwise:
7: Update $T \leftarrow T - u$.
8: **end while**

Observation 4.36. *The distribution of the value W calculated by Algorithm 4.7 is identical to the distribution of W_ℓ as calculated by Algorithm 4.5 when it applies the second option.*

Proof. During each iteration of the loop starting on Line 6 of Algorithm 4.5, the algorithm identifies a pair constituted of an element u and a set $\mathcal{D}_u$ maximizing $f(u \mid A \cup \mathcal{D}_u)$ and obeying some other conditions, including the requirement that u belongs to a set T containing every element with probability $(d+2)^{-1}$, independently.

On the other hand, Algorithm 4.7 has a loop that looks for a pair of an element u and a set $\mathcal{D}_u$ maximizing $f(u \mid A \cup \mathcal{D}_u)$ and obeying the same conditions, except for requiring u to belong to T. Once such a pair is found, the algorithm makes a random decision whether to keep u in T, and then uses the pair to increase the solution A if and only if it decides to keep u in T.

Notice that the only difference between these two procedures is the point when the algorithm decides whether each element u should belong to T. Algorithm 4.5 makes the decisions for all elements at the beginning, while Algorithm 4.7 makes the decisions only when necessary. However, regardless of when the membership of elements in T is decided, the set of pairs used to increase the solution A is the same given that the same random decisions are made by both algorithms. □

To analyze the distribution of the value W calculated by Algorithm 4.7 we need some additional notation. Let $\hat{\ell}+1$ be the number of iterations performed by the loop of Algorithm 4.7 (*i.e.*, $\hat{\ell}$ is the number of times elements are added to the set A), and for every $1 \leq i \leq \hat{\ell}$ let $\hat{A}_i$, $\hat{T}_i$ and $\hat{W}_i$ denote the sets A and T and the value W, respectively, immediately after the i-th iteration of this loop. For consistency, we also denote by $\hat{A}_0$, $\hat{T}_0$ and $\hat{W}_0$ these sets and value before the first iteration. Additionally, for every $1 \leq i \leq \hat{\ell}$, let $\hat{u}_i$ denote the element u chosen at iteration i of the loop. Finally, for every $0 \leq i \leq \hat{\ell}$, let us denote by OPT_i the maximum value independent set that can be obtained from $\hat{A}_i$ by adding only T elements. Formally,

$$OPT_i = \operatorname*{arg\,max}_{S \in \mathcal{I} \mid \hat{A}_i \subseteq S \subseteq \hat{A}_i \cup \hat{T}_i} f(S) \ .$$

Observe that OPT_i is well defined since the set $\hat{A}_i$ is independent for every $0 \leq i \leq \hat{\ell}$.

Next, observe that Algorithm 4.7 can be viewed as a process of the kind described in Section 4.8.1. More precisely, each iteration i of Algorithm 4.7 corresponds to one iteration of the process, the value of this iteration is $f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$ and this value is accepted if and only if $\hat{u}_i$ is kept in T (and $f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$ is added to W). Note that, as required by the process definition, the value $f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$ depends only on the acceptances of previous values, and the acceptances of the value $f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$ is independent of anything else. Taking this point of view, $W_{\hat{\ell}}$ is exactly the sum of the accepted values, and thus, can be analyzed using Corollary 4.33. To use the last

corollary, we need to determine the parameters of the process:

- By definition, m^* upper bounds $f(\hat{u}_i \mid (\hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) \cap \mathcal{D}^+(u)) \geq f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$. Hence, one can choose $B = f(OPT)/[256(d+1)^2] \geq m^*$ for the process given by Algorithm 4.7.
- Every value is accepted with probability $(d+2)^{-1}$, thus, this is the value of p.

We are left to determine a possible value for the parameter L. For that purpose we need a few claims.

Observation 4.37. $OPT_{\hat{\ell}} = \hat{A}_{\hat{\ell}}$.

Proof. By definition $OPT_{\hat{\ell}}$ contains the elements of $\hat{A}_{\hat{\ell}}$ and (possibly) additional elements of $\hat{T}_{\hat{\ell}}$ that do not violate independence when added to $\hat{A}_{\hat{\ell}}$. However, the fact that Algorithm 4.7 stopped during the $\hat{\ell}+1$ iteration implies that no elements of $\hat{T}_{\hat{\ell}}$ can be added to $\hat{A}_{\hat{\ell}}$ without violating independence. Therefore, $OPT_{\hat{\ell}}$ cannot contain any elements beside the elements of $\hat{A}_{\hat{\ell}}$. □

For every $0 \leq i \leq \hat{\ell}$, let L_i be the sum of the first i values of the process corresponding to Algorithm 4.7. Formally,

$$L_i = \sum_{j=1}^{i} f(\hat{u}_j \mid \hat{A}_{j-1} \cup \mathcal{D}_{\hat{u}_j}) \ .$$

Lemma 4.38. *For every* $0 \leq i \leq \hat{\ell}$, $f(\hat{A}_i) \leq (d+1) \cdot L_i$.

Proof. We prove by induction on i that for every $0 \leq i \leq \hat{\ell}$:

$$f(\hat{A}_i) \leq (d+1) \cdot \sum_{\hat{u}_j \in \hat{A}_i} f(\hat{u}_j \mid \hat{A}_{j-1} \cup \mathcal{D}_{\hat{u}_j}) \ . \tag{5}$$

Observe that the lemma follows from the last claim since $\hat{u}_j$ can belong to the set $\hat{A}_i$ only when $j \leq i$. For $i = 0$, Equation (5) is trivial since $f(\hat{A}_0) = f(\varnothing) = 0$. Next, assume Equation (5) holds for $i - 1 \geq 0$, and let us prove it for i. If $\hat{u}_i$ is removed from T then this is true since in this case $\hat{A}_i = \hat{A}_{i-1}$. Thus, we can safely assume in the rest of the proof that $\hat{u}_i$ remains in T.

For every $0 \leq i \leq \hat{\ell}$, let $N_i = \{\hat{u}_j \notin A_i \mid 1 \leq j \leq i\}$. Order the elements of $\hat{A}_i \setminus \hat{A}_{i-1}$ in an arbitrary order, and let v_j denote the j-th element in this order. Consider some $1 \leq j \leq |\hat{A}_i \setminus \hat{A}_{i-1}|$,

if $v_j \notin N_{i-1}$ then the pair of the element v_j and the set $\{v_1, v_2, \ldots, v_{i-1}\} \cap \mathcal{D}^+(v_j)$ form a possible pair that Algorithm 4.7 could select on iteration i since $v_j \in \mathcal{N} \setminus (N_{i-1} \cup \hat{A}_{i-1}) = \hat{T}_{i-1} \setminus \hat{A}_{i-1}$. Hence,

$$\begin{aligned} & f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) \\ & \quad \geq f(v_j \mid \hat{A}_{i-1} \cup (\{v_1, v_2, \ldots, v_{i-1}\} \cap \mathcal{D}^+(v_j))) \\ & \quad \geq f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \ldots, v_{i-1}\}) \ . \end{aligned}$$

On the other hand, if $v_j \in N_{i-1}$, then there must be some $1 \leq h < i$ such that $v_j = \hat{u}_h$, and thus, $v_j \in \hat{T}_h$. By a similar argument to the one used above we get:

$$\begin{aligned} & f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) \\ & \geq f(v_j \mid \hat{A}_{h-1} \cup ((\hat{A}_{i-1} \cup \{v_1, v_2, \ldots, v_{i-1}\}) \cap \mathcal{D}^+(v_j))) \\ & \geq f(v_j \mid \hat{A}_{h-1} \cup (\hat{A}_{i-1} \cup \{v_1, v_2, \ldots, v_{i-1}\})) \\ & = f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \ldots, v_{i-1}\}) \ . \end{aligned}$$

Adding the inequalities we got for every $1 \leq j \leq |\hat{A}_i \setminus \hat{A}_{i-1}|$ gives:

$$\begin{aligned} & f(\hat{A}_i) - f(\hat{A}_{i-1}) \\ & \quad = \sum_{j=1}^{|\hat{A}_i \setminus \hat{A}_{i-1}|} f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \ldots, v_{i-1}\}) \\ & \quad = \sum_{v_j \in \hat{A}_i \setminus (\hat{A}_{i-1} \cup N_{i-1})} f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \ldots, v_{i-1}\}) + \\ & \qquad \sum_{v_j \in \hat{A}_i \cap N_{i-1}} f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \ldots, v_{i-1}\}) \\ & \quad \leq |\hat{A}_i \setminus (\hat{A}_{i-1} \cup N_{i-1})| \cdot f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) + \\ & \qquad \sum_{u_h \in \hat{A}_i \cap N_{i-1}} f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) \ . \end{aligned}$$

Observe that $|\hat{A}_i \setminus (\hat{A}_{i-1} \cup N_{i-1})| \leq d + 1$ and $\hat{A}_i \cap N_{i-1} \subseteq \hat{A}_i \setminus \hat{A}_{i-1} - \hat{u}_i$. Combining both

observations with the last inequality yields:

$$\begin{aligned} f(\hat{A}_i) &\leq f(\hat{A}_{i-1}) + (d+1) \cdot f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_j}) + \\ &\quad \sum_{u_h \in \hat{A}_i \setminus \hat{A}_{i-1} - \hat{u}_i} f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) \\ &\leq f(\hat{A}_{i-1}) + (d+1) \cdot \sum_{u_h \in \hat{A}_i \setminus \hat{A}_{i-1}} f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) \\ &\leq (d+1) \cdot \sum_{u_h \in \hat{A}_i} f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) \ , \end{aligned}$$

where the last inequality holds by the induction hypothesis. □

Lemma 4.39. *For every* $0 \leq i \leq \hat{\ell}$, $f(OPT_i) + (d+1) \cdot L_i \geq f(OPT)$.

Proof. We prove the lemma by induction on i. For $i = 0$ the lemma is trivial since $f(OPT_0) = f(OPT)$. Next, assume that the lemma holds for $i - 1 \geq 0$, and let us prove it for i. There are two cases to consider. First, let us consider the case where $\hat{u}_i$ is removed from T and $\hat{A}_i = \hat{A}_{i-1}$. In this case one potential candidate for OPT_i is $OPT_{i-1} - \hat{u}_i$. If $\hat{u}_i \notin OPT_{i-1}$, then we get $f(OPT_{i-1} - \hat{u}_i) = f(OPT_{i-1})$. On the other hand, if $\hat{u}_i \in OPT_{i-1}$ then the pair of the element $\hat{u}_i$ and the set $(OPT_{i-1} \setminus \hat{A}_{i-1}) \cap \mathcal{D}^+(\hat{u}_i)$ is a possible pair that Algorithm 4.7 could select on iteration i. Hence,

$$\begin{aligned} & f(OPT_{i-1} - \hat{u}_i) \\ &= f(OPT_{i-1}) - f(\hat{u}_i \mid \hat{A}_{i-1} \cup (OPT_{i-1} \setminus \hat{A}_{i-1} - \hat{u}_i)) \\ &\geq f(OPT_{i-1}) - \\ &\quad f(\hat{u}_i \mid \hat{A}_{i-1} \cup ((OPT_{i-1} \setminus \hat{A}_{i-1}) \cap \mathcal{D}^+(\hat{u}_i))) \\ &\geq f(OPT_{i-1}) - f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) \ . \end{aligned}$$

Therefore, regardless of $\hat{u}_i$'s membership in OPT_{i-1}, we can lower bound by $f(OPT_{i-1} - \hat{u}_i)$ by

$f(OPT_{i-1}) - f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$. Using the induction hypothesis, we now get:

$$\begin{aligned} f(OPT_i)&+(d+1)\cdot L_i \\ &\geq f(OPT_{i-1} - \hat{u}_i) + (d+1)\cdot L_{i-1} + \\ &\quad (d+1)\cdot f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) \\ &\geq f(OPT_{i-1}) + (d+1)\cdot L_{i-1} \geq f(OPT) \ . \end{aligned}$$

Next, let us consider the case where $\hat{u}_i$ is kept in T and $\hat{A}_i = \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i} + \hat{u}_i$. In this case, by standard matroid properties, one can obtain a candidate for OPT_i by starting with $OPT_{i-1} \cup \mathcal{D}_{\hat{u}_i} + \hat{u}_i$ and removing from it a subset $\Delta \subseteq OPT_{i-1} \setminus \hat{A}_i$ of up to $d+1$ elements. Let us denote by OPT'_i this candidate, *i.e.*, $OPT'_i = (OPT_{i-1} \cup \mathcal{D}_{\hat{u}_i} + \hat{u}_i) \setminus \Delta$.

Order the elements of Δ in an arbitrary order, and let v_j denote the j-th element in this order. Observe that by monotonicity:

$$\begin{aligned} f(OPT_{i-1}) &- f(OPT'_i) \\ &\leq f(OPT_{i-1}) - f(OPT_{i-1} \setminus \Delta) \\ &= \sum_{j=1}^{|\Delta|} f(v_j \mid OPT_{i-1} \setminus \{v_1, v_2, \ldots, v_j\}) \\ &\leq \sum_{j=1}^{|\Delta|} f(v_j \mid A_{i-1} \cup \\ &\quad ((OPT_{i-1} \setminus \{v_1, v_2, \ldots, v_j\}) \cap \mathcal{D}^+(v_j))) \\ &\leq (d+1)\cdot f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}^+(\hat{u}_i)) \ . \end{aligned}$$

where the last inequality holds since $v_j \in OPT_{i-1} \setminus \hat{A}_i \subseteq \hat{T}_{i-1}$ for every $1 \leq j \leq |\Delta|$, and thus, the pair of the element v_j and the set $[(OPT_{i-1} \setminus \{v_1, v_2, \ldots, v_j\}) \setminus \hat{A}_{i-1}] \cap \mathcal{D}^+(v_j)$ is a possible pair that

Algorithm 4.7 could select on iteration i. Using the induction hypothesis, we now get:

$$\begin{aligned} f(OPT_i)+&(d+1)\cdot L_i \\ &\geq f(OPT'_i) + (d+1)\cdot L_{i-1}+ \\ &\quad (d+1)\cdot f(\hat{u}_i \mid \hat{A}_{i-1}\cup \mathcal{D}_{\hat{u}_i}) \\ &\geq f(OPT_{i-1}) + (d+1)\cdot L_{i-1} \geq f(OPT) \ . \end{aligned}$$

□

Corollary 4.40. *In the process corresponding to Algorithm 4.7 the parameter L of can be chosen to be $f(OPT)/[2(d+1)]$.*

Proof. Observe that any value that always lower bounds $L_{\hat{\ell}}$ can be used as a value for the parameter L. Combining Lemmata 4.38 and 4.39 gives:

$$2(d+1)\cdot L_{\hat{\ell}} \geq f(\hat{A}_{\hat{\ell}}) + [f(OPT) - f(OPT_{\hat{\ell}})] = f(OPT) \ ,$$

where the equality holds by Observation 4.37. □

Now that we have all the parameters of the process corresponding to Algorithm 4.7, we can use Corollary 4.33 to give a guarantee on W_ℓ.

Lemma 4.41. *When Algorithm 4.5 applies the second option and $m^* \leq f(OPT)/[256(d+1)^2]$, then, with probability at least $7/8$, $W_\ell \geq \frac{f(OPT)}{8(d+2)^2}$.*

Proof. Recall that $\hat{W}_{\hat{\ell}}$ is the sum of the accepted values in the process corresponding to Algo-

rithm 4.7. Hence, by Corollary 4.33,

$$\begin{aligned}
&\Pr\left[\hat{W}_{\hat{\ell}} < \frac{f(OPT)}{8(d+2)^2}\right] \\
&\leq \Pr\left[\hat{W}_{\hat{\ell}} < \frac{pL}{2} - \frac{pL}{4}\right] \\
&\leq \frac{pB(L+2B)}{4(pL/4)^2} = \frac{4B(L+2B)}{pL^2} \\
&= \frac{4 \cdot \frac{f(OPT)}{256(d+1)^2} \cdot \left(\frac{f(OPT)}{2(d+1)} + \frac{2 \cdot f(OPT)}{256(d+1)^2}\right)}{\frac{1}{d+2} \cdot \left(\frac{f(OPT)}{2(d+1)}\right)^2} \\
&\leq \frac{4 \cdot \frac{1}{256(d+1)^2} \cdot \frac{1}{d+1}}{\frac{1}{d+2} \cdot \frac{1}{4(d+1)^2}} = \frac{(d+2)}{16(d+1)} \leq \frac{1}{8} \ .
\end{aligned}$$

The lemma now follows since W_ℓ and $\hat{W}_{\hat{\ell}}$ have the same distribution by Observation 4.36. □

To complete the analysis of Algorithm 4.5 we also need the following notation and lemma. Given a set $S \subseteq \mathcal{N}$, let $S(p)$ be a random set containing every element $u \in S$, independently, with probability p.

Lemma 4.42. *For every set $S \subseteq \mathcal{N}$, $\mathbb{E}[f(S(p))] \geq p^{d+1} \cdot f(S)$.*

Proof. For every element $u \in S$, let X_u be an indicator for the event that $S \cap \mathcal{D}^+(u) + u \in S(p)$. Clearly, $\Pr[X_u = 1] \geq p^{d+1}$. Also, let $u_1, u_2, \ldots, u_{|S|}$ denote an arbitrary order of the elements of S. Then,

$$\begin{aligned}
&\mathbb{E}[f(S(p))] \\
&= \sum_{i=1}^{|S|} \mathbb{E}[f(\{u_i\} \cap S(p) \mid S(p) \cap \{u_1, u_2, \ldots, u_{i-1}\})] \\
&\geq \sum_{i=1}^{|S|} \mathbb{E}[X_i \cdot f(u_i \mid \{u_1, u_2, \ldots, u_{i-1}\})] \\
&\geq p^{d+1} \cdot \sum_{i=1}^{|S|} f(u_i \mid \{u_1, u_2, \ldots, u_{i-1}\}) = p^{d+1} \cdot f(S) \ ,
\end{aligned}$$

where the first inequality follows from the definition of $\mathcal{D}^+(u)$. □

Lemma 4.43. *If $m^* \leq f(OPT)/[256(d+1)^2]$, then Algorithm 4.5 is $O(\beta)$-competitive.*

Proof. Throughout this proof we assume that Algorithm 4.5 applies the second option. This event happens with probability 1/2, thus, it is enough to prove that Algorithm 4.5 is $O(\beta)$-competitive given this event.

Let $R = \mathcal{N} \setminus T_\ell$ be the set of elements that are not observed by Algorithm 4.5. By Lemma 4.42:

$$\begin{aligned}\mathbb{E}[f(OPT \cap R)] &= \mathbb{E}\left[f\left(OPT\left(1 - \frac{1}{d+2}\right)\right)\right] \\ &\geq \left(1 - \frac{1}{d+2}\right)^{d+1} \cdot f(OPT) \\ &\geq e^{-1} \cdot f(OPT) \ .\end{aligned}$$

Hence,

$$\begin{aligned}\mathbb{E}[f(OPT(R))] &\geq \mathbb{E}[f(OPT \cap R)] \\ &\geq \frac{f(OPT)}{e} \geq \frac{f(OPT)}{4} \ ,\end{aligned}$$

where $OPT(R)$ is the independent subset of R maximizing f. Since $f(OPT(R))$ is always upper bounded by $f(OPT)$, this implies the following claim:

$$\Pr\left[f(OPT(R)) \geq \frac{f(OPT)}{10}\right] \geq \frac{1}{6} \ .$$

On the other hand, $W_\ell \geq f(OPT)/[8(d+2)^2]$ with probability at least 7/8 by Lemma 4.41. Hence, by the union bound we have:

$$W_\ell \geq \frac{f(OPT)}{8(d+2)^2} \qquad \text{and} \qquad f(OPT(R)) \geq \frac{f(OPT)}{10}$$

with probability at least 1/24. To complete the proof it is enough to show that ALG is $O(\beta)$-competitive when the last two inequalities hold. This follows from the definition of ALG when $\mathsf{opt}_{80(d+2)^2}$ is a valid approximation for $f(OPT(R))$, *i.e.*, when we have $f(OPT(R))/[80(d+2)^2] \leq$

$\mathsf{opt}_{80(d+2)^2} \leq f(OPT(R))$. Thus, in the rest of the proof we prove these inequalities:

$$\mathsf{opt}_{80(d+2)^2} = \frac{W_\ell}{10} \leq \frac{f(OPT)}{10} \leq f(OPT(R)) \ ,$$

where the first inequality follows from Lemma 4.35. On the other hand,

$$\begin{aligned} f(OPT(R)) \leq f(OPT) &\leq 8(d+2)^2 W_\ell \\ &= 80(d+2) \cdot \mathsf{opt}_{80(d+2)^2} \ . \end{aligned}$$

□

Note that Theorem 4.17 follows immediately from Lemmata 4.18 and 4.43.

5 Working Together: Committee Selection and the Supermodular Degree

This section is based on [64].

Consider the following scenario (see, e.g., [30, 83]). An airline is willing to increase the satisfaction of the travelers by letting them choose the set of movies that will be available on their flight. It is decided to store on the airplane some fixed number k of movies. The airline surveys the preferences of the prospective passengers of the flight, and is willing to make the best decision given their preferences. Two questions immediately arise. First, how should the preferences of the prospective travelers be modeled? Second, given the preferences of the travelers, how should the set of movies be chosen? This problem of choosing some fixed number of candidates to the satisfaction of the voters is a fundamental problem. Generally speaking, we have a set V of n voters and a set C of m candidates, and we would like to select k candidates out of the m, such that the voters will be most satisfied. The answers to the two questions above vary in the literature. For example, by the Chamberlin-Courant rule, each voter has a value for each of the candidates, and the satisfaction of a voter is measured by the highest value she has for any elected candidate. The overall satisfaction is either the sum of the values of the voters or the value of the least satisfied voter (utilitarian [16] or egalitarian [9] variant, respectively). Other possibilities are to aggregate for every voter her value for every elected candidate or to give higher weight for candidates ranked higher by her (e.g. Borda rule). In a recent work Skowron, Faliszewski and Lang [83] introduce an elegant model that captures the latter examples as well as others. They model the preferences of each voter by an intrinsic value for each of the candidates. Then, they calculate the value attributed by a voter to a set of k candidates, by ordering her k intrinsic values for the k candidates, and multiplying them by some weight that corresponds to their rank in the order. This vector of weights is called "OWA operator" (Ordered Weighted Average). Skowron, Faliszewski and Lang [83] study their model for different restrictions on the OWA vector. Among their results, they show a $(1-1/e)$-approximation algorithm for the case of non-increasing weights OWA vectors, by showing that it is captured by submodular set functions.

However, none of the models above capture positive correlation (i.e. synergy) between specific candidates Such positive correlation can happen in various cases: from two candidates to the parliament that are working great together (see Woolley et al. [88] for a research on collective intelligence), to a series of movies that people tend to prefer watching the latter parts only after watching the former parts. In this paper we suggest a voting rule that captures positive correlation between specific candidates. Specifically, our answers to the two questions above are:

- The preferences of each of the candidates are modeled by a non-decreasing monotone set function from subsets of candidates to non-negative real numbers.
- A set of k candidates that maximizes the sum of values of the voters is elected.

We study applications for the proposed model. As part of our proposed framework, we demonstrate how preference elicitation can be practically done in Section 5.3.1.

Additionally, we study the computability of our voting rule. We show that computing the optimum is, generally, $\mathcal{NP}$-hard, but that one can approximate the optimum with guarantee that depends linearly on the amount of synergy between different candidates. In order to get such approximation guarantees, we extend the supermodular degree (see Section 3) to capture multiple set functions, by introducing the joint supermodular degree. The joint supermodular degree enables us to use existing algorithms for set functions that were designed for the supermodular degree, in order to get approximation algorithms for our voting rule. We justify the naturalness of the joint supermodular degree from an applicative view point in Section 5.3. We formally show the algorithmic result and a hardness result in Section 5.4.

5.1 Our contribution

We introduce a new model for voting rules, based on set functions, together with the required conceptual framework. This model can be used to model both synergy between candidates (i.e. compliments) and substitutes (e.g., two candidates that each of them is worth 1 and both of them together are worth 1, as well). Since general set functions might be highly complex, we introduce the joint supermodular degree, which we see as a natural extension of the supermodular degree of

Feige and Izsak ([36], Section 3). We demonstrate applications for our model in Section 5.3. In particular, we suggest practical preference elicitation that is tailored for the joint supermodular degree in Section 5.3.1.

Finally, in Section 5.4, we show how the joint supermodular degree enables one to easily use existing algorithms for functions maximization that are tailored for the supermodular degree to achieve approximations for our voting rule. Since there exists such algorithms both for offline and online settings, one can use either and immediately get approximation guarantees for our voting rule in the corresponding setting. Moreover, future algorithms for the supermodular degree can also be easily used by our framework, to get computational results for committee selection. Theoretically speaking, the result of the approximation algorithms can also be seen as the voting rule itself (see Skowron, Faliszewski and Lang [83]). We complement our algorithmic result with a proof of computational hardness.

To the best of our knowledge, our results represent the first voting rules that capture synergy between specific candidates.

5.2 The model

We formally define our model. Let $V = \{v_1, \ldots, v_n\}$ be a set of n voters, let C be a set of m candidates and let k be an integer. Let $f_1, \ldots, f_n : 2^C \to \mathbb{R}^+$ be preference (set) functions, associated with the voters $v_1, \ldots, v_n$, respectively. We assume that the preferences functions are normalized (i.e., $\forall_i f_i(\varnothing) = 0$) and non-decreasing monotone (i.e., $\forall_{i,S' \subseteq S \subseteq M} f_i(S') \leq f_i(S)$). Our aim is to choose a set $C_{max} \subseteq C$ of size k that maximizes the satisfaction of the voters by their personal preferences:

$$C_{max} = \operatorname*{argmax}_{S \subseteq C || S| = k} \sum_{i=1}^{n} f_i(S) \ .$$

We refer to this problem as (the) k-COMMITEE SELECTION problem and to the selected subset as the selected committee. Note that this problem can be seen as a voting rule. Alternatively, an approximation algorithm to this problem can be seen as the voting rule (see also Skowron, Faliszewski and Lang [83]).

5.2.1 The joint supermodular degree

We introduce the following natural extensions of the definitions in Section 3 to a collection of set functions.

Definition 5.1. *Let $f_1, \ldots, f_t$ be set functions for some $t \in \mathbb{N}$ and let $c \in C$. The joint supermodular dependency set of c by $f_1, \ldots, f_t$ is $\bigcup_{i=1}^{t} \mathcal{D}^{+}_{f_i}(c)$.*

Definition 5.2. *The joint supermodular degree of $f_1, \ldots, f_t$ is the maximum cardinality among the cardinalities of joint dependency sets of items of C by $f_1, \ldots, f_t$.*

The main property of the joint supermodular degree that we use is that the sum function of functions with joint supermodular degree of at most d has supermodular degree of at most d.

We think that this definition is natural for voting rules, since it means that positive correlation between the candidates can be modeled, when it is inherent to the candidates themselves, rather than to the perspective of the voters about them.

For example, if a candidate is working well together with 2 other candidates, then each of the voters has the possibility to give these 3 candidates or any subset of them a score that is higher than the sum of their individual scores. However, if a candidate does not work well with some other candidate, then none of the voters has the possibility to give them together a score that is higher than the sum of their individual scores. That is, the set of other candidates that the candidate has synergy with depends on the candidate herself. The decision of whether to take this into account depends on each of the voters. So, the supermodular dependency set of a candidate c, by any of the preference functions of the voters, will contain only other candidates that have synergy (i.e. are working well together) with c.

5.3 Applications

We discuss in this section applications of our model. Specifically, we demonstrate its merits for three real world examples (see [30]).

- Parliamentary elections: In voting to the parliament, it is possible that candidates complement each other, and work better together. It was actually shown by Woolley et al. [88] that there is a measure for the collective intelligence of a group of people that is different from the intelligence quantities of different people in the group. So, it seems reasonable to allow the voters to give extra value for choosing *together* a pair of candidates that are known to work well together on, e.g., suggesting complex laws in the parliament. Note that the fact that two candidates are working well together is related to the candidates and not to the voters, and indeed, the joint supermodular degree of the voters will reflect the synergies between the candidates.

- Movie selection: Consider the problem of choosing k movies to be available on an airplane (passengers can watch on their flight movies from the selected set). It seems reasonable that people would prefer to watch latter parts of a series only after the former. Moreover, it might be unreasonable to consider a series of movies as one movie, if, e.g., physical storage is a limitation. Then, it is plausible to give the prospective passengers the possibility to give higher values for movies in the series, given that all the former are selected, as well. Additionally, movie selection can admit submodular behaviour (i.e. substitutes). For example, since the time of the flight is bounded, the number of movies one can watch out of the k selected movies is bounded, as well. This means that, if for example, $k = 100$ and the time of the flight allows one passenger to watch up to 5 movies, then any movie out of the k that is not among the 5 best for that passenger is redundant for her. So her value will not increase given that we add to the selected set other great movies. On the other hand, we do want to allow k to be large enough to allow different passengers to enjoy different movies. The latter behaviour is submodular. Synergy between selected movies is supermodular. Our model enables one to express such preferences. Furthermore, submodularity does not hurt the approximation guarantees, since it does not increase the joint supermodular degree of the preference functions (see Section 5.4).

5.3.1 Preference elicitation

Consider the movies selection example. When a prospective passenger is asked to express her preferences about possible movies, it seems unreasonable to require her to specify her values for all the exponentially many possibilities (as needed theoretically in the general case of a valuation function). In this section we demonstrate a simple user interface to express some real world preferences in that case, while enabling the users to benefit from the possibility of expressing positive correlations.

The user interface will be as follows. Each of the prospective passengers will be able to give a value for each of the possible movies (these are the values of the singleton subsets). In addition, the prospective passengers will be able to add for each of the movies other values – the marginal values of a movie, with respect to a subset of its joint supermodular dependency set. In order to select such a subset of the movies, a list of the movies in the joint supermodular dependency set will be presented, and a passenger will be able to select the relevant movies (e.g. by checking them by a 'V'). In order to enforce the preference functions of the prospective passengers to be well defined (i.e. a single value for each of the subsets), we will let the prospective passengers check by a 'V' only the movies that were former to a movie in a series.

Note that the supermodular dependency is symmetric (see Section 3.9 for a proof). So, in a series of movies, also the former movies are dependent on the latter movies. As an example, one can think of two movies, where each of them is worth 1, but the second one is worth 10 with respect to the first. Then, both movies together are worth 11, and the marginal contribution of each of them with respect to the other is 10, instead of 1 (as it is with respect to the empty set).

Generally speaking, this example interface can be extended in any way that enforces the preference functions to be well defined (e.g. by ordering the items and letting the prospective passengers to check a dependency by 'V' only if it is before the current item in that ordering).

To see the power of combining supermodular dependencies with submodular behaviour, note that we can also ask each passenger how many movies she would like to watch in her flight (with a maximum that depends on the duration of the flight), and then calculate as her preference, the best subset of that number of movies, from any input subset of movies.

Note that it is easy to emulate both value and supermodular queries using such a representation, and then to use the algorithms of Feldman and Izsak [41], as described in Section 5.4.

5.4 Computational results

The following theorem shows that there exists an approximation algorithm with approximation guarantee that is linear in the amount of synergy between the candidates, as measured by the joint supermodular degree of the preference functions of the voters. For submodular set functions, the result described by the theorem coincides with the optimal result for submodular set functions of Fisher, Nemhauser and Wolsey [48] that is used by Skowron, Faliszewski and Lang [83].

Theorem 5.1. *When the joint supermodular degree of the preferences functions of the voters is d, the k-committee selection problem admits an approximation algorithm with guarantee $(1 - e^{-1/(d+1)}) \geq 1/(d+2)$. The algorithm gets access to the preference functions by value queries and supermodular queries, and its running time is* $\texttt{Poly}(n, m, 2^d)$.

Note that the above result captures the example of movies selection from the introduction (see Section 5.3 for further discussion). Note also that the proof of the above result applies to the case of committee selection subject to a *general* matroid constraint (cardinality constraint is a special case of a matroid constraint), but with an approximation guarantee of $1/(d+2)$, by using the respective algorithm of Section 3.

Moreover, one can use the algorithms of Section 4, in order to get an online (secretary like) version of Theorem 5.1. In this online model, the candidates are arriving one by one in a random order, and one should decide on the spot, irrevocably, whether to hire a candidate, based on his contribution to the team hired thus far, and on some information about other candidates he has synergy with (see Section 4 for more details). As an example, consider hiring a team to a project, where each of the candidates meets with a few interviewers. Then, an optimal team of candidates should be hired, according to the preferences of the interviewers (which are the voters).

By using the algorithm of Section 4 for a cardinality constraint, one gets an approximation guarantee polynomial in the joint supermodular degree. Any approximation guarantee that depends only on the joint supermodular degree gives a constant approximation guarantee, if the candidates

admit synergy only with a constant number of other candidates (e.g. if there is a positive correlation only within series of movies, and all the series suggested are of length up to 3). See also Oren and Lucier [79] for a different secretary like model.

Additionally, we show a hardness result for the case of non-bounded joint supermodular degree, even when the supermodular degree of all the set functions is bounded by 1. For this, we use a reduction from the k-dense subgraph problem (see e.g. Bhaskara el al. [10]).

Definition 5.3. *The k-dense subgraph problem is the following. We are given as input a graph $G = (V, E)$ and an integer $k \in \mathbb{N}$, and our aim is to select k vertices such that the number of edges in their induced subgraph is maximized.*

This problem is NP-hard and it is highly believed it is hard to approximate it within any constant guarantee. Actually, no efficient algorithm is currently known that approximates it within a guarantee better than n^c, for some constant c (see e.g. [10, 81, 82]).

Theorem 5.2. *The k-commitee selection problem is at least as hard as the k-dense subgraph problem, even if the supermodular degree of the set functions is 1, and even if an explicit representation of the preference functions is given. This means, in particular, that it is $\mathcal{NP}$-hard*[15] *and $\mathcal{SSE}$-hard (see [81] and also [82]).*

Proof of Theorem 5.1. Let V be the set of n voters, let C be the set of m candidates, let k be the requested number of elected candidates and let $f_1, \ldots, f_n : 2^C \to \mathbb{R}^+$ be the preference functions of the voters. We prove that since the joint supermodular degree of $f_1, \ldots, f_n$ is upper bounded by d, then the supermodular degree of their summation function $f_\Sigma(S) \stackrel{\text{def}}{=} \sum_{i=1}^{n} f_i(S)$ is upper bounded by d, as well. Note that this would not be necessarily true if only the supermodular degree of $f_1, \ldots, f_n$ was bounded by d (or even by 1). Actually, Theorem 5.2 serves as a counter example to the latter for $d = 1$.

To prove the bound on the supermodular degree of the summation function f_Σ, we show that every supermodular dependency by f_Σ induces the same supermodular dependency by one of the f_is in the sum. Let $c, c' \in C$ and $S \subseteq C$ be such that $f_\Sigma(c \mid S \cup \{c'\}) > f_\Sigma(c \mid S)$. Then, by the definition of f_Σ, $\sum_{i=1}^{n} f_i(c \mid S \cup \{c'\}) > \sum_{i=1}^{n} f_i(c \mid S)$. So, $\exists_{1 \le i \le n}$ s.t. $f_i(c \mid S \cup \{c'\}) > f_i(c \mid S)$,

[15] NP-hardness is actually true also for submodular set functions, i.e. supermodular degree of 0.

as claimed.

Now, we can just use the algorithm of [41] for monotone function maximization subject to a uniform matroid constraint (i.e. cardinality constraint) on the function f_Σ with a constraint k. Note that the latter algorithm gives an optimal approximation guarantee for submodular set functions, and generally its guarantee deteriorates linearly with the supermodular degree, as promised by Theorem 5.1. Moreover, its running time is as promised by the Theorem. This concludes the proof of Theorem 5.1. □

Proof of Theorem 5.2. The proof is somewhat similar to the proof of $\mathcal{SSE}$-hardness for maximizing set function subject to cardinality constraint, given by [41]. Given an algorithm for solving the k-commitee selection problem within approximation guarantee α, we show how to solve any input instance of the k-dense subgraph problem within approximation guarantee α. Let $G = (S, E)$ be an instance of the k-dense graph problem. Then, our set of candidates C will be S (the set of vertices of G). We also introduce a voter v_e for every edge $e = \{v_{e1}, v_{e2}\} \in E$ and let $V = \bigcup_{e \in E} \{v_e\}$. For every voter v_e, her preference set function is:

$$f_e = \begin{cases} 1 & \text{if } v_{e_1} \text{ and } v_{e_2} \text{ are both elected.} \\ 0 & \text{otherwise} \end{cases}$$

That is, in this instance of the k-committee selection problem, our aim is to find a subset of k candidates (where the set of candidates corresponds exactly to the set S of vertices of G), such that the number of pairs of candidates, that correspond to the preference functions of the voters, is maximized (where these pairs of candidates are exactly the edges E of G). This is exactly the k-dense subgraph problem. That is, given a solution to this instance of k-committee selection problem, we just output the subset of vertices of S that corresponds to the candidates in C that were selected, as a solution to the input instance of the k-dense subgraph problem. This gives us a feasible solution with the same value, and thus with the same approximation guarantee α. This concludes the proof of Theorem 5.2. □

6 Non-Monotone Valuation Functions: Beyond Submodularity

This section is based on a paper with Uriel Feige [37].

Let M be a ground set. The (unconstrained) FUNCTION MAXIMIZATION problem is to find a subset that maximizes the value of an input non-negative set function $f : 2^M \to \mathbb{R}^+$. This problem was studied extensively for submodular set functions in the value oracle model, and an optimal $\frac{1}{2}$-approximation algorithm was designed by Buchbinder, Feldman, Naor and Schwartz (SICOMP, 2015).

In this section, we explore the FUNCTION MAXIMIZATION problem beyond the submodular regime. Specifically, we characterize the local optimality behaviour of non-negative set functions with respect to their supermodular degree (defined in Section 3). We show that unlike the case of submodular set functions, the value of a local optimum subset might be smaller than the value of a subset that is either contained in it or contains it. Moreover, we show that the multiplicative gap between their values might be super-linear in the supermodular degree, but is upper bounded by some polynomial of the supermodular degree. Additionally, we show that the multiplicative gap between the values of a local optimum and of a global optimum of the function might be arbitrarily large, when the local optimum has a non-trivial intersection with an optimal subset (that is, it is neither contained in it nor contains it).

6.1 Local optimality

Let M be a ground set and let $f : 2^M \to \mathbb{R}^+$ be a set function.

Definition 6.1 (Local optimum)**.** *For every two subsets S, T, let $d(S,T) = |S \setminus T| + |T \setminus S|$ be the distance between S and T (i.e.the items of S and T that are members in exactly one of the two sets).*

We say that a subset $S \subseteq M$ is a single item local optimum of f, if for every $T \subseteq M$ with $d(S,T) \leq 1$, $f(S) \geq f(T)$.

*More generally, for an integer $k \geq 1$, we say that a subset $S \subseteq M$ is a **k-local opt** of f (denoted by k-LO) if for every subset $T \subseteq M$ with $d(S,T) \leq k$, $f(S) \geq f(T)$. Clearly, an $|M|$-local opt of f is a (global) optimum of f (denoted by opt).*

Given a subset S, we sometimes restrict ourselves to subsets T, such that, either $S \subseteq T$ or $S \supseteq T$, by adding the word "monotone" (e.g.monotone 1-LO).

If we would like to relax the condition for local optimality of a subset S, such that $f(S) \geq \rho \cdot f(T)$ for some $\rho \leq 1$, we denote it by ρ-LO. If we use it together with k, we denote it by (ρ, k)-LO (for $\rho = 1$ we will not use it without k).

For submodular set functions, it is known that any single item local optimum (1-LO) is a monotone (global) optimum and also a $\frac{1}{3}$-opt (see [38]). For general set functions, this is far from being the case, as the following states.

Observation 6.1. *Let $t \in \mathbb{N}$. There exists a ground set M, a set function $f : 2^M \to \mathbb{R}^+$, a single item local optimal subset $S \subseteq M$, and a subset $S^+ \supseteq S$, such that $f(S^+) > t \cdot f(S)$. Moreover, S^+ contains only 2 items more than S, regardless of t.*

Proof. Let $M = \{j_1, j_2\}$ and let $f : 2^M \to \mathbb{R}^+$ be the following set function:

$$f(X) = \begin{cases} 1 & \text{if } X = M. \\ 0 & \text{otherwise} \end{cases}$$

Clearly $S = \varnothing$ is a local optimum of value 0, which completes the proof. □

Note that given an integer k, we can set $M = \{j_1, \ldots, j_{k+1}\}$, and then the same result would apply to k-LO. Moreover, f will have a supermodular degree of exactly k, since if $M = \{j_1, j_2, j_{k+1}\}$, then for all $i, i' \in [k+1]$, $i \neq i'$, we have that $f(j_i \mid X \setminus \{j_i\}) > f(j_i \mid X \setminus \{j_i, j_{i'}\})$, and generally $f(j_i \mid Z) = 0$ for any subset $Z \neq M \setminus \{j_i\}$.

That is, for a general set function f, one cannot expect to have positive results for k-LO, unless $k \geq \mathcal{D}_f^+ + 1$. Interestingly, $(\mathcal{D}_f^+ + 1)$-LO is indeed meaningful for monotone (global) optimality, and furthermore, a weaker requirement that is naturally based on the supermodular degree suffices.

Specifically, we suggest the following definition for local optimality:

Definition 6.2 (Supermodular Local Optimum)**.** *We say that a subset $S \subseteq M$ is a supermodular dependencies local optimal subset of f (or just supermodular local optimal subset / supermodular local optimum), if for every $j \in M$ and for every $D^+ \subseteq \mathcal{D}^+(j)$, $f(S) \geq f(S \cup \{j\} \cup D^+)$ and $f(S) \geq f(S \setminus (\{j\} \cup D^+))$. We denote a supermodular local optimum by* SMD-LO*.*

Clearly, every subset that is SMD-LO is also 1-LO, and every subset that is $(\mathcal{D}_f^+ + 1)$-LO is also SMD-LO.

We show the following results for monotone SMD-LO subsets. We start with an upper bound.

Theorem 6.2. *Let M be a ground set and let $f : 2^M \to \mathbb{R}^+$ be a set function. Let $S \subseteq M$ be a supermodular local optimal subset of f. Then, for every subset $S^- \subseteq S$, $f(S) \geq O\left(\frac{1}{\mathcal{D}_f^{+3}}\right) \cdot f(S^-)$, and for every subset $S^+ \supseteq S$, $f(S) \geq O\left(\frac{1}{\mathcal{D}_f^{+3}}\right) \cdot f(S^+)$.*

On the flip side, we show the following.

Theorem 6.3. *There exists a ground set M, a set function $f : 2^M \to \mathbb{R}^+$, a supermodular local optimal subset $S \subseteq M$, and a subset $S^+ \supseteq S$, such that $f(S^+) \geq \Omega\left(\mathcal{D}_f^{+2} / \log \mathcal{D}_f^+\right) \cdot f(S)$. Additionally, there exists a ground set M, a set function $f : 2^M \to \mathbb{R}^+$, a supermodular local optimal subset $S \subseteq M$, and a subset $S^- \subseteq S$, such that $f(S^-) \geq \Omega\left(\mathcal{D}_f^{+2} / \log \mathcal{D}_f^+\right) \cdot f(S)$.*

When, a subset has a non-trivial intersection with a supermodular local optimal subset, we show that its value can be larger by any factor, independently of the supermodular degree of the function. Specifically, we show the following.

Proposition 6.4. *Let $t \in \mathbb{N}$. There exists a ground set M, a set function $f : 2^M \to \mathbb{R}^+$ of supermodular degree 1, a supermodular local optimal subset $S \subseteq M$, and a subset $T \subseteq M$, such that $f(T) \geq t \cdot f(S)$. Moreover, for every integer k, there exists a set function $f : 2^M \to \mathbb{R}^+$ of supermodular degree 1 with a k-LO subset that has an $\Omega(n/k)$ multiplicative gap from* opt*.*

6.2 Proofs of results

We start with proving the following Lemma:

Lemma 6.5. *Let $f : 2^M \to \mathbb{R}^+$ be a set function with supermodular degree d and let $f' : 2^M \to \mathbb{R}^+$*

be $f'(S) = f(M \setminus S)$. *Then* f' *has supermodular degree* d, *as well.*

In order to prove Lemma 6.5, we first prove the following claim:

Claim 6.6. *Let* $f : 2^M \to \mathbb{R}^+$ *be a set function with supermodular degree* d *and let* $f' : 2^M \to \mathbb{R}^+$ *be* $f'(S) = f(M \setminus S)$. *Then, for every item* $j \in M$ *and for every subset* $S \subseteq M$, *we have,*

$$f'(j \mid M \setminus (S \cup \{j\}) = -f(j \mid S) \ .$$

Proof of Claim 6.6:

$$f'(j \mid M \setminus (S \cup \{j\}) = f'(j \cup (M \setminus (S \cup \{j\})) - f'(M \setminus (S \cup \{j\}) = f(S) - f(S \cup \{j\}) = -f(j \mid S) \ ,$$

where, the first and third equalities follow by the definition of marginal set function, and the second equality follows by the definition of f' with respect to f. □

We now use Claim 6.6, in order to prove Lemma 6.5.

Proof of Lemma 6.5: Let $j_1, j_2 \in M$. We show that if j_1 and j_2 are supermodular dependent by f, then they are also supermodular dependent by f'. Since it is true that $f(M \setminus S) = f'(S)$, as well, it will show that j_1 and j_2 are supermodular dependent by f if and only if they are supermodular dependent by f'. Let $S \subseteq M$ be a subset such that

$$f(j_1 \mid S \cup \{j_2\}) > f(j_1 \mid S) \ . \tag{6}$$

Such a subset exists by the definition of supermodular dependency. By Claim 6.6, we have that $f(j_1 \mid S) = -f'(j_1 \mid (M \setminus S \cup \{j_1\}))$, and $f(j_1 \mid S \cup \{j_2\}) = -f'(j_1 \mid M \setminus (S \cup \{j_2, j_1\})$. Then, by (6), we have that

$$-f'(j_1 \mid M \setminus (S \cup \{j_2, j_1\}) > -f'(j_1 \mid M \setminus (S \cup \{j_1\})) \ .$$

So, $f'(j_1 \mid M \setminus (S \cup \{j_1\})) > f'(j_1 \mid M \setminus (S \cup \{j_2, j_1\}))$. This means that j_1 and j_2 are supermodular dependent for f' (because of the subset $M \setminus (S \cup \{j_2, j_1\})$). We conclude the proof of Lemma 6.5. □

Note that for any two subsets $A \subseteq B \subseteq M$, we have that $M \setminus A \supseteq M \setminus B$. Therefore, by Lemma 6.5, we can prove Theorems 6.2 and 6.3 only for one side of containment, and the other will follow.

Lemma 6.7. *Let $f : M \to \mathbb{R}^+$. Let A be a supermodular local optimal subset. Then, for every $A' \subseteq M$ such that $A \subseteq A'$, $f(A') \leq O(\mathcal{D}_f^{+3}) \cdot f(A)$.*

Proof. Let $B = A' \setminus A$. Clearly $A \cup B = A'$. We show that $f(A \cup B) \leq O(\mathcal{D}_f^{+3}) \cdot f(A)$. First, note that $f(A \cup B) = f(B \mid A) + f(A)$. Therefore, it is sufficient to upper bound $f(B \mid A)$. Let $b = |B|$ and let $B = \{j_1, \ldots, j_b\}$. Then,

$$f(B \mid A) = \sum_{j_i \in B} f(j_i \mid \{j_{i+1} \ldots, j_b\} \cup A) \leq \sum_{j_i \in B} f(j_i \mid Dep^+(j_i)^{>i} \cup A) \ , \tag{7}$$

where, $Dep^+(j_i)^{>i} \stackrel{\text{def}}{=} Dep^+(j_i) \cap \{j_{i+1}, \ldots j_b\}$, and the inequality follows by the definition of supermodular dependencies. By supermodular local optimality of A, for every $j \in B$ and $S \subseteq Dep^+(j) \setminus A$, $f(\{j\} \cup S \mid A) \leq 0$. Therefore, for every $j_i \in B$, $f(\{j_i\} \cup Dep^+(j_i)^{>i} \mid A) \leq 0$, and then,

$$\sum_{j_i \in B} f(\{j_i\} \cup Dep^+(j_i)^{>i} \mid A) \leq 0 \ . \tag{8}$$

Therefore,

$$\begin{aligned} f(B \mid A) &\leq f(B \mid A) - \sum_{j_i \in B} f(\{j_i\} \cup Dep^+(j_i)^{>i} \mid A) \\ &\leq \sum_{j_i \in B} f(j_i \mid Dep^+(j_i)^{>i} \cup A) - \sum_{j_i \in B} \left(f(j_i \mid Dep^+(j_i)^{>i} \cup A) + f(Dep^+(j_i)^{>i} \mid A) \right) \\ &= \sum_{j_i \in B} \left(-f(Dep^+(j_i)^{>i} \mid A) \right) \ , \end{aligned}$$

where the first inequality follows by (8) and the second by (7), together with the definition of a marginal set function. We proceed by bounding $\sum_{j_i \in B} \left(-f(Dep^+(j_i)^{>i} \mid A) \right)$. Note that by non-negativity, for every $j_i \in B$, $f(Dep^+(j_i)^{>i} \cup A) = f(Dep^+(j_i)^{>i} \mid A) + f(A) \geq 0$, which implies

$-f(Dep^+(j_i)^{>i} \mid A) \leq f(A)$. That is, we already have a bound of

$$\sum_{j_i \in B} \left(-f(Dep^+(j_i)^{>i} \mid A)\right) \leq |B| \cdot f(A) \ .$$

We show how to derive a better upper bound. By non-negativity, we have that

$$f\left(\bigcup_{j_i \in B} Dep^+(j_i)^{>i} \cup A\right) = f\left(\bigcup_{j_i \in B} Dep^+(j_i)^{>i} \mid A\right) + f(A) \geq 0 \ ,$$

which implies $-f(\bigcup_{j_i \in B} Dep^+(j_i)^{>i} \mid A) \leq f(A)$. Therefore, it is sufficient to prove that

$$\sum_{j_i \in B} \left(-f(Dep^+(j_i)^{>i} \mid A)\right) \leq -O\left(\mathcal{D}_f^{+3}\right) \cdot f\left(\bigcup_{j_i \in B} Dep^+(j_i)^{>i} \mid A\right) \ . \tag{9}$$

Note that if f had been submodular, then (9) would have been correct, even without the $O(\mathcal{D}_f^{+3})$ factor, since,

$$\begin{aligned} f\left(\bigcup_{j_i \in B} Dep^+(j_i)^{>i} \middle| A\right) &= \sum_{i=1}^{b} f\left(Dep^+(j_i)^{>i} \middle| A \cup \bigcup_{\ell=i+1}^{b} Dep^+(j_\ell)^{>\ell}\right) \\ &\leq \sum_{i=1}^{b} f\left(Dep^+(j_i)^{>i} \middle| A\right) \\ &= \sum_{j_i \in B} f\left(Dep^+(j_i)^{>i} \middle| A\right) \end{aligned} \tag{10}$$

We denote $Dep^+(j_1)^{>1}, \ldots, Dep^+(j_b)^{>b}$ by $D_1^*, \ldots, D_b^*$, respectively. We show how to iteratively choose $\Omega(\mathcal{D}_f^{+3})$ of the subsets $D_1^*, \ldots, D_b^*$, such that:

- There are no overlaps and no supermodular dependencies between chosen subsets.
- At every iteration, we choose one subset D_i^* and decide to not consider for future iterations (*i.e.*"drop") up to $O(\mathcal{D}_f^{+3})$ subsets $\hat{D}_1^*, \ldots, \hat{D}_k^*$, where $\forall_{\ell \in [k]} : -f(\hat{D}_\ell^* \mid A) \leq -f(D_i^* \mid A)$. The next iteration will run without the subset chosen and without the subsets "dropped".

Note that the first condition is sufficient in order (10) will be correct (the single inequality that is

dependent on submodularity will be correct). Therefore, if we show an iterative process obeying both conditions, it will prove (9), and therefore the Lemma.

Our iterative process is greedy. At every iteration we choose a subset D_i^* with a maximal $-f(D_i^* \mid A)$, and "drop" every subset D_j^* such that either $D_i^* \cap D_j^* \neq \varnothing$ or there exists items $j_1' \in D_i^*, j_2' \in D_j^*$, such that j_1' and j_2' are supermodularly dependent (that is one of them is in the supermodular dependency subset of the other. Note that the supermodular dependency relation is symmetric. See Lemma 3.21).

It now remains to show that for every item chosen, we "drop" up to $O(\mathcal{D}_f^{+3})$ others. Let $d = \mathcal{D}_f^{+3}$. By definition of the supermodular degree, for every $j_i \in B$, $|Dep^+(j_i)^{>i}| \leq d$. Additionally, each of the items $j \in Dep^+(j_i)^{>i}$ has at most d supermodular dependencies. Each of the $O(d^2)$ items described thus far might be the supermodular dependency of at most d items (by symmetry of supermodular dependency relation, see Section 3.9), and therefore belong together to at most $O(d^3)$ different $Dep^+(j_i)^{>i}$s. This concludes the proof of Lemma 6.7. □

Proof of Theorem 6.2. Let M be a ground set and let $f : 2^M \to \mathbb{R}^+$ be a set function. Let $S \subseteq M$ be a supermodular local optimal subset of f. By Lemma 6.7, for every subset $S^+ \supseteq S$,

$$f(S) \geq \Omega\left(\frac{1}{\mathcal{D}_f^{+3}}\right) \cdot f(S^+) \ . \tag{11}$$

Let $S^- \subseteq S$. By Lemma 6.5, the non-negative set function $f' : 2^M \to \mathbb{R}^+$, defined as $f'(S) = f(M \setminus S)$, has a supermodular degree of $\mathcal{D}_f^+$. Moreover, $M \setminus S \subseteq M \setminus S^-$. Therefore, By Lemma 6.7, we have that

$$f(S) = f'(M \setminus S) \geq \Omega\left(\frac{1}{\mathcal{D}_f^{+3}}\right) \cdot f'(M \setminus S^-) = \Omega\left(\frac{1}{\mathcal{D}_f^{+3}}\right) \cdot f(S^-) \ . \tag{12}$$

Theorem 6.2 follows by (11) and (12). □

Proof of Theorem 6.3. We show how to build a hypergraph representation of a function f, as promised by Theorem 6.3, for the case of a local optimal subset $S \subseteq M$ and a subset $S^+ \supseteq S$. This representation will actually be a graphical representation (*i.e.*we will use non-zero weighted

hyperedges only for hyperedges of ranks up to 2). Note that by letting f' be a set function with $f'(X) = f(M \setminus X)$, for every subset X of the ground set of f, and by applying Lemma 6.5, we will get immediately a function for the case of $S^- \subseteq S$, as well.

We let the ground set be $M \stackrel{\text{def}}{=} \{j, j_1, \ldots, j_n\}$ with $m = n + 1$ items. We let $f : 2^M \to \mathbb{R}^+$ be defined by the following hypergraph $G = (M, E, w)$. The vertices $j_1, \ldots, j_n$ will form a d-regular expander. The weights of the edges of the expander will be 1, and of the vertices 0. Additionally there will be edges of weight (-1) for every pair $\{j, j_i\}$, for $i \in [n]$. The weight of the vertex j will be the size of the largest independent set in the expander (which is $\Theta(n\frac{\log d}{d})$ for some expanders). It is easy to see that f is a non-negative set function with a supermodular degree of d. Additionally, $\{j\}$ is a supermodular local optimum. Finally, adding the items of the expander to this supermodular local optimum gives a value of $(n \cdot d)/2 - n = \Theta(n \cdot d)$. Thus the ratio between $f(M)$ and $f(\{j\}$ is indeed $\Omega(d^2/\log d)$. Finally, of course, $\{j\} \subseteq M$, so we can let $S \stackrel{\text{def}}{=} \{j\}$ and $S^+ \stackrel{\text{def}}{=} M$. We conclude the proof of Theorem 6.3. □

Proof of Proposition 6.4. Figure 5 is a hypergraph representation of a function serving as an example proving the first part of Proposition 6.4. Specifically, $\{j, j_1\}$ is a supermodular local optimal subset of value $\epsilon + \epsilon/2$, but $\{j_1, j_1'\}$ has value of $1 + \epsilon/2$. Choosing ϵ such that $1 + \epsilon/2 > t \cdot (\epsilon + \epsilon/2)$ completes the proof.

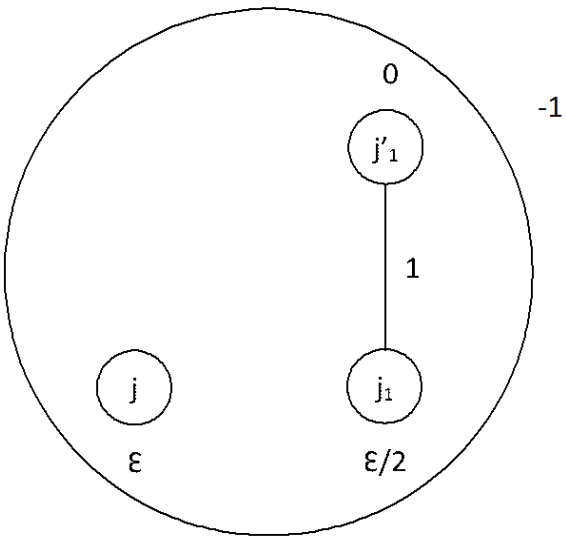


Figure 5: A hypergraph representation of a function with an unbounded local optimum.

Note that the function described by Figure 5 shows that a local search algorithm that is looking for a *supermodular local optimum* (by allowing the algorithm to not only add or remove single items, but also their supermodular dependencies) will have an *unbounded* approximation guarantee.

Looking at the last example, one may consider more sophisticated local search algorithms. For example, if we allow replacement of items (*i.e.*., not only either addition or removal of items, but both), an optimal solution can be actually reached in the example depicted in Figure 5. However, this is not the case for the set function described by Figure 6. Another sophistication one can think of is using enhancements of the current solution *greedily*. That is, to not just add/remove/replace items and their supermodular dependencies in *any* possible way that strictly increases the value of the current solution, but to *greedily* choose an option that gains the maximal possible increase. However, Figure 6 is resistant also to such an algorithm – the supermodular local optimum that is reached has a value that is smaller than the value of a global optimum by any multiplicative factor. Interestingly, even combining the latter enhancements of the local search, by allowing both greedy choice and replacement, will result in an unbounded local optimum.

In Figure 6 below, we have a hypergraph representation of a set function of supermodular degree 1. Given the empty set, the best local improvement one has is to add j. The resulting subset is $\{j\}$, which is a supermodular local optimum of value $1 + \epsilon$. However, the subset $\{j_1, j'_1, j_2, j'_2, \ldots, j_k, j'_k\}$ has value of more than k.

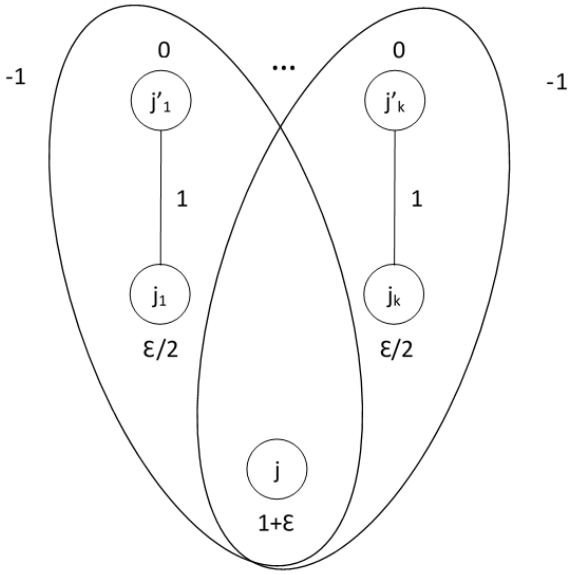


Figure 6: A hypergraph representation of a function with an unbounded supermodular local optimum, even when replacement of items and their supermodular dependencies (as in Definition 6.2) is allowed.

To get the result for ℓ-LO, one can just let the value of j be $\ell + \epsilon$. □

7 Welfare maximization and Maximum over Positive Hypergraphs

This section is based on a paper with Uriel Feige, Michal Feldman, Nicole Immorlica, Brendan Lucier and Vasilis Syrgkanis [33].

We introduce a new hierarchy of monotone set functions called *maximum over positive hypergraphs* ($\mathcal{MPH}$), whose level captures the degree of complementarity. A new hierarchy is useful if it has a strong expressiveness power on the one hand, and algorithmic implications on the other. We show that important classes of functions are captured in low levels of our hierarchy. We then present algorithmic results that illustrate the usefulness of our hierarchy. In particular, we develop an algorithm that approximates the welfare maximization problem to within a factor of $k+1$, where k is the degree of complementarity of the valuations, as captured by our hierarchy.

7.1 Some of our results

We obtain results on the expressiveness power of the $\mathcal{MPH}$ hierarchy and show applications of it for approximating social welfare in combinatorial auctions.

Expressiveness

Theorem 7.1. *The $\mathcal{MPH}$ hierarchy captures many existing hierarchies, as follows:*

1. *By definition, $\mathcal{MPH}$-1 is equivalent to the class $\mathcal{XOS}$ (defined by Lehmann, Lehmann and Nisan [73]) and every function that has a positive hypergraph representation of rank k (defined by Abraham et al. [2]) is in $\mathcal{MPH}$-k.*

2. *Every monotone graphical valuation (defined by Conitzer et al. [22]) is in $\mathcal{MPH}$-2. Furthermore, every monotone function with positive rank 2 is $\mathcal{MPH}$-2.*

3. *Every monotone function that has a hypergraph representation with positive rank k and laminar negative hyperedges (with arbitrary rank) is in $\mathcal{MPH}$-k.*

4. Every monotone function that has supermodular degree k (see Section 3) is in $\mathcal{MPH}$-$(k+1)$.

Applications to welfare maximization

Theorem 7.2. *If all players have $\mathcal{MPH}$-k valuations, then there exists an algorithm that gives $k+1$ approximation to the optimal social welfare. This algorithm runs in polynomial time given an access to demand oracles for the valuations.*

A hierarchy for non-monotone set functions Most of our expressiveness results showing that a certain function belongs to $\mathcal{MPH}$-k are established by showing that the function satisfies a certain requirement that we refer to as the Positive Lower Envelope (PLE) condition. We also observe that, together with monotonicity, this requirement becomes a sufficient and necessary condition for membership in $\mathcal{MPH}$-k. This observation motivates the definition of a new hierarchy, referred to as $\mathcal{PLE}$. The class $\mathcal{PLE}$-k contains $\mathcal{MPH}$-k, but also includes non-monotone functions. While monotonicity is a standard assumption in the context of combinatorial auctions, $\mathcal{PLE}$ can be applicable outside the scope of combinatorial auctions. We show that any set function with positive rank 1 is in $\mathcal{PLE}$-1. On the other hand, we show that there exists a set function with rank 2 that is not in $\mathcal{PLE}$-k for every k. Note that this is not the case for functions that are symmetric or laminar.

Extensions One of the main open problems suggested by this work is the relation between hypergraph valuations of rank k and $\mathcal{MPH}$-k valuations. We make the following conjecture:

Conjecture 7.3. *Every hypergraph function with rank k (positive or negative) is in $\mathcal{MPH}$-$O(k^2)$.*

We make partial progress toward the proof of this conjecture, by confirming it for the case of symmetric functions. For non-symmetric, observe that for the case of laminar negative hyperedges, we show an even stronger statement in item (3) of Theorem 7.1.

Theorem 7.4. *Every monotone symmetric hypergraph function with rank k (positive or negative) is in $\mathcal{MPH}$-$O(k^2)$.*

For symmetric functions, we conjecture a more precise bound of $\lceil \frac{k}{2} \rceil \lceil \frac{k+1}{2} \rceil$, suggested by a computer-aided simulation based on a non-trivial LP formulation. For the special cases of symmetric

functions of ranks $k = 3$ and 4, we show that they are in $\mathcal{MPH}$-4 and $\mathcal{MPH}$-6, respectively, and that this is tight. We use an LP formulation whose optimal solution is the worst symmetric function possible for a given rank, and its value corresponds to the level of this worst function in the $\mathcal{MPH}$ hierarchy. We bound the value of this LP, by using LP duality.

7.2 The $\mathcal{MPH}$ hierarchy

Recall that a hypergraph representation of a set function $v : 2^M \to \mathbb{R}^+$ is a (normalized but not necessarily monotone) set function $h : 2^M \to \mathbb{R}$ that satisfies $v(S) = \sum_{T \subseteq S} h(T)$. It is easy to verify that any set function v admits a unique hypergraph representation and vice versa. A set S such that $h(S) \neq 0$ is said to be a *hyperedge* of h. Pictorially, the hypergraph representation can be thought of as a weighted hypergraph, where every vertex is associated with an item in M, and the weight of each hyperedge $e \subseteq M$ is $h(e)$. Then the value of the function for any set $S \subseteq M$, is the total value of all hyperedges that are contained in S.

The *rank* of a hypergraph representation h is the largest cardinality of any hyperedge. Similarly, the *positive rank* (respectively, *negative rank*) of h is the largest cardinality of any hyperedge with strictly positive (respectively, negative) value. The rank of a set function v is the rank of its corresponding hypergraph representation, and we refer to a function v with rank r as a *hypergraph-r* function. Last, if the hypegraph representation is non-negative, i.e. for any $S \subseteq M$, $h(S) \geq 0$, then we refer to such a function as a *positive hypergraph-r* ($\mathcal{PH}$-r) function .

We define a parameterized hierarchy of set functions, with a parameter that corresponds to the degree of complementarity.

Definition 7.1 (Maximum Over Positive Hypergraph-k ($\mathcal{MPH}$-k) class)**.** *A monotone set function $v : 2^M \to \mathbb{R}^+$ is* Maximum over Positive Hypergraph-k *($\mathcal{MPH}$-k) if it can be expressed as a maximum over a set of $\mathcal{PH}$-k functions. That is, there exist $\mathcal{PH}$-k functions $\{v_\ell\}_{\ell \in \mathcal{L}}$ such that for every set $S \subseteq M$,*

$$v(S) = \max_{\ell \in \mathcal{L}} v_\ell(S), \tag{13}$$

where $\mathcal{L}$ is an arbitrary index set.

7.3 Positive Lower Envelopes

Proving that a particular set function $f : 2^M \to \mathbb{R}^+$ can be expressed as $\mathcal{MPH}$-k requires constructing a set of $\mathcal{PH}$-k valuations that constitutes the index set $\mathcal{L}$ over which the maximum is taken. In what follows we present a canonical way of constructing the set $\mathcal{L}$. The idea is to create a $\mathcal{PH}$-k function for every subset S of the ground set M. The collection of these $\mathcal{PH}$-k functions, one for each subset, constitutes a valid $\mathcal{MPH}$-k representation if they adhere to what we define as Positive Lower Envelopes.

Definition 7.2 (Positive Lower Envelope (PLE))**.** *Let $f : 2^M \to \mathbb{R}^+$ be a monotone set function. A positive lower envelope (PLE) of f is any positive hypergraph function g such that:*

- $g(M) = f(M)$.
- *For any $S \subseteq M$, $g(S) \leq f(S)$. [No overestimate]*

Before presenting the characterization, we need the following definition. A function $f : 2^M \to \mathbb{R}^+$ *restricted to a subset* S, $S \subseteq M$, is a function $f_S : 2^S \subseteq \mathbb{R}^+$ with $f_S(S') = f(S')$ for every $S' \subseteq S$. We show that a monotone set function is in $\mathcal{MPH}$-k if and only if f_S admits a PLE of rank k for every set $S \subseteq M$. This characterization follows directly from the following two propositions.

Proposition 7.5. *Every monotone set function f such that f_S admits a lower envelope of rank k for every set $S \subseteq M$ is in $\mathcal{MPH}$-k.*

Proposition 7.6. *Every function f that is in $\mathcal{MPH}$-k is monotone. Moreover, For every set $S \subseteq M$, f_S admits a positive lower envelope of rank k.*

Proof. **First direction: Monotone, PLE of rank $k \Rightarrow \mathcal{MPH}$-$k$:** Let $f : 2^M \to \mathbb{R}^+$ be a monotone set function. For any $T \subseteq M$, let g_T be a positive lower envelope of rank k of f restricted to T. We argue that $\{g_T\}_{T \subseteq M}$ is an $\mathcal{MPH}$-k representation of f. Specifically, we show that for every $S \subseteq M$, it holds that $\max_{T \subseteq M} g_T(S) = f(S)$. Let $S \subseteq M$. By the first property of Definition 7.2, it holds that $g_S(S) = f(S)$. Therefore,

$$\max_{T \subseteq M} g_T(S) \geq g_S(S) = f(S) \tag{14}$$

Additionally, for any $T \subseteq M$, $g_T(S) = g_T(S \cap T) \leq f(S \cap T) \leq f(S)$, where the equality follows from the fact that g_T is restricted to T; the first inequality follows from the no-overestimate property of Definition 7.2 and the last inequality follows from monotonicity of f. Therefore,

$$\max_{T \subseteq M} g_T(S) \leq f(S) \tag{15}$$

The first direction follows by Equations (14) and (15).

Second direction: $\mathcal{MPH}$-k $\Rightarrow$ Monotone, PLE of rank k: Let $f : 2^M \to \mathbb{R}^+$ be a function in $\mathcal{MPH}$-k and let $\mathcal{L}$ be an $\mathcal{MPH}$ representation of it. We first prove that f is monotone. Assume towards contradiction that f is not monotone. Then, there exist $S' \subset S \subseteq M$ such that $f(S') > f(S)$. Since $\mathcal{L}$ is an $\mathcal{MPH}$ representation of f, there exists a positive hypergraph function $f_h \in \mathcal{L}$ such that $f_h(S') = f(S')$. This means that $f_h(S) \geq f(S') > f(S)$, which implies that $\max_{g \in \mathcal{L}} g(S) > f(S)$, deriving a contradiction. The monotonicity of f follows. We next show that for every set $S \subseteq M$, f_S admits a positive lower envelope of rank k. Let $S \subseteq M$. There exists a positive hypergraph function $f_h \in \mathcal{L}$ such that $f_h(S) = f(S)$. Moreover, no set $S' \subseteq S$ can have value strictly greater than $f(S')$ according to f_h, since if it does, this will be a lower bound on the value of S' according to $\mathcal{L}$. Therefore f_h is a positive lower envelope of f_S, as desired. The second direction follows.

□

7.4 Algorithmic result

In this section we consider the purely algorithmic problem, ignoring incentive constraints. While constant factor approximations exist for welfare maximization in the absence of complementarities (see [26, 34]), it is not hard to see that complementarities can make the welfare problem as hard as independent set and hence inapproximable to within an almost linear factor. Our hierarchy offers a linear degradation of the approximation as a function of the degree of complementarity. At a high level, our algorithm works as follows: define the *configuration linear program* (LP) (introduced in [26]) by introducing a variable $x_{i,S}$ for every agent i and subset of items S. Given the valuation

function v_i of each agent i, the *configuration LP* is:

$$\begin{aligned} \text{maximize} \quad & \sum_{i,S} x_{i,S} \cdot v_i(S) \\ \text{s.t.} \quad & \sum_{S} x_{i,S} \leq 1 \quad \forall i \in N \\ & \sum_{i,S|j \in S} x_{i,S} \leq 1 \quad \forall j \in M \text{ and } x_{i,S} \geq 0 \quad \forall i \in N, S \subseteq M \end{aligned} \tag{16}$$

The first set of constraints guarantees that no agent is allocated more than one set and the second set of constraints guarantees that no item belongs to more than one set. This LP provides an upper bound on the optimal welfare. To find a solution that approximates the optimal welfare, we first solve this LP (through duality using *demand queries*) and then round it (see below).

Rounding the LP The rounding proceeds in two steps. First each agent i is assigned a tentative set S_i' according to the probability distribution induced by the variables $x_{i,S}$. Note that this tentative allocation has the same expected welfare as the LP. However, it may be infeasible as agents' sets might overlap. We must resolve these contentions. Several approaches for doing this when there are no complementarities were proposed and analyzed in [26, 34]. However, these approaches will fail badly in our setting, due to the existence of complementarities. Instead, we resolve contention using the following technique: We generate a uniformly random permutation π over the agents and then at each step t for $1 \leq t \leq n$, assign agent $i = \pi(t)$ items $S_i = S_i' \setminus \{\cup_{i'=\pi(1)}^{\pi(t-1)} S_{i'}\}$, i.e., those items in his tentative set that have not already been allocated.

The following proposition shows that this way of contention resolution guarantees a loss of at most a factor of $k+1$, when all agents have $\mathcal{MPH}$-k valuations.

Proposition 7.7. *If all agents have $\mathcal{MPH}$-k valuations, then given a solution to the configuration LP, the above random permutation rounding algorithm produces (in expectation) an allocation that approximates the maximum welfare within a ratio no worse than $k+1$.*

Proof. First, note that the solution is feasible, since every item is allocated at most once. We upper bound the approximation guarantee. The sum of values of tentative sets preserve, in expectation,

the value of the optimal welfare returned by the configuration LP. Consider an arbitrary agent and his tentative set T. This set attained its value according to some positive hypergraph H that has no edges of rank larger than k. Consider an arbitrary edge of H contained in T, and let $r \leq k$ be its rank. We claim that its expected contribution (expectation taken over the random choices of the other agents and the random permutation) towards the final welfare is at least $1/(r+1)$ of its value. The expected number of other agents who compete on items from this edge is at most r (by summing up the fractional values of sets that contain items from this edge). Given that there are ℓ other competing agents, the agent gets all items from the edge with probability exactly $1/(\ell+1)$. As the expectation of ℓ is at most r, the expectation of $1/(\ell+1)$ is at least $1/(r+1)$ (by convexity) and hence at least $1/(k+1)$ as the valuation function is $\mathcal{MPH}$-k. The proof follows from linearity of expectation. □

It is known that there is an integrality gap of $k-1+\frac{1}{k}$ for hypergraph matching in k-uniform hypergraphs (see Chan and Lau [17] and references therein). These instances are special cases of welfare maximization with $\mathcal{MPH}$-k valuations. Hence, our rounding technique in Proposition 7.7 is nearly best possible. For completeness, we show this integrality gap for our setting. Recall also that even for the case of single-minded bidders with sets of size up to k, it is $\mathcal{NP}$-hard to approximate the welfare maximization problem to a better factor than $\Omega(\frac{\ln k}{k})$.[16]

Proposition 7.8. *Let $k \in \mathbb{N}$ be such than $k-1$ is a power of prime. There exists an instance of the welfare maximization problem with $\mathcal{PH}$-k valuations and integrality gap $k-1+\frac{1}{k}$ for the configuration LP. Note that such an instance is in particular $\mathcal{MPH}$-k.*

Proof. Let FPP_{k-1} be the finite projective plane of order $k-1$ (it is known to exist, since $k-1$ is a power of prime). We set the following hypergraph $H=(V,E)$. For each point in FPP_{k-1}, we have a vertex in V, and for each line, we have a hyperedge in E, containing the vertices representing the points that are on this line. The following follows from the definitions of finite projective planes:

- $|V| = |E| = (k-1)^2 + (k-1) + 1 = (k-1)k + 1$.
- Any two hyperedges in E have a vertex in common.

[16] This hardness is obtained by an approximation preserving reduction from k-set packing given in [74], together with a hardness result of [59].

- Any hyperedge in E contains exactly k vertices (i.e. the hyperedges in E are all of rank k).
- Any vertex in V is contained in exactly k hyperedges.

Our instance of the welfare maximization problem has $(k-1)^2 + k$ agents. Each agent i has one distinct preferred hyperedge $e_i \in E$. The valuation function v_i of agent i has value 1 for any subset containing all the items represented by vertices in e_i and 0 otherwise. It is trivial that v_i is in $\mathcal{PH}$-k. Furthermore, the agents are single minded. An optimal *integral* solution of this instance is an allocation of all the items represented by vertices in e_i to agent i, for some arbitrary i. This solution has value of 1. However, there exists a better *fractional* solution. Any agent i gets a fraction of $\frac{1}{k}$ of all the items represented by vertices in e_i. It is easy to verify that this is a feasible fractional solution with value of $((k-1)k+1)/k = k - 1 + \frac{1}{k}$, as desired. □

8 References

[1] Ittai Abraham, Moshe Babaioff, Shaddin Dughmi, and Tim Roughgarden. Combinatorial auctions with restricted complements. In *EC*, pages 3–16, New York, NY, USA, 2012. ACM. 7, 9, 10, 15, 23, 25

[2] Ittai Abraham, Moshe Babaioff, Shaddin Dughmi, and Tim Roughgarden. Combinatorial auctions with restricted complements. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 3–16, New York, NY, USA, 2012. ACM. 116

[3] Pablo D. Azar, Robert Kleinberg, and S. Matthew Weinberg. Prophet inequalities with limited information. In *SODA*, pages 1358–1377, 2014. 56

[4] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In Moses Charikar, Klaus Jansen, Omer Reingold, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Lecture Notes in Computer Science, pages 16–28. Springer Berlin / Heidelberg, 2007. 56, 61

[5] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exchanges*, 7(2):7:1–7:11, Jun 2008. 56

[6] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, pages 434–443, 2007. 56, 61, 69

[7] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Transactions on Algorithms*, 9(4):32, 2013. 56, 61, 62

[8] P. Berman. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. *Nordic Journal of Computing*, 7:178–184, 2000. Preliminary version in SWAT'00. 19

[9] N. Betzler, A. Slinko, and J. Uhlmann. On the computation of fully proportional representation. *Artificial Intelligence Research*, 47:475–519, 2013. 97

[10] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 201–210, 2010. 104

[11] Sushil Bikhchandani and John W. Mamer. Competitive equilibrium in an exchange economy with indivisibilities. *Journal of Economic Theory*, 74(2):385–413, 1997. 26

[12] L. Blumrosen and N. Nisan. On the computational power of demand queries. *SIAM Journal on Computing*, 39:1372–1391, 2009. 11

[13] Liad Blumrosen and Noam Nisan. On the computational power of demand queries. *SIAM Journal on Computing*, 39:1372–1391, 2009. 11, 21

[14] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *FOCS*, pages 649–658, 2012. 26

[15] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011. 13, 25, 26

[16] B. Chamberlin and P. Courant. Representative deliberations and representative decisions: Proportional representation and the borda rule. *American Political Science Review*, 77:718–733, 1983. 97

[17] Lau L. C. Chan. Y. H. On linear and semidefinite programming relaxations for hypergraph matching. *Mathematical Programming*, 135:123–148, 2012. 56, 61, 122

[18] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Multiagent resource allocation in k-additive domains: preference representation and complexity. *Annals of Operations Research*, 163:49–62, 2008. 9, 10, 25

[19] M. Conforti and G. Cornuèjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Disc. Appl. Math.*, 7(3):251–274, 1984. 25

[20] V. Conitzer, T. Sandholm, and P. Santi. Combinatorial auctions with k-wise dependent valuations. In *AAAI*, pages 248–254, 2005. 7, 9

[21] V. Conitzer, T. Sandholm, and P. Santi. Combinatorial auctions with k-wise dependent valuations. In *AAAI*, pages 248–254, 2005. 10, 25

[22] Vincent Conitzer, Tuomas Sandholm, and Paolo Santi. Combinatorial auctions with k-wise dependent valuations. In *In Proc. 20th National Conference on Artificial Intelligence (AAAI-05*, pages 248–254. AAAI Press, 2005. 116

[23] Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *ICML*, pages 1057–1064, 2011. 24

[24] Nedialko B. Dimitrov and C. Greg Plaxton. Competitive weighted matching in transversal matroids. *Algorithmica*, 62(1–2):333–348, 2012. 56, 61

[25] Michael Dinitz and Guy Kortsarz. Matroid secretary for regular and decomposable matroids. *SIAM J. Comput.*, 43(5):1807–1830, 2014. 61

[26] S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. *Mathematics of Operations Research*, 35:1–13, 2010. Preliminary version in STOC'05. 7, 22, 24, 26, 120, 121

[27] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *SODA*, pages 1064–1073, 2006. 7, 24, 26

[28] E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Sov. Math. Dokl.*, 4, 1963. 56

[29] J. Edmonds. Maximum matching and a polyhedron with $0,1$-vertices. *Journal of Research of the National Bureau of Standards*, 69:125–130, 1965. 50

[30] Edith Elkind, Piotr Faliszewski, Piotr Skowron, and Arkadii Slinko. Properties of multiwinner voting rules. In *AAMAS*, pages 53–60, 2014. 97, 100

[31] Yuval Emek, Magnús M. Halldórsson, Yishay Mansour, Boaz Patt-Shamir, Jaikumar Radhakrishnan, and Dror Rawitz. Online set packing. *SIAM Journal on Computing*, 41(4):728–746, 2012. 62

[32] U. Feige. On maximizing welfare when utility functions are subadditive. *SIAM Journal on Computing*, 39:122–142, 2009. Preliminary version in STOC'06. 7, 10

[33] U. Feige, M. Feldman, N. Immorlica, R. Izsak, B. Lucier, and V. Syrgkanis. A unifying hierarchy of valuations with complements and substitutes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 872–878, 2015. 2, 27, 116

[34] Uriel Feige. On maximizing welfare when utility functions are subadditive. In *STOC*, 2006. 120, 121

[35] Uriel Feige. On maximizing welfare when utility functions are subadditive. *SIAM Journal on Computing*, 39:122–142, 2009. Preliminary version in STOC'06. 24, 26

[36] Uriel Feige and Rani Izsak. Welfare maximization and the supermodular degree. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 247–256, New York, NY, USA, 2013. ACM. 2, 8, 27, 99

[37] Uriel Feige and Rani Izsak. Function maximization: Beyond submodularity, 2016. In preparation. 2, 106

[38] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40(4):1133–1153, 2011. 107

[39] Uriel Feige and Jan Vondrák. The submodular welfare problem with demand queries. *Theory of Computing*, 6(1):247–290, 2010. 24, 26

[40] Michal Feldman, Ophir Friedler, Jamie Morgenstern, and Guy Reiner. Simple mechanisms for agents with complements. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 251–267, 2016. 27

[41] Moran Feldman and Rani Izsak. Constrained Monotone Function Maximization and the Supermodular Degree. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 160–175, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 2, 27, 61, 103, 105

[42] Moran Feldman and Rani Izsak. Building a good team: Secretary problems and the supermodular degree. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1651–1670, 2017. 2, 27, 56

[43] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Improved competitive ratios for submodular secretary problems. In *APPROX*, pages 218–229, 2011. 61, 63

[44] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, 2011. 26

[45] Moran Feldman, Joseph (Seffi) Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems. In *ESA*, pages 784–798, 2011. 13, 26

[46] Moran Feldman, Ola Svensson, and Rico Zenklusen. A simple order-oblivious $O(\log\log(rank))$-competitive algorithm for the matroid secretary problem. In *SODA*, pages 1189–1201, 2015. 56, 61

[47] Moran Feldman and Rico Zenklusen. The submodular secretary problem goes linear, 2015. To appear in FOCS 2015. 57, 61

[48] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions – II. In *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Study*, pages 73–87. North-Holland Publishing Company, 1978. 15, 22, 26, 103

[49] H. N. Gabow. *Implementation of algorithms for maximum matching on nonbipartite graphs*. PhD thesis, Stanford University, 1974. 50

[50] Z. Galil. Efficient algorithms for finding maximal matching in graphs. Technical report, Columbia University, New York, 1983. 50

[51] Z. Galil, S. Micali, and H. N. Gabow. An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs. *SIAM Journal on Computing*, 15:120–130, 1986. 50

[52] Shayan Oveis Gharan and Jan Vondrák. On variants of the matroid secretary problem. *Algorithmica*, 67(4):472–497, 2013. 62

[53] Gagan Goel, Chinmay Karande, Pushkar Tripathi, and Lei Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. *SIGecom Exchanges*, 9(1):8, 2010. 26

[54] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatoria*, 1(2):169–197, 1981. 26

[55] Faruk Gul and Ennio Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87(1):95–124, 1999. 26

[56] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: offline and secretary algorithms. In *WINE*, pages 246–257. Springer-Verlag, 2010. 56, 57, 60, 61

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

[58] D. Hausmann, B. Korte, and T. Jenkyns. Worst case analysis of greedy type algorithms for independence systems. *Math. Prog. Study*, 12:120–131, 1980. 25

[59] E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k-set packing. *Computational Complexity*, 15:20–39, 2006. 122

[60] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. *Computational Complexity*, 15(1):20–39, May 2006. 19, 23, 42, 61

[61] Sungjin Im and Yajun Wang. Secretary problems: Laminar matroid and interval scheduling. In *SODA*, pages 1265–1274, 2011. 61

[62] Satoru Iwata and Kiyohito Nagano. Submodular function minimization under covering constraints. In *FOCS*, pages 671–680, 2009. 26

[63] Satoru Iwata and James B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *SODA*, pages 1230–1237, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics. 26

[64] Rani Izsak. Working together: Committee selection and the supermodular degree. AAMAS, 2017. accepted as an extended abstract. 27, 97

[65] Rani Izsak and Ola Svensson. Online welfare maximization: Beyond submodularity, 2016. In preparation. 9, 27

[66] Patrick Jaillet, José A. Soto, and Rico Zenklusen. Advances on matroid secretary problems: Free order model and laminar case. In *IPCO*, pages 254–265, 2013. 61, 62

[67] T. Jenkyns. The efficacy of the greedy algorithm. *Cong. Num.*, 17:341–350, 1976. 25, 26

[68] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA*, pages 630–631, 2005. 56, 61

[69] B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. *Annals of Discrete Math.*, 2:65–74, 1978. 25

[70] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *ICALP*, pages 508–520, 2009. 61

[71] Oded Lachish. $O(\log \log rank)$ competitive-ratio for the matroid secretary problem. In *FOCS*, pages 326–335, 2014. 56, 61

[72] Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Math. Oper. Res.*, 35(4):795–806, 2010. 26

[73] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55:270–296, 2006. Preliminary version in EC'2001. 7, 10, 15, 23, 24, 25, 28, 116

[74] Daniel J. Lehmann, Liadan O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49(5):577–602, 2002. 19, 122

[75] Tengyu Ma, Bo Tang, and Yajun Wang. The simulated greedy algorithm for several submodular matroid secretary problems. In *STACS*, pages 478–489, 2013. 61

[76] Julián Mestre. Greedy in approximation algorithms. In *ESA*, pages 528–539, 2006. 13

[77] G. Nemhauser and L. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978. 25

[78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming*, 14:265–294, 1978. 25

[79] Joel Oren and Brendan Lucier. Online (budgeted) social choice. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1456–1462, 2014. 104

[80] Konstantinos Poularakis, George Iosifidis, Georgios Smaragdakis, and Leandros Tassiulas. One step at a time: Optimizing SDN upgrades in ISP networks. In *IEEE INFOCOM*, 2017. to appear. 27

[81] Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *STOC*, pages 755–764, 2010. 104

[82] Prasad Raghavendra, David Steurer, and Madhur Tulsiani. Reductions between expansion problems. In *IEEE Conference on Computational Complexity*, pages 64–73, 2012. 104

[83] Piotr Skowron, Piotr Faliszewski, and Jérôme Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. In *AAAI*, pages 2131–2137, 2015. 97, 99, 103

[84] José A. Soto. Matroid secretary problem in the random assignment model. *SIAM Journal on Computing*, 42(1):178–211, 2013. 61, 62

[85] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, pages 67–74, 2008. 24

[86] Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304, 2013. 26

[87] Justin Ward. A (k+3)/2-approximation algorithm for monotone submodular k-set packing and general k-exchange systems. In *STACS*, pages 42–53, 2012. 26

[88] Anita Williams Woolley, Christopher F. Chabris, Alex Pentland, Nada Hashmi, and Thomas W. Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330:265–294, 2010. 56, 98, 101