OPTIMIZATION AND SEPARATION FOR STRUCTURED SUBMODULAR FUNCTIONS WITH CONSTRAINTS

A Thesis
Presented to
The Academic Faculty

by

Jiajin Yu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Algorithms, Combinatorics, and Optimization

Georgia Institute of Technology
May 2015

OPTIMIZATION AND SEPARATION FOR STRUCTURED SUBMODULAR FUNCTIONS WITH CONSTRAINTS

Approved by:

Shabbir Ahmed, Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Santanu S. Dey
School of Industrial and Systems Engineering
Georgia Institute of Technology

Richard J. Lipton
School of Computer Science
Georgia Institute of Technology

Sebastian Pokutta
School of Industrial and Systems Engineering
Georgia Institute of Technology

Prasad Tetali
School of Mathematics
Georgia Institute of Technology

Date Approved: December 1st, 2014

To my parents.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my advisor Professor Shabbir Ahmed, for his guidance and support throughout my years at Georgia Tech. His insightful ideas always enlighten me along the journey of research; his high standard for research sets an example for me. Our weekly discussion have always been fun and joyful. I really appreciate him allowing me to pursue research topics of my own interest freely. My Ph.D. experience is stimulating and exciting because of him.

I also want to thank the ACO program director, Professor Robin Thomas for offering me the opportunity to study in Georgia Tech and his great effort in running the ACO program with outstanding quality. I am also grateful to all the professors in the ACO program for their inspiring teaching in classes and sharing insightful ideas with me during talks and day-to-day discussions.

I cannot complete my research in Georgia Tech without the help of my colleagues: Abhishek Banerjee, Cristóbal Guzman, Qie He, Chun-Hung Liu, Lei Wang, Qianyi Wang, Linji Yang, and many more I am failing to mention here. Discussions with you have always been source of my inspiration in research. The jokes and laughs drive away the frustration during the research.

My life in Atlanta will not be so colorful without my friends: Chong Han, Fan Shi, Liangyi Sun, Peng Tang, Yi Wang, Yi Xiao, Biancun Xie, and Ke Zhou. Thank you all for bringing happiness during my stay in Atlanta. I will always cherish the memory with you guys and keep your kindness in my heart.

Special thanks to my girlfriend Minglu Lin, for her love and support when I needed most.

Finally I offer my gratitude to my family, especially to my parents. Their wholehearted love and encouragement make everything possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Various kinds of optimization problems involve nonlinear functions of binary variables that exhibit a property of diminishing marginal returns. Such a property is known as *submodularity*. Vast amount of work has been devoted to the problem of submodular optimization. On the algorithmic side, researchers have developed polynomial time algorithms that find an (approximately) optimal solution in the presence of simple constraints under the assumption of an oracle model of the function. On the integer programming side, where the goal is to find an optimal solution, existing approaches tackle the nonlinearity usually by solving a continuous convex relaxation of the nonlinear function of binary variables, and use a branch-and-bound procedure to force integrality constraints. The oracle model ignores available structural information of the function and the continuous relaxation of a nonlinear function of binary variables in an explicit form is usually quite loose. In this thesis, we exploit structural information for several classes of submodular optimization problems. We strive for polynomial time algorithms with improved approximation ratio and strong mixed-integer linear formulations of mixed-integer non-linear programs where the epigraph and hypograph of submodular functions of a specific form appear as a substructure together with other side constraints. Our contributions are as follows.

First, we develop approximation algorithms for the expected utility knapsack problem. We use the sample average approximation framework to approximate the stochastic problem as a deterministic knapsack-constrained submodular maximization problem, and then use an approximation algorithm to solve the deterministic counterpart. We show that a polynomial number of samples is enough for a deterministic approximation that is close in relative error. Then, exploiting the strict monotonicity of typical utility functions, we present an algorithm that maximizes an increasing submodular function over a knapsack constraint with approximation ratio better than the classical $(1 - 1/e)$ ratio.

Next, we present polyhedral results for the expected utility knapsack problem. We study

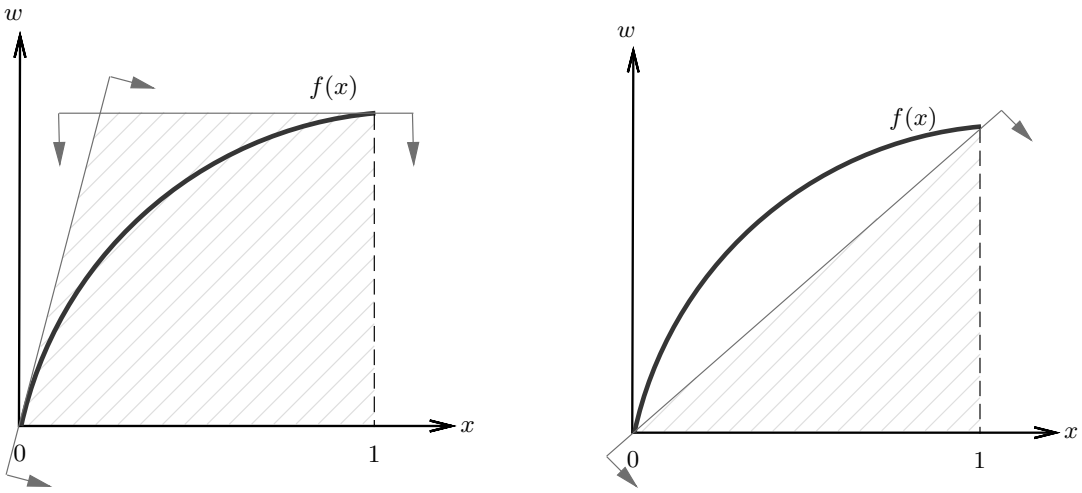
the mixed-integer nonlinear set: $\mathcal{P} = \{(w, x) \in \mathbb{R} \times \{0,1\}^n \mid w \leq f(a'x + d), b'x \leq B\}$ where $f : \mathbb{R} \mapsto \mathbb{R}$ is a concave function, $a, b \in \mathbb{R}^n$ are nonnegative vectors, d is a real number and B is a positive real number. We propose a family of inequalities for the convex hull of $\mathcal{P}$ by exploiting both the structure of the submodular function $f(a'x + d)$ and the knapsack constraint. Effectiveness of the proposed inequalities is shown by computational experiments on expected utility maximization problem with budget constraint using a branch-and-cut framework.

Finally, we study the mixed-integer nonlinear set: $\mathcal{Q} = \{(w, x) \in \mathbb{R} \times \{0,1\}^n : w \geq f(a'x), e'x \leq k\}$ where $f : \mathbb{R} \mapsto \mathbb{R}$ is a concave function, $a \in \mathbb{R}^n$ is a nonnegative vector, e is a vector of ones, and k is a positive integer. The set $\mathcal{Q}$ arises as a substructure in various constrained submodular minimization problems. We develop a strong linear formulation of the convex hull of $\mathcal{Q}$ by exploiting both the submodularity of $f(a'x)$ and the cardinality constraint $e'x \leq k$. We provide a full description of the convex hull of $\mathcal{Q}$ when the vector a has identical components. We also develop a family of facet-defining inequalities when the vector a has nonidentical components. We demonstrate the effectiveness of the proposed inequalities by solving mean-risk knapsack problems using a branch-and-cut framework.

CHAPTER I

INTRODUCTION

Various optimization problems, such as capital budgeting, mean-risk optimization, revenue maximization, involve nonlinear functions of binary variables that exhibit a property of diminishing marginal returns. Such property is known as *submodularity* and often arises from the concavity of risk aversion. Vast amount of work has been devoted to the problem of submodular optimization. On the algorithmic side, researchers have developed polynomial time algorithms that find an optimal solution, or a solution whose value is approximately close to the optimal if the problem is **NP**-hard. These works typically assume that only values of the function can be queried through an oracle and nothing else of the function is known. On the integer programming side, where the goal is to find an exact optimal solution, existing approaches tackle the nonlinearity usually by solving a continuous convex relaxation or an outer approximation of the nonlinear functions, and use branch-and-bound to force integrality constraints. Both approaches have their respective drawbacks. First, the available structural information of the function is often much richer than the one of an oracle model. Second, given an explicit form of a nonlinear function of binary variables, a continuous relaxation at the beginning of the branch-and-bound procedure can give a very loose bound that otherwise could be tightened if submodularity is exploited directly in the space of binary variables. In Figure 1 we illustrate this for a concave function $f(x)$ over $\{0, 1\}$. In Figure 1a, the shaded region represents the relaxation obtained by building an outer approximation of the nonlinear function $f(x)$ by two gradient inequalities at the points $x = 0$ and $x = 1$. In Figure 1b, the actual convex hull of the hypograph of $f(x)$ over $\{0, 1\}$ is shown. In this case, the convex hull can be obtained by studying how f behaves over $\{0, 1\}$, that is by the submodularity of $f(x)$. In this thesis, we exploit structural information for several classes of submodular optimization problems. We strive for polynomial time algorithms with improved approximation ratio and strong mixed-integer



(a) An outer approximation of $f(x)$ (b) The convex hull of $f(x)$

Figure 1: Comparison of outer approximation and convex hull of $f(x)$ over $x \in \{0, 1\}$

linear formulations of mixed-integer non-linear programs where the epigraph and hypograph of submodular functions of an explicit form appear as a substructure together with other side constraints.

In the remainder of this chapter, we will first define some notations and review some basic concepts of submodular optimization and mixed-integer programming. Then we will briefly describe the contributions of this thesis.

1.1 Notations and preliminaries

Submodular functions play a significant role in discrete optimization. Various basic functions in optimization, such as utility functions, cut functions and coverage functions, are submodular (cf. [49] for a comprehensive treatment). Let the index set $N = \{1, \ldots, n\}$. A function of binary variables $f : \{0, 1\}^N \to \mathbb{R}$ is submodular if for any two n-dimensional binary vectors x, y with $x \leq y$ (component-wise) and a unit vector e^i with $y_i = 0$, we have

$$f(x + e^i) - f(x) \geq f(y + e^i) - f(y).$$

Sometimes it is convenient to discuss submodular function using the notation of set functions. A set function $f : 2^N \to \mathbb{R}$ is submodular if for all $S, T \subseteq N$ with $S \subseteq T$, and

$i \in N \backslash T$,

$$f(S \cup \{i\}) - f(S) \geq f(T \cup \{i\}) - f(T).$$

From the definition above, we see similarity between submodular functions and concave functions as they both exhibit diminishing marginal returns.

From a different perspective, one might argue that submodularity is analogous to convexity. Specifically, the *continuous extension* of a submodular set function is a convex function and if the continuous extension of a set function is convex, then it is submodular [32]. This connection implies that submodular minimization can be done in polynomial time by the Ellipsoid method [20]. Later, combinatorial algorithms have been developed that minimize a submodular function in strongly polynomial time [27, 42].

Except for minimization without constraint, most problems in submodular optimization of general functions are **NP**-hard. For specific submodular functions arising in combinatorial optimization such as rank of an independent set in a matroid, the size of a cut of a graph, maximization without constraint or minimization with constraints are usually as hard as general submodular functions [13, 17, 28]. In recent years, there have been vast amount of elegant works on polynomial time approximation algorithms for submodular optimization (cf. [12, 17, 28, 30, 51]). Most of these assume a "value oracle" model where nothing except function values of different sets can be queried. For maximizing an increasing general submodular function with a cardinality constraint, the classic greedy algorithm in [36] (see Algorithm 1) finds an approximate optimal solution. Starting from an empty set, each time the algorithm selects an item in the ground set that has the largest marginal returns on the set of current selected items until k items (i.e., the cardinality constraint) have been selected. The value of the resulting set is at least $(1 - 1/e)$ factor of the optimal value. Later, Sviridenko [46] generalizes the algorithm to work with a knapsack constraint. The value oracle assumption is reasonable in some scenarios, but we lose the opportunity of improvement by not exploiting the structure of a function when it is available at hand.

In many applications, an exact optimal, not approximately optimal, solution is needed. In addition to that, optimization problems may have various kinds of side constraints, not

Algorithm 1: Greedy algorithm for an increasing general submodular function

Data: The set of elements $N = \{e_1, \ldots, e_n\}$
Result: A set S of size k

$S \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** k **do**
  $e \leftarrow \text{argmax}_{e' \in N} (f(S \cup \{e'\}) - f(S))$;
  $S \leftarrow S \cup \{e\}$;
  $N \leftarrow N \backslash \{e\}$;
end

only simple constraints like knapsack or matroid constraints. Therefore existing approximation algorithms, which usually can only deal with one kind of simple constraint, are not applicable. In these situations, a general approach to solve the problem is to build a mixed-integer linear program that models the optimization problem, and use the available solvers to find an optimal solution of the mixed-integer linear program. Below we first briefly review concepts of mixed-integer linear program, then we demonstrate two existing techniques of linearizing submodular constraints which can be used in building a mixed-integer linear programming formulation.

A *mixed-integer linear program* (MILP) is of the form:

$$\begin{aligned} \min \quad & c'x \\ \text{such that} \quad & Ax \leq b \\ & l \leq x \leq u \\ & x_i \in \mathbb{Z}, \quad \forall i \in I, \end{aligned} \tag{1}$$

where c, l, u are n-dimensional vectors, x is an n-dimensional vector of variables among which a subset $I \subseteq \{1, \ldots, n\}$ must be integers. The constraints on x are specified by A, an $m \times n$ matrix and b, an m-dimensional vector. We assume here all vectors and matrices are rational. Let P be the set of all feasible solutions of the MILP (1) and $\text{conv}(P)$ be the convex hull of the set P. By a fundamental theorem proved by Meyer [34], there exists a rational matrix $\tilde{A}$ and rational vector $\tilde{b}$ such that $\left\{x \mid \tilde{A}x \leq \tilde{b}\right\}$ defines the convex hull of P.

The exact convex hull of P is usually difficult to get. Often we turn to a *relaxation* that

is a superset of the convex hull. Consider a relaxation $P^1 \supset \text{conv}(P)$ which is represented only by linear constraints and no integrality constraints as $P^1 = \{x \mid A^1 x \leq b^1\}$. Let x^1 be an optimal solution of the linear program $\min\{c'x \mid A^1 x \leq b^1\}$. If there exists a point $x^0 \in P$ available to us and the objective $c'x^0$ equals to $c'x^1$, then we know that x^0 is an optimal solution of $\min\{c'x \mid x \in P\}$. On the other hand, if $x^1 \notin P$, then we can attempt to *separate* the point $x^1 \in P^1 \backslash P$ from P by a *cutting plane* $\alpha x \leq \beta$. In particular, a cutting plane $\alpha x \leq \beta$ is *valid* for $\text{conv}(P)$ when every point in $\text{conv}(P)$ satisfies the inequality, and the cutting plane separates a point $x^1 \notin \text{conv}(P)$ if $\alpha x^1 > \beta$. If for any point $x^1 \notin \text{conv}(P)$, we can find a valid cutting plane for $\text{conv}(P)$ that separates x^1 from $\text{conv}(P)$ in polynomial time, then we say we have a *polynomial time separation algorithm* for $\text{conv}(P)$. By the Ellipsoid method, through a polynomial time separation algorithm, an optimal solution of $\min\{c'x \mid x \in P\}$ can be found in polynomial time [20].

We often use *lifting* to generate valid inequalities in this thesis. The lifting procedure constructs a valid inequality for a high dimensional polyhedron from a given valid inequality for a lower dimensional polyhedron. We give a brief review of the idea of lifting in the following. For a comprehensive treatment, we refer readers to [35]. Consider a mixed integer set $P = \left\{(w, x) \in \mathbb{R} \times \{0,1\}^N \mid \sum_{i \in N} a_i x_i + w \leq b\right\}$. For a set $S \subseteq N$, let

$$P^0 = \{(w, x) \in P \mid x_i = 0, \quad \forall i \in N \backslash S\}$$

be a restriction of P. Let $N \backslash S = \{i_1, \ldots, i_m\}$ and denote $S^l = S \cup \{i_1, \ldots, i_l\}$, $P^l = \left\{(w, x) \in P \mid x_i = 0, \forall i \in N \backslash S^l\right\}$ for $l = 1, \ldots, m$. Denote $S^0 = S$. For $l = 1, \ldots, m$, given an inequality $\sum_{i \in S^{l-1}} \pi_i x_i + \pi_{n+1} w \leq \pi_0$ valid for $\text{conv}(P^{l-1})$, let

$$\gamma_l = \max\left\{\sum_{i \in S^{l-1}} \pi_i x_i + \pi_{n+1} w \mid \sum_{i \in S^{l-1}} a_i x_i + a_{i_l} + w \leq b, (w, x) \in \mathbb{R} \times \{0,1\}^n\right\}.$$

Then for any $\pi_{i_l} \leq \pi_0 - \gamma_l$, $\sum_{i \in S^{l-1}} \pi_i x_i + \pi_{i_l} x_{i_l} + \pi_{n+1} w \leq \pi_0$ is a valid inequality for $\text{conv}(P^l)$. This process is called *uplifting* since we have variable $x_{i_l} = 0$ in P_{l-1} and it is free in P_l. We can construct coefficients π_{i_l} from $l = 1$ to m and have a valid inequality for $\text{conv}(P)$. This process is called sequential lifting. If we lift variables in different orders,

then we get different coefficients. Define a lifting function $\Phi : \mathbb{R} \to \mathbb{R}$ as

$$\Phi(\alpha) = \pi_0 - \max \left\{ \sum_{i \in S} \pi_i x_i + \pi_{n+1} w \mid \sum_{i \in S} a_i x_i + \alpha + w \leq b, (w, x) \in \mathbb{R} \times \{0,1\}^n \right\}.$$

If the function is *superadditive* (i.e., $\Phi(\alpha_1) + \Phi(\alpha_2) \leq \Phi(\alpha_1 + \alpha_2)$), then we can lift all variables in one pass and get a valid inequality of $\text{conv}(P)$ as

$$\sum_{i \in S} \pi_i x_i + \sum_{i \in N \setminus S} \Phi(a_i) x_i + \pi_{n+1} w \leq \pi_0.$$

This is called *sequence independent lifting* since the order of variables to be lifted does not matter. We can also restrict binary variables to be one and solve corresponding optimization problems, which is similar to uplifting. This process is called *downlifting*.

Solvers of mixed-integer linear programs, such as CPLEX and Gurobi, have become quite efficient in recent years. In most cases, solvers do not need the complete description of the convex hull to find an optimal solution. Instead they use a *branch-and-cut* framework. On one hand, solvers exhaustively search for feasible solutions by branching; on the other hand, they derive valid cutting planes that produce tighter and tighter relaxations along the process. Eventually, solvers find a feasible solution whose objective value is the same as the optimal value of the relaxation. Since mixed-integer linear programs are quite general, formulating a given problem as MILP and using standard solvers to obtain an optimal solution can bypass the need for developing specialized algorithms for different problems. In this thesis, we develop general methods to transform mixed-integer non-linear formulations of some classes of submodular optimization problems to mixed-integer linear programs so that optimization applications involving submodularity can also benefit from the generality and efficiency of MILP solvers.

Below we review two important approaches to obtain mixed-integer linear formulations of the epigraph and hypograph of a submodular function, respectively. These two approaches will be the cornerstones of techniques we develop in this thesis for linearizing models involving structured submodular functions.

First, for the epigraph of a submodular function $E_f = \{(w, x) \in \mathbb{R} \times \{0,1\}^n \mid w \geq f(x)\}$, Edmonds [11] gives a complete description of $\text{conv}(E_f) = \{(w, x) \mid \pi_0 + \pi x \leq w, (\pi_0, \pi) \in \Pi\}$,

Algorithm 2: Edmonds' algorithm

Data: A n-dimensional vector $x \in [0,1]^n$
Result: The optimal primal solution p_S and its dual solution $\pi \in \mathbb{R}^{n+1}$
Sort components of x so that $x_1 \geq x_2 \geq \cdots \geq x_n$;
Add auxiliary $x_0 = 1, x_{n+1} = 0$;
$S_0 = \emptyset, p_{S_0} = x_0 - x_1, \pi_0 = f(\emptyset)$;
for $i \leftarrow 1$ **to** n **do**
  $S_i \leftarrow \{1, \ldots, i\}$;
  $p_{S_i} \leftarrow x_i - x_{i+1}$;
  $\pi_i = f(S_i) - f(S_{i-1})$;
end

where

$$\Pi = \left\{ (\pi_0, \pi) \in \mathbb{R}^{n+1} \mid \pi_0 + \sum_{i \in S} \pi_i \leq f(\mathbf{1}^S) - f(0), \forall S \subseteq N \right\},$$

and $\mathbf{1}^S$ is the characteristic vector of $S \subseteq N$. Although the number of inequalities for a complete description of $\text{conv}(E_f)$ is exponential, we have a polynomial time separation algorithm for $\text{conv}(E_f)$. The separation algorithm basically verifies whether a point (w, x) is a convex combination of points in E_f. In particular, for any point $(w, x) \in \mathbb{R} \times [0,1]^n$, we define the *continuous extension* of a submodular function f as $\mathcal{L}_f(x)$, whose value at point x is the optimal value of the following linear program

$$\begin{aligned} \mathcal{L}_f(x) := \min \quad & \sum_{S \subseteq N} p_S f(S) \\ \text{such that} \quad & \sum_{S: i \in S, S \subseteq N} p_S = x_i, \quad \forall i \\ & \sum_{S \subseteq N} p_S = 1 \\ & p_S \geq 0, \forall S, \quad S \subseteq N. \end{aligned}$$

A point (w, x) belongs to $\text{conv}(E_f)$ if and only if $\mathcal{L}_f(x) \leq w$. Algorithm 2 developed by Edmonds [11] solves the above linear program in polynomial time and gives a valid inequality $\pi_0 + \sum_i \pi_i x_i \leq w$ for $\text{conv}(E_f)$.

For the hypograph of general submodular function $H_f = \{(w, x) \in \mathbb{R} \times \{0,1\}^n \mid w \leq f(x)\}$, we do not know a polynomial time separation algorithm of its convex hull. Nevertheless, a mixed-integer linear formulation is known [35]. We use the set notation in the following and

let $\rho_i(S) = f(S \cup \{i\}) - f(S)$. For a submodular function f, the *submodular inequalities* [35] are defined as follows

$$w \leq f(S) - \sum_{i \in S} \rho_i(N \setminus \{i\})(1 - x_i) + \sum_{i \in N \setminus S} \rho_i(S) x_i, \quad \forall S \subseteq N \tag{2}$$

$$w \leq f(S) - \sum_{i \in S} \rho_i(S \setminus \{i\})(1 - x_i) + \sum_{i \in N \setminus S} \rho_i(\emptyset) x_i, \quad \forall S \subseteq N. \tag{3}$$

Thus a MILP formulation of the hypograph $H_f = \{(w, x) \in \mathbb{R} \times \{0,1\}^n \mid (2) \text{ or } (3)\}$.

1.2 Contributions

For applications in capital budgeting, mean-risk optimization, or utility maximization, submodularity often arises in a different form from the ones in combinatorial optimization (cf [4, 5, 29, 33, 37, 53]). Below we introduce a class of submodular functions which is applicable in these settings. The function is a univariate concave function applied to a linear function of binary variables. Let a be an n-dimensional nonnegative rational vector, $f : \mathbb{R} \to \mathbb{R}$ be a concave function, and $a'x$ be the dot product of two vectors of same dimension. Then the function of n-dimensional binary variables $g(x) := f(a'x)$ is submodular. To verify this, we check that for any $x, y \in \{0,1\}^n, x \leq y$ and a unit vector e^i with $y_i = 0$, we have $g(x + e^i) - g(x) \geq g(y + e^i) - g(y)$ because f is concave and every component of a is nonnegative. See Figure 2 for an example. If $a \in \mathbb{R}^n_-$, function $g(x)$ is also submodular. In this thesis, we exploit the structure of $g(x)$ to develop algorithms with better approximation ratio than the ones of general submodular functions and strong mixed-integer linear formulations of mixed-integer non-linear programs where the epigraph and hypograph of $g(x)$ appear as a substructure together with other side constraints. In Table 1, we summarize the results in this thesis.

In Chapter 2, we develop an approximation algorithm for the expected utility knapsack problem. The problem is to pick a set of items whose values are described by random variables to maximize the expected utility of the total value of the items picked while satisfying a constraint on the total weight of items picked. We consider the following solution approach for this problem: (i) use the sample average approximation framework to approximate the stochastic problem as a deterministic knapsack-constrained submodular

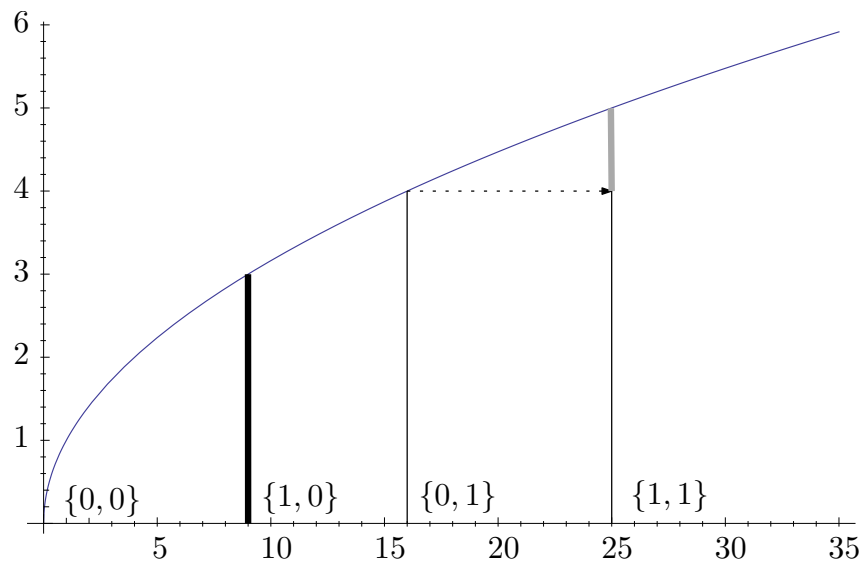


Figure 2: Diminishing marginal returns of $g(x) = f(a'x)$
An example of a submodular function $g(x) = f(a'x)$ when $n = 2$, $a = (9, 16)$ and the concave function $f(t) = \sqrt{t}$. The black bold line indicates the increase $g(1, 0) - g(0, 0)$; the gray bold line indicates the increase $g(1, 1) - g(0, 1)$.

maximization problem, and then (ii) use an approximation algorithm on the deterministic counterpart. We show that a polynomial number of samples is enough for a deterministic approximation that is close in relative error. Then, exploiting the strict monotonicity of typical utility functions, we present an algorithm that maximizes an increasing submodular function over a knapsack constraint with approximation ratio better than the $(1 - 1/e)$ ratio by Sviridenko [46]. For power utility functions we provide explicit approximation ratios leading to a polynomial time approximation algorithm. Assuming that the random values are completely described by a fixed and finite set of realizations, we also give a fully polynomial approximation scheme (FPTAS) for the expected utility knapsack problem with power utilities. The work in this chapter appears in the manuscript [57].

In Chapter 3, we present polyhedral results for the expected utility knapsack problem. For the mixed-integer nonlinear set: $\mathcal{P} = \{(w, x) \in \mathbb{R} \times \{0, 1\}^n \mid w \leq f(a'x + d), b'x \leq B\}$ where $f : \mathbb{R} \mapsto \mathbb{R}$ is a concave function, n is a positive integer, B is a positive real number, $a, b \in \mathbb{R}^n$ are nonnegative vectors, and $x'y$ denotes the scalar product of vectors x and y of same dimension, we propose a family of valid inequalities based on the MILP formulation

defined by (3) for the hypograph of f. We exploit both the structure of the submodular function and the knapsack constraint. Effectiveness of the proposed inequalities is shown by computational experiments on the expected utility maximization problem with budget constraint using a branch-and-cut framework. The work in this chapter appears in the manuscript [56].

In Chapter 4, motivated by concave cost combinatorial optimization problems, we study the mixed integer nonlinear set: $\mathcal{P} = \{(w, x) \in \mathbb{R} \times \{0,1\}^n : w \geq f(a'x), e'x \leq k\}$ where $f : \mathbb{R} \mapsto \mathbb{R}$ is a concave function, n and k are positive integers, $a \in \mathbb{R}^n$ is a nonnegative vector, $e \in \mathbb{R}^n$ is a vector of ones, and $x'y$ denotes the scalar product of vectors x and y of same dimension. A standard linearization approach for $\mathcal{P}$ is to exploit the fact that $f(a'x)$ is submodular with respect to the binary vector x and use the linearization due to Edmonds [11] discussed before. We extend this approach to take the cardinality constraint $e'x \leq k$ into account and provide a full description of the convex hull of $\mathcal{P}$ when the vector a has identical components. We also develop a family of facet-defining inequalities when the vector a has nonidentical components. Computational results using the proposed inequalities in a branch-and-cut framework to solve mean-risk knapsack problems show significant decrease in both time and the number of nodes over standard methods. The work in this chapter appears in the manuscript [58].

Table 1: Summary of results

	Algorithm	Polyhedral results
$\max f(a'x), e'x \leq k$	**NP**-hard, $(1-1/e)$ approximation algorithm [36] better than $(1-1/e)$ approximation ratio, [9, 52]	Valid inequalities for unconstrained case, i.e., $k = n$ [1] **Chapter 3: valid inequalities for** $k < n$
$\max f(a'x), b'x \leq B$	$(1-1/e)$ approximation ratio for general increasing submodular function, [46] **Chapter 2: better than** $(1-1/e)$ **approximation ratio**	**Chapter 3: valid inequalities** by deriving $e'x \leq k$ from $b'x \leq B$
$\min f(a'x), e'x \leq k$	Polynomial time solvable [38, 5]	Polynomial time separable for unconstrained case, i.e., $k = n$, [20] **Chapter. 4:** **identical components in** a**: complete convex hull description, which is polynomial time separable** **nonidentical components in** a**: valid inequalities**
$\min f(a'x), b'x \leq B$	**NP**-hard	**Chapter 4: valid inequalities** by deriving $e'x \leq k$ from $b'x \leq B$

CHAPTER II

MAXIMIZING EXPECTED UTILITY OVER A KNAPSACK CONSTRAINT: APPROXIMATION ALGORITHMS

2.1 Introduction

This chapter develops approximation algorithms for the expected utility knapsack problem. Given a ground set of n items $\mathcal{U} = \{1, \ldots, n\}$; a random non-negative vector of values $\tilde{a}$ for the items; a positive integer vector b of weights for the items; a positive integer capacity of B; and a *utility function* $f : \mathbb{R}_+ \mapsto \mathbb{R}_+$; the expected utility knapsack problem is to pick a subset S of items so to

$$\max_{S \subseteq \mathcal{U}} \{F(S) := \mathrm{E}\left[f(\tilde{a}(S))\right] \mid b(S) \leq B\}, \tag{4}$$

where $x(S) := \sum_{i \in S} x_i$. Note that the expectation above is with respect to the distribution of $\tilde{a}$. Throughout this chapter, we assume $f(0) = 0$ and $f(a(S)) \geq 1$ for any $a \sim \tilde{a}$ and $S \neq \emptyset$. Therefore $F(\emptyset) = 0$ and $F(S) \geq 1$ for $S \neq \emptyset$. We use **poly** to denote a polynomial in all its parameters.

Expected utility theory is a well-known framework for choice under uncertain payoffs [50, 41]. Choice A is better than choice B if the expected utility of the payoff of A is larger than that of B. Risk attitudes may be different across different decision makers, and utility functions serve to model their risk preferences. In this chapter we assume that the utility function f is strictly increasing and concave which correspond to risk-averse preferences. Commonly used utility functions such as log-utility $f(t) = \log t$, exponential utility $f(t) = 1 - e^{-\alpha t}$ for $\alpha > 0$, and power utility $f(t) = t^p$ for $0 < p < 1$, all satisfy this assumption.

Concavity of f along with the non-negativity of $\tilde{a}$ imply that the expected utility F is a submodular function of the selected set S. Accordingly, (4) is a submodular maximization problem with a knapsack constraint. It is well-known that in general the approximation ratio for such problems is bounded by $1 - 1/e$ [13]. Moreover a variant of the greedy algorithm achieves this bound [46]. However these results assume a value oracle model

where the underlying submodular function is general and can be evaluated exactly.

In (4) evaluation of F requires evaluating a multidimensional integral over the distribution of $\tilde{a}$. Moreover, the distribution of $\tilde{a}$ may not be explicitly available, but only available through a sampling oracle. In such a setting, exact evaluation of F is impossible. In this chapter we adopt the sample average approximation (SAA) framework [43] towards approximately evaluating F. In SAA the original distribution of the uncertain parameters is replaced by an empirical distribution by sampling a certain number of scenarios.

The sample average approximation of (4) is

$$\max_{S \subseteq \mathcal{U}} \left\{ F^N(S) = \frac{1}{N} \sum_{i=1}^{N} f(a_i(S)) \mid b(S) \leq B \right\}, \tag{5}$$

where $\{a_1, \ldots, a_N\}$ is an i.i.d sample of $\tilde{a}$. Note that F^N is a submodular function and (5) is a deterministic knapsack constrained submodular maximization problem. It follows from classical SAA theory [43] that by solving (5) corresponding to a sufficient number of samples N using an approximation algorithm of a given absolute error δ, with high probability, we can obtain a solution to the original problem (4) whose absolute error is not too large compared to δ. Moreover the required sample size N is polynomial with respect to problem dimension.

If (5) is solved using a *relative* error approximation algorithm (such as those in the submodular optimization literature) we need to adapt the SAA theory to recover a corresponding relative error for the true problem (4). We make this adaptation. Further we develop an approximation algorithm for solving (5) based on maximizing increasing submodular functions over a knapsack constraint. As an aside, we also develop a fully polynomial-time approximation scheme (FPTAS) for (4) when the underlying distribution is finite and the utility function is positively homogenous. Specifically, the contribution of this chapter is three-fold:

SAA analysis under relative error: We prove that with high probability only polynomial number of samples is enough for an approximation algorithm that solves the SAA problem with relative error to give an approximate solution to the true stochastic problem

of similar relative error. The works by Shmoys and Swamy [48], and Charikar et al. [8] are most relevant to our work as they both considered approximation algorithms with relative error for 2-stage stochastic optimization, rather than the absolute error usually considered in stochastic programming. However the polynomial sample size in their results depends on ratio between the cost of the first stage and the cost of the second stage, which is not applicable to our single stage setting.

Increasing submodular maximization over a knapsack: The increasing and concavity properties of common utility functions imply that (5) involves maximizing an increasing submodular function over a knapsack constraint. Sviridenko [46] recently developed a greedy algorithm to maximize an increasing submodular function over a knapsack constraint with approximation ratio $1 - 1/e$. We adapt this algorithm and its analysis exploiting the strict monotonicity of the utility function and show an approximation ratio *better than* the $1-1/e$ bound. For power utility functions, we explicitly characterize the approximation ratio as a function of the budget B and the exponent of the power function. Some other works that have improved on the $1 - 1/e$ bound are by Conforti and Cornuéjols [9], and Vondràk [52]. However these consider cardinality constraints and matroid constraints, respectively, and are not applicable in our knapsack setting.

An FPTAS for finite distribution: When the distribution is finite and utility function is positively homogenous, we give an FPTAS for the problem. Our algorithm largely follows the work of Ibarra and Kim [25] who give an FPTAS for the standard knapsack problem.

We close this section with a brief discussion of some additional related literature. Li and Deshpande [31] study the problem of maximizing expected utility for various combinatorial optimization problems. They assume that the random coefficients are independent to simplify the expectation operation and use an approximation of the utility function. We allow more general distribution but are restricted to the knapsack setting. Klastorin [29] study a similar problem but he assumes exact evaluation of the expectation objective and gives

an algorithm that solves a continuous relaxation of the problem and then uses that in a branch-and-bound algorithm. Asadpour et.al. [3] study maximizing a stochastic submodular function under matroid constraints. They assume exact evaluation of the expectation objective and do not consider increasing submodular functions. Mehrez and Sinuany-Stern [33] study a variation of the problem arising in resource allocation applications, but in their model the utility of items are separable which is different from our setting.

2.2 Sample average approximation

In this section, we adapt the classical SAA theory (cf. [44]) which corresponds to an absolute error setting to our required setting of relative error. We consider a generalization of (4):

$$\max_{S\subseteq\mathcal{U}} \{F(S) = \mathrm{E}\left[f(\tilde{a}, S)\right] \mid S \in X\}, \tag{6}$$

where X is the constraint set (e.g. knapsack constraint) and $f : 2^{\mathcal{U}} \mapsto \mathbb{R}_+$, parameterized by a, is a nonnegative set function. The sample average approximation of (6) is

$$\max_{S\subseteq\mathcal{U}} \left\{ F^N(S) = \frac{1}{N}\sum_{i=1}^{N} f(a_i, S) \mid S \in X \right\}, \tag{7}$$

where $\{a_1, \ldots, a_N\}$ is an i.i.d sample of $\tilde{a}$. Then (4) is a special case of (6) and (5) is a special case of (7). Let S^* be an optimal solution of (6). We make the following assumption on $f(a, S)$ and $\mathrm{E}\left[f(\tilde{a}, S)\right]$.

Assumption 1. For any $a \sim \tilde{a}, S \in X$, and $S \neq \emptyset$, we assume $f(a, S) \geq 1$. Therefore $F^N(S) \geq 1$ and $F(S) \geq 1$. For any $S \in X$, we also assume $\mathrm{E}\left[f(\tilde{a}, S)\right]$ is well-defined and finite, and $\mathrm{E}\left[e^{tf(\tilde{a},S)}\right]$ is finite in a neighborhood of $t = 0$.

Using the above assumption and standard Large Deviation analysis (cf. [43]), we can show that if N is large enough, for every $S \in X$, $F^N(S)$ is close to $F(S)$ in a relative sense.

Lemma 2. *Given $\gamma > 0$, let $\sigma^2 = \max\{\mathrm{Var}\left[f(\tilde{a}, S)\right] \mid S \in X\}$ and S^* be an optimal solution of the problem. If $N \geq \frac{2\sigma^2}{\epsilon^2}\log\frac{|X|}{\gamma}$, then*

$$\Pr\left\{ \bigcap_{S\in X} \left|F(S) - F^N(S)\right| \leq \epsilon F(S^*) \right\} \geq 1 - 2\gamma. \tag{8}$$

Proof. Let $\{a_1, \ldots, a_N\}$ be the i.i.d sample defining $F^N(S)$. Let A_1 be the event that there exists a set S such that $F(S) - F^N(S) > \epsilon F(S^*)$, and let A_2 be the event that there exists a set S such that $F^N(S) - F(S) > \epsilon F(S^*)$. Let $\sigma_S^2 = \mathrm{Var}\left[f(\tilde{a}, S)\right]$, then $\sigma^2 = \max_{S \in X} \sigma_S^2$. If we can show that when $N \geq \frac{2\sigma^2}{\epsilon^2} \log \frac{|X|}{\gamma}$ we have $\Pr\{A_1\} \leq \gamma$ and $\Pr\{A_2\} \leq \gamma$ then we have the desired inequality (8).

Let us prove that $\Pr\{A_1\} \leq \gamma$.

$$\begin{aligned}
&\Pr\left\{\bigcup_{S \in X} F(S) - F^N(S) > \epsilon F(S^*)\right\} \\
\leq &\sum_{S \in X} \Pr\left\{F(S) - F^N(S) > \epsilon F(S)\right\} && \text{(since } \ F(S^*) \geq F(S)) \\
= &\sum_{S \in X} \Pr\left\{F^N(S) < (1-\epsilon) F(S)\right\}
\end{aligned}$$

By Assumption 1, we know that $F(S)$ is finite for every S and $\mathrm{E}\left[e^{tf(\tilde{a},S)}\right]$ is finite in a neighborhood of $t = 0$. So if ϵ is sufficiently small, by Large Deviation Theory (cf. [44, Sec 7.2.8]), we have

$$\Pr\left\{F^N(S) < (1-\epsilon) F(S)\right\} \leq \exp\left(-\frac{N(\epsilon F(S))^2}{2\sigma_S^2}\right).$$

Thus

$$\begin{aligned}
\sum_{S \in X} \Pr\left\{F^N(S) < (1-\epsilon) F(S)\right\} \leq &\sum_{S \in X} \exp\left(-\frac{N(\epsilon F(S))^2}{2\sigma_S^2}\right) \\
\leq &|X| \exp\left(-\frac{N\epsilon^2}{2\sigma^2}\right) && \text{(by Assumption 1)} \\
\leq &\gamma && \text{(by the chosen value of } N),
\end{aligned}$$

which proves $\Pr\{A_1\} \leq \gamma$. The proof for $\Pr\{A_2\} \leq \gamma$ is identical, which we omit here. □

Equipped with the lemma above, we are ready to show that we can use any algorithm that solves (7) approximately to solve (6) without losing too much.

Theorem 3. *Given an algorithm that solves* (7) *with approximation ratio* β*, with probability at least* $1 - 2\gamma$*, we can use the same algorithm to solve the stochastic problem* (6) *with approximation ratio* $\beta(1-\epsilon) - \epsilon$ *by sampling* $\tilde{a}$ *at least* $\frac{2\sigma^2}{\epsilon^2} \log \frac{|X|}{\gamma}$ *times.*

Proof. If we sample $N \geq \frac{2\sigma^2}{\epsilon^2} \log \frac{|X|}{\gamma}$ times of $\tilde{a}$, by Lemma 2, we know that for any $S' \in X$, we have $\left|F(S') - F^N(S')\right| \leq \epsilon F(S^*)$ with probability at least $1 - 2\gamma$. Assume the event happens. Let S^* be an optimal solution of (6). Let S be a β-approximation solution of (7). Then $F^N(S) \geq \beta F^N(S^*)$. We have

$$\begin{aligned}
F(S) &\geq F^N(S) - \epsilon F(S^*) && \text{(by Lemma 2)} \\
&\geq \beta F^N(S^*) - \epsilon F(S^*) && \text{(by Approximation ratio associated with } S) \\
&\geq \beta(F(S^*) - \epsilon F(S^*)) - \epsilon F(S^*) && \text{(by Lemma 2)} \\
&= (\beta(1-\epsilon) - \epsilon) F(S^*). && \square
\end{aligned}$$

Remark 4. Note that σ^2 is a problem specific parameter. We can bound σ^2 under additional assumptions (see Theorem 11 in Section 2.3.2.)

2.3 Increasing submodular maximization over a knapsack

In this section, we will give an algorithm for maximizing a nonnegative increasing submodular function over a knapsack constraint. Recall the universe is $\mathcal{U} = \{1, \ldots, n\}$ and the weight of element i is $b_i \in \mathbb{N}_+$. Let $F : 2^{\mathcal{U}} \mapsto \mathbb{R}_+$ be a nonnegative increasing function. It is a submodular function if and only if

$$F(S \cup \{i\}) - F(S) \geq F(T \cup \{i\}) - F(T), \forall S \subseteq T, i \notin T.$$

Another property of an increasing submodular function is the following:

$$F(T) \leq F(S) + \sum_{i \in T \setminus S} (F(S \cup \{i\}) - F(S)), \forall S, T \subseteq \mathcal{U}. \tag{9}$$

The problem we are interested in is of the following general form:

$$\max_{S \subseteq \mathcal{U}} \{F(S) \mid b(S) \leq B\}. \tag{10}$$

Following Sviridenko [46], we propose Algorithm 4 to solve (10) approximately. The algorithm first picks the best set S_1 among all sets of size less than a prescribed constant K. In the second step, for each set of size K, the algorithm greedily packs items into the set. Let the best set in this step be S_2. Finally, the algorithm outputs the better one of

Algorithm 3: Greedy

Let K be a constant ;
$S_1 = \text{argmax}_S \{F(S) \mid |S| < K\}$;
initialize $S_2 = \emptyset$;
forall the $\underline{S} \subseteq \mathcal{U}, |\underline{S}| = K$ **do**
  $U = \mathcal{U}, S = \underline{S}$;
  while $U \backslash S \neq \emptyset$ **do**
    $i = \text{argmax}_{i \in U \backslash S} \frac{F(S \cup \{i\}) - F(S)}{b_i}$, break tie arbitrary ;
    if $b(S) + b_i \leq B$ **then**
      $S \leftarrow S \cup \{i\}$;
    else
      $U \leftarrow U \backslash \{i\}$;
    end
  end
  if $F(S) > F(S_2)$ **then**
    $S_2 \leftarrow S$;
  end
end
Result: $\max \{F(S_1), F(S_2)\}$ and its corresponding set

S_1 and S_2. The main departure from the algorithm in [46] is that here we enumerate and extend all sets of size K, instead of sets of size 3 as in [46].

The constant K used in Algorithm 3 is based on the following measure of monotonicity of the increasing function $F(S)$:

Definition 5. Given an instance of (10), let S^* be an optimal solution of (10). We define

$$\alpha = \max_{i \notin S^*} \frac{F(S^* \cup \{i\}) - F(\{i\})}{F(S^*) - F(\{i\})}. \tag{11}$$

Note that since F is increasing, $\alpha > 1$. We now present the theorem showing that Algorithm 3 gives better approximation ratio than the one given by Sviridenko in [46] when the function is strictly increasing.

Theorem 6. *Given a nondecreasing submodular function F with α, by setting $K = \left\lceil e^{\alpha'} \right\rceil$ for any $\alpha' \geq 1$, Algorithm 3 solves the problem with an approximation ratio of at least $1 - e^{-\min(\alpha, \alpha')}$.*

Remark 7. Note that since F is increasing, even if we do not know the exact value of α as defined in (11), by setting $K = \left\lceil e^{\alpha'} \right\rceil$ with $\alpha' > 1$, Algorithm 3 is a polynomial time

algorithm with approximation ratio strictly better than the $(1-1/e)$ ratio of the algorithm proposed in [46].

2.3.1 Proof of Theorem 6

Without loss of generality, we assume that any optimal solution of (10) is of size at least K, because otherwise we would find an optimal solution as S_1. Let S^* be an optimal set of size at least K. Before going further, we define a subset of S^* of size K. We order the items in S^* such that

$$j_t \in \underset{i \in S^* \setminus \{j_1,\ldots,j_{t-1}\}}{\operatorname{argmax}} \left(F(\{j_1,\ldots,j_{t-1}\} \cup \{i\}) - F(\{j_1,\ldots,j_{t-1}\})\right), 1 \le t \le |S^*|,$$

and let $Y = \{j_1,\ldots,j_K\}$ be the set of the first K items in S^*. Let S be the set the algorithm extends from the set Y. It suffices to prove the desired approximation result for S since the greedy algorithm enumerates all sets of size K.

We may assume $S \neq S^*$. Let $S^0 \subset S$ be the last set that the algorithm considers while extending Y such that $S^0 \subset S^*$. Note that $Y \subseteq S^0$. From the assumption $S \neq S^*$, we know that $S^0 \neq S$ because otherwise the greedy algorithm should not stop at S, and $S^0 \neq S^*$ because otherwise S is strictly better than S^*. Following the set S^0, let the items added into the solution by the greedy algorithm be $i_1,\ldots,i_T$. Let $i_{T+1} \in S^*$ be the first one excluded by the greedy algorithm because of the budget overflow. It is without loss of generality since if some item is neither in S^* nor in the greedy solution, then we may remove it from the universe without affecting the analysis. Item i_{T+1} must exist because $S \neq S^*$. Otherwise since every item is considered at some time during the greedy algorithm, it must be the case that $S^* \subset S$. Then $F(S) > F(S^*)$, which is a contradiction.

Let $S^t = S^0 \cup \{i_1,\ldots i_t\}$. We call these sets *partial solutions* of the greedy algorithm. Let $c_t = (F(S^t) - F(S^{t-1}))/b_{i_t}$. Define $g(S^t) = F(S^t) - F(S^0)$, and let $g(S^*) = F(S^*) - F(S^0)$. Denote $\bar{B} = B - b(S^0)$. Note that the greedy solution S may be strictly larger than S^T. But we will show $F(S^T)$ is large enough to give us the approximation ratio we need.

We will first show that when $\alpha' = \alpha$, the partial solution S^T obtained by extending Y as defined earlier, achieves the desired approximation ratio $1-e^{-\alpha}$. In the end we will show the case when $\alpha' \neq \alpha$.

To prove the approximation ratio, we decompose $F(S^T) = F(S^0) + g(S^T)$ and replace $g(S^T)$ by two terms $g(S^T) - g(S^{T+1})$ and $g(S^{T+1})$. We then lower bound each term as a constant fraction of the optimal value separately.

First we lower bound the value of $g(S^T) - g(S^{T+1})$. The lower bound needs the following lemma which is a generalization of an inequality in [46].

Lemma 8. *For an item $j \in \mathcal{U} \backslash Y$, and set $Z \subseteq \mathcal{U} \backslash \{j_1, \ldots, j_K, j\}$, the difference between $F(Y \cup Z \cup \{j\})$ and $F(Y \cup Z)$ can be upperbounded as*

$$F(Y \cup Z \cup \{j\}) - F(Y \cup Z) \leq \frac{1}{K} F(Y). \tag{12}$$

Proof. To see this, recall we order $S^* = \left\{j_1, \ldots, j_{|S^*|}\right\}$ by the following rule:

$$j_t \in \underset{i \in S^* \backslash \{j_1, \ldots, j_{t-1}\}}{\operatorname{argmax}} \left(F(\{j_1, \ldots, j_{t-1}\} \cup \{i\}) - F(\{j_1, \ldots, j_{t-1}\})\right), 1 \leq t \leq |S^*|,$$

and Y is the set of the first K items among S^*. Notice that the following inequalities hold for any $0 \leq t \leq K-1$.

$$\begin{aligned}
& F(Y \cup Z \cup \{j\}) - F(Y \cup Z) \\
\leq & F(\{j_1, \ldots, j_t\} \cup \{j\}) - F(\{j_1, \ldots, j_t\}) && \text{(Submodularity)} \\
\leq & F(\{j_1, \ldots, j_t, j_{t+1}\}) - F(\{j_1, \ldots, j_t\}) && \text{(ordering of the set } S^*)
\end{aligned}$$

Summing up all inequalities of form

$$F(Y \cup Z \cup \{j\}) - F(Y \cup Z) \leq F(\{j_1, \ldots, j_t, j_{t+1}\}) - F(\{j_1, \ldots, j_t\}),\ 0 \leq t \leq K-1$$

we have

$$K\left(F(Y \cup Z \cup \{j\}) - F(Y \cup Z)\right) \leq F(Y). \qquad \square$$

Now let $Y \cup Z = S^T$ and $j = i_{T+1}$. Apply Lemma 8, and we have

$$g(S^{T+1}) - g(S^T) \leq \frac{1}{K} F(Y). \tag{13}$$

To bound $g(S^{T+1})$, we will show that it is a constant fraction of $g(S^*)$. First we use the definition of α and the property of the increasing submodular function (9) to upper bound $g(S^*)$ as follows.

Lemma 9.

$$g(S^*) \leq \min_{1 \leq t \leq T} \left\{ g(S^t) + \frac{1}{\alpha} \bar{B} c_{t+1} \right\}. \tag{14}$$

Proof. First we show for any $S^t, 1 \leq t \leq T$, $F(S^* \cup S^t) - F(S^t) \geq \alpha(F(S^*) - F(S^t))$. Let i^* be an item that attains the maximum in (11). Since we enumerate all sets of size K, there must be a set including the i^* defined in (11). Let S^1 be the first partial solution that includes i^*. Therefore we have $F(S^t) \geq F(i^*), F(S^t) < F(S^*)$ and $F(S^t \cup S^*) \geq F(\{i^*\} \cup S^*)$ by the monotonicity of F. Also F is nonnegative everywhere. Then

$$\begin{aligned} \frac{F(S^* \cup S^t) - F(S)}{F(S^*) - F(S^t)} &\geq \frac{F(S^* \cup S^t) - F(\{i^*\})}{F(S^*) - F(\{i^*\})} \\ &\geq \frac{F(S^* \cup \{i^*\}) - F(\{i^*\})}{F(S^*) - F(\{i^*\})} \\ &= \alpha \end{aligned}$$

We illustrate the deduction of inequality above in Figure 3. Starting from inequality $F(S^*) \leq F(S^t) + \frac{1}{\alpha}\left(F(S^* \cup S^t) - F(S^t)\right)$, we apply (9) and get

$$F(S^*) \leq F(S^t) + \frac{1}{\alpha} \sum_{i \in S^* \setminus S^t} \left(F(S^t \cup \{i\}) - F(S^t)\right).$$

Replace F by g, and we have

$$\begin{aligned} g(S^*) &\leq g(S^t) + \frac{1}{\alpha} \sum_{i \in S^* \setminus S^t} \left(g(S^t \cup \{i\}) - g(S^t)\right) \\ &\leq g(S^t) + \frac{1}{\alpha} \sum_{i \in S^* \setminus S^t} b_i c_{t+1} && \text{(By Algorithm 3)} \\ &\leq g(S^t) + \frac{1}{\alpha} \sum_{i \in S^* \setminus S^0} b_i c_{t+1} \\ &= g(S^t) + \frac{1}{\alpha}(b(S^*) - b(S^0)) c_{t+1} \\ &\leq g(S^t) + \frac{1}{\alpha}(B - b(S^0)) c_{t+1}. && \square \end{aligned}$$

We break down the right side of (14) into a summation of smaller increments. This is achievable because $g(S^{t+1}) - g(S^t) = F(S^{t+1}) - F(S^t) = c_t b_{i_t}$ and b_{i_t} is an integer. In particular, let $\bar{B}_t = b(S^t) - b(S^0), 1 \leq t \leq T+1$, and $\bar{B}_0 = 0$. Also recall $\bar{B} = B - b(S^0)$. Define $\rho_l, 1 \leq l \leq \bar{B}_{T+1}$ as follows: $\rho_l = c_t$ if $\bar{B}_{t-1} + 1 \leq l \leq \bar{B}_t$. Therefore we have

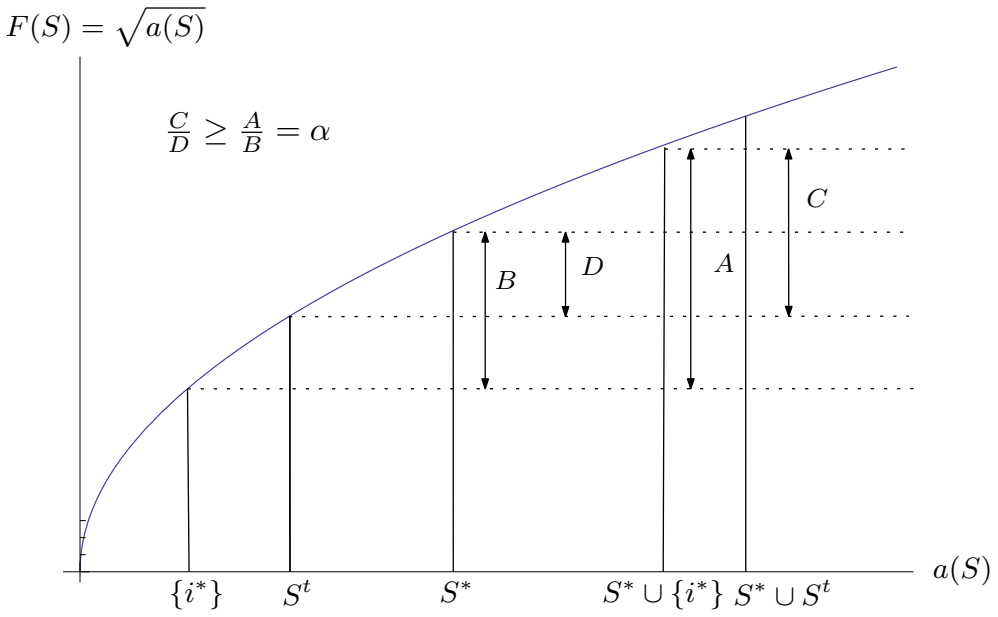


Figure 3: Illustration of $\frac{F(S^* \cup S^t) - F(S)}{F(S^*) - F(S^t)} \geq \alpha$

Let i^* be an item that attains the maximum in (11). For each segment, $A = F(S^* \cup \{i^*\}) - F(\{i^*\}), B = F(S^*) - F(i^*), C = F(S^* \cup S^t) - F(S^t)$ and $D = F(S^*) - F(S^t)$. By Definition 5, $\alpha = A/B$. In Figure 3, we see $C/D \geq A/B$.

$g(S^t) = \sum_{l=1}^{\bar{B}_t} \rho_l$. Notice that $\rho_1, \ldots, \rho_{\bar{B}_{T+1}}$ is a nonincreasing sequence. Then the right side of (14) can be written in the following way.

Lemma 10.

$$\min_{1 \leq t \leq T} \left\{ g(S^t) + \frac{\bar{B}}{\alpha} c_{t+1} \right\} = \min_{1 \leq s \leq B_{T+1}} \left\{ \sum_{j=1}^{s-1} \rho_j + \frac{\bar{B}}{\alpha} \rho_s \right\}. \tag{15}$$

Proof. We can rewrite the left side of (15) in the following way.

$$\min_{1 \leq t \leq T} \left\{ g(S^t) + \frac{\bar{B}}{\alpha} c_{t+1} \right\} = \min_{1 \leq t \leq T} \left\{ \sum_{l=1}^{B_t} \rho_l + \frac{\bar{B}}{\alpha} \rho_{B_{t+1}} \right\}.$$

Then we claim that

$$\min_{1 \leq t \leq T} \left\{ \sum_{l=1}^{B_t} \rho_l + \frac{\bar{B}}{\alpha} \rho_{B_{t+1}} \right\} = \min_{1 \leq s \leq B_{T+1}} \left\{ \sum_{j=1}^{s-1} \rho_j + \frac{\bar{B}}{\alpha} \rho_s \right\}.$$

Assume that $\min_{1 \leq s \leq \bar{B}_{T+1}} \left\{ \sum_{l=1}^{s-1} \rho_l + \frac{\bar{B}}{\alpha} \rho_s \right\}$ does not attain minimum at an $s = \bar{B}_t + 1$ for some t. Let $\bar{B}_t + 1 < s \leq \bar{B}_{t+1}$ for some t. Then $\sum_{l=1}^{\bar{B}_t} \rho_l + \frac{\bar{B}}{\alpha} \rho_{\bar{B}_t + 1} < \sum_{l=1}^{s-1} \rho_l + \frac{\bar{B}}{\alpha} \rho_s$ because $\bar{B}_t < s - 1$ and $\rho_{\bar{B}_t + 1} \leq \rho_s$. This leads to a contradiction. □

To bound the right side of (15), we need the following inequality. For a positive integer P, $D > 0$, $\rho_l \geq 0$ for $1 \leq l \leq P$, and $\rho_1 > 0$, then

$$\frac{\sum_{l=1}^{P} \rho_l}{\min_{t=1,\ldots,P}(\sum_{l=1}^{t-1} \rho_l + D\rho_t)} \geq 1 - (1 - \frac{1}{D})^P \geq 1 - e^{-\frac{P}{D}}. \tag{16}$$

The above inequality is analogous to one in [55]. The only difference is that D is a positive integer in [55], but the fact that D is an integer is not required in its proof. Notice $g(S^{T+1}) = \sum_{l=1}^{\bar{B}_{T+1}} \rho_l$. Combine (14) and (15), and apply (16) by setting $P = \bar{B}_{T+1}, D = \bar{B}/\alpha$, we have

$$\frac{g(S^{T+1})}{g(S^*)} \geq \frac{\sum_{l=1}^{\bar{B}_{T+1}} \rho_l}{\min_{1 \leq s \leq \bar{B}_{T+1}}(\sum_{j=1}^{s-1} \rho_j + \frac{\bar{B}}{\alpha}\rho_s)} \geq 1 - e^{-\alpha \bar{B}_{T+1}/\bar{B}}.$$

Since adding item i_{T+1} makes the budget overflow, we have $b(S^T) + b_{T+1} > B$. Therefore $\bar{B}_{T+1} = b(S^T) + b_{T+1} - b(S^0) \geq B - b(S^0) = \bar{B}$, and

$$g(S^{T+1}) \geq \left(1 - e^{-\alpha \bar{B}_{T+1}/\bar{B}}\right) g(S^*) \geq \left(1 - e^{-\alpha}\right) g(S^*) \tag{17}$$

Note that we have used the fact $\overline{B}_{T+1} \geq \overline{B}$, where the inequality is actually strict. This is the same as in the proof of $1 - 1/e$ in [46].

Now we combine (13) and (17), and get the desired approximation ratio of Theorem 6.

$$\begin{aligned}
F(S) &\geq F(S^T) = F(S^0) + g(S^T) - g(S^{T+1}) + g(S^{T+1}) \\
&\geq F(S^0) - \frac{1}{K} F(Y) + (1 - e^{-\alpha}) g(S^*) \\
&= F(S^0) - F(Y) + F(Y) - \frac{1}{K} F(Y) + (1 - e^{-\alpha}) g(S^*) \qquad (18) \\
&\geq (1 - e^{-\alpha})(F(S^0) - F(Y)) + (1 - e^{-\alpha}) F(Y) + (1 - e^{-\alpha}) g(S^*) \qquad (K \geq e^{\alpha}) \\
&= (1 - e^{-\alpha})(F(S^0) + (g(S^*)) = (1 - e^{-\alpha}) F(S^*).
\end{aligned}$$

Now notice for $\alpha' < \alpha$, as long as $K = \left\lceil e^{\alpha'} \right\rceil$, (18) can be lower bounded by $(1 - e^{-\alpha'})F(S^*)$. For $\alpha' > \alpha$, since we enumerate more subsets ($K > e^{\alpha}$), the approximation ratio $1 - e^{-\alpha}$ can also be achieved.

2.3.2 Combining SAA and Algorithm 3

Since (4) and (5) are special cases of (7) and (6), and we know that (5) is a submodular maximization problem, we have the following corollary.

Theorem 11. *Assume that* $\max_i \operatorname{Var}[\tilde{a}_i]$ *can be bounded by a constant. Given* $\gamma > 0$ *and* $\alpha' \geq 1$*, with probability at least* $1 - 2\gamma$*, Algorithm 3 solves the stochastic problem* (4) *with approximation ratio* $(1 - e^{-\min(\alpha,\alpha')})(1-\epsilon) - \epsilon$ *in time* $\mathbf{poly}\left(2^{e^{\alpha'}}, n, \frac{1}{\epsilon}, \log\frac{1}{\gamma}\right)$.

Proof. Recall that $\sigma^2 = \max_{S\subseteq\mathcal{U}}\{\operatorname{Var}[f(\tilde{a}(S))] \mid b(S) \leq B\}$. By Theorem 3, we need $\frac{2\sigma^2}{\epsilon^2}\log\frac{|X|}{\gamma}$ i.i.d. samples of $\tilde{a}$ to ensure that the sample average approximation is close to the expected value with probability at least $1-2\gamma$. Take the upper bound of $|X|$ as 2^n then $N = \frac{2\sigma^2}{\epsilon^2}(n+\log\frac{1}{\gamma})$ in (5). Since f is a concave function, we have $f(a(S)) \leq f(0)+f'(0)\cdot a(\mathcal{U})$ for every S and every sample a. Therefore $\operatorname{Var}[f(\tilde{a}(S))] \leq (f'(0))^2 \cdot \operatorname{Var}[\tilde{a}(\mathcal{U})] \leq (f'(0))^2 \cdot (n\max_i \operatorname{Var}[\tilde{a}_i] + n^2 \max_{i,j}\operatorname{Cov}[\tilde{a}_i,\tilde{a}_j])$. Notice that $\operatorname{Cov}[\tilde{a}_i,\tilde{a}_j] \leq \sqrt{\operatorname{Var}[\tilde{a}_i]}\sqrt{\operatorname{Var}[\tilde{a}_j]}$, therefore $\sigma^2 = O(n^2)$ by our assumption that $\max_i \operatorname{Var}[\tilde{a}_i]$ is upper bounded by a constant. We enumerate all sets of size $K \geq e^{\alpha'}$ in Algorithm 3, and for each set of size K, extending it greedily needs time polynomial in n and N. Therefore the total running time is $\mathbf{poly}\left(2^{e^{\alpha'}}, n, \frac{1}{\epsilon}, \log\frac{1}{\gamma}\right)$. Combining Theorem 6 and Theorem 3, the approximation ratio follows. □

2.4 Power utility functions

In this section, we consider a particular class of concave utility function: power utility function. Let the concave utility function in (4) be $f(t) = t^p, 0 < p < 1$. It is also called isoelastic function for utility or constant relative risk aversion (CRRA) utility function in the economics literature. We show that we can calculate an approximation ratio that only depends on the exponent p and the budget B.

2.4.1 Sample average approximation

For power utility function $f = t^p$, the following problem is a special case of (7).

$$\max_{S\subseteq\mathcal{U}}\left\{F(S) = \sum_{r=1}^{N} f(a_r(S)) \mid b(S) \leq B\right\}, \tag{19}$$

where $a_r = (a_{r1}, \ldots, a_{rn}) \in \mathbb{R}_+^n$. Let S^* be an optimal solution in the following.

We aim to show the following theorem, where the approximation ratio no longer depending on the optimal solution.

Theorem 12. *For the power utility function $f(t) = t^p$ where p is a constant, given $\gamma > 0$, with probability at least $1 - 2\gamma$, Algorithm 3 runs in time* **poly** $\left(n, \frac{1}{\epsilon}, \log \frac{1}{\gamma}\right)$ *and solves the stochastic problem* (4) *with approximation ratio* $\left(1 - e^{-\alpha(B,p)}\right)(1 - \epsilon) - \epsilon$.

First we give a brief overview of the proof of Theorem 12 before proceeding to the technique details. Recall that set S^1 is the first partial solution constructed by the greedy algorithm that is not contained in S^*, *ie.*, $S^1 \backslash S^* \neq \emptyset$. We first give a lower bound of the ratio between $F(S^* \cup S^1) - F(S^1)$ and $F(S^*) - F(S^1)$ in terms of p and B. Then for power utility functions, we can write the approximation ratio of the greedy algorithm as a function of p and B. Then we upper bound K, which is the size of enumeration in Algorithm 3, by a function only depending on p so that Algorithm 3 runs in polynomial time when p is a constant. We assume that $B > 1$ since all data is integral and $B = 1$ is trivial.

To bound the ratio between $F(S^* \cup S^1) - F(S^1)$ and $F(S^*) - F(S^1)$ in terms of p and B, we first need the following simple lemma.

Lemma 13. *Let S^0 be the partial solution just before S^1. We have*

$$F(S^1) - F(S^0) \geq \frac{1}{B}(F(S^*) - F(S^0)). \tag{20}$$

Proof. Let $c_1 = \frac{F(S^1) - F(S^0)}{b_{i_1}}$. By (9), we have

$$F(S^*) - F(S^0) \leq \sum_{j \in S^* \backslash S^0} F(S^0 \cup \{j\}) - F(S^0) = \sum_{j \in S^* \backslash S^0} b_j \frac{F(S^0 \cup \{j\}) - F(S^0)}{b_j}.$$

By the greedy algorithm,

$$c_1 \geq \max \left\{ \frac{F(S^0 \cup \{j\}) - F(S^0)}{b_j} \mid j \in S^* \backslash S^0 \right\},$$

therefore

$$F(S^*) - F(S^0) \leq c_1 \sum_{j \in S^* \backslash S^0} b_j \leq c_1 B.$$

Replace c_1 by $(F(S^1) - F(S^0))/b_{i_1}$, we have

$$F(S^1) - F(S^0) \geq \frac{b_{i_1}}{B}(F(S^*) - F(S^0)) \geq \frac{1}{B}(F(S^*) - F(S^0)),$$

where the last inequality comes from the fact that $b_i \in \mathbb{N}_+$. □

Second, we need the following function to be increasing.

Proposition 14.

$$f(x) = \frac{\left(1 + \left(\frac{1}{B} + \frac{B-1}{B}x\right)^{1/p} - x^{1/p}\right)^p - \frac{1}{B} - \frac{B-1}{B}x}{1 - \frac{1}{B} - \frac{B-1}{B}x},$$

where $B \geq 1, 0 < p < 1, 0 \leq x < 1$ *is increasing in* x.

Proof. To prove $f(x)$ is increasing, we will show its derivative $f'(x) \geq 0$ for $0 \leq x < 1$. First let us simplify the notation and define

$$f_1(x) = 1 + \left(x + \frac{1}{B}(1-x)\right)^{1/p} - x^{1/p}$$
$$f_2(x) = 1 - x - \frac{1}{B}(1-x).$$

Then

$$f(x) = \frac{f_1(x)^p - 1}{f_2(x)} + 1,$$

and its derivative

$$f'(x) = \frac{f_1(x)^{p-1} f_3(x) f_2(x) + f_1(x)^p \frac{B-1}{B} - \frac{B-1}{B}}{f_2^2(x)},$$

where

$$f_3(x) = p f_1'(x) = \frac{B-1}{B}\left(\frac{1}{B} + \frac{B-1}{B}x\right)^{1/p-1} - x^{1/p-1}.$$

To show $f'(x) \geq 0$, we will show $\min_{0 \leq x < 1} f'(x) \geq 0$. First notice $f'(0) > 0$. Now we prove this is also true for $0 < x < 1$. Consider the function

$$f_4(x) = f_1(x)^{p-1} f_3(x) f_2(x) + f_1(x)^p \frac{B-1}{B}, 0 < x < 1.$$

First consider the case where $p \geq 1/2$. It suffices to show that $f_4'(x) < 0$. Indeed, since $f_4(x)$ is continuous and $\lim_{x \to 1} f_4(x) = (B-1)/B$, we know $f_4(x) > (B-1)/B$ and hence $f'(x) > 0$.

Now let us prove $f_4'(x) < 0$. The derivative of $f_4(x)$ is

$$f_4'(x) = -\frac{(B-1)(1-p)(1-x) f_1(x)^{p-2} g(x)}{Bpx^2(1 + (B-1)x)^2},$$

where

$$g(x) = x^{1/p}(1+(B-1)x)^2 + \left(x + \frac{1}{B}(1-x)\right)^{1/p}\left(x^{1/p} - x^2(B-1)^2\right).$$

Notice that all the terms except $g(x)$ is positive, so the sign of $f_4'(x)$ is the same as the sign of $-g(x)$. First multiply $g(x)$ by $B^{1/p}$.

$$B^{1/p}g(x) = (Bx)^{1/p}(1+(B-1)x)^2 + (Bx+1-x)^{1/p}\left(x^{1/p} - x^2(B-1)^2\right).$$

Dividing it by $x^{1/p}(Bx+1-x)^{1/p}$ will not change the sign when $0 < x < 1$. We now have a new function

$$\begin{aligned} g_1(x) &= B^{1/p}(Bx+1-x)^{2-1/p} - x^{2-1/p}(B-1)^2 + 1 \\ &= B^2\left(\frac{Bx+1-x}{B}\right)^{2-1/p} - (B-1)^2x^{2-1/p} + 1 \end{aligned}$$

Since $2 - 1/p \geq 0$, $\left(\frac{Bx+1-x}{B}\right)^{2-1/p} \geq x^{2-1/p}$ and $B^2 > (B-1)^2$, we have $g_1(x) > 0$ and $f_4'(x) < 0$.

Now consider the case $0 < p < 1/2$. First we show $g_1''(x) < 0$.

$$\begin{aligned} g_1''(x) = {} & B^{1/p}(Bx-x+1)^{-1/p}(B-1)^2(2-1/p)(1-1/p) \\ & - x^{-1/p}(B-1)^2(2-1/p)(1-1/p) \end{aligned}$$

Now $g_1''(x) < 0$ if and only if $B^{1/p}(Bx-x+1)^{-1/p} - x^{-1/p} < 0$, which is true since $(Bx-x+1)/B > x$. Then consider

$$g_1'(x) = (B-1)B^{1/p}(2-1/p)(1-x+Bx)^{1-1/p} - (B-1)^2(2-1/p)x^{1-1/p}.$$

It is continuous, decreasing, and $g_1'(0) > 0$ and $g_1'(1) < 0$ Therefore $g_1(x)$ is first increasing when $0 < x \leq x_0$ for some $0 < x_0 < 1$, then increasing for $x_0 < x < 1$, and $g_1(0) < 0, g_1(1) > 0$, we have $f_4'(x) > 0, 0 < x \leq x_0$ and $f_4'(x) < 0, x_0 < x < 1$. Therefore $f_4(x)$ is greater than $\min\{\lim_{x\to 0} f_4(0), \lim_{x\to 1} f_4(1)\} = (B-1)/B$ and therefore again $f'(x) > 0$. □

Now we are ready to bound the ratio between $F(S^* \cup S^1) - F(S^1)$ and $F(S^*) - F(S^1)$.

Lemma 15. *Given a problem* (19) *with utility function* $f(t) = t^p$ *and a knapsack constraint* B, $F(S^* \cup S^1) - F(S^1) \geq \alpha(B,p)(F(S^*) - F(S^1))$, *where the function* $\alpha(B,p) := \frac{(B^{1/p}+1)^p - 1}{B-1}$.

Proof. We first define a quantity that is a lower bound of the ratio. For $1 \leq r \leq N$, let $x_r = a_r(S^1 \backslash S^*), y_r = a_r(S^1 \cap S^*) = a_r(S^0)$ and $A_r = a_r(S^*)$ so that $(A_r + x_r)^p = f(a_r(S^* \cup S^1), (x_r + y_r)^p = f(a_r(S_1)), A_r^p = f(a_r(S^*))$. Let $x = (x_1, \ldots, x_N), y = (y_1, \ldots, y_N)$. Define

$$h(x,y) = \frac{\sum_{r=1}^N (A_r + x_r)^p - \sum_{r=1}^N (x_r + y_r)^p}{\sum_{r=1}^N A_r^p - \sum_{r=1}^N (x_r + y_r)^p}, (x,y) \in X$$

where

$$X = \left\{ (x,y) \mid \sum_{r=1}^N (x_r + y_r)^p - \sum_{r=1}^N y_r^p \geq \frac{1}{B} \left(\sum_{r=1}^N A_r^p - \sum_{r=1}^N y_r^p \right) \right\}. \tag{21}$$

By (20), we know that the ratio is at least $\min \{h(x,y) \mid (x,y) \in X\}$. Now we calculate $\min \{h(x,y) \mid (x,y) \in X\}$ to show

$$\min \{h(x,y) \mid (x,y) \in X\} = \alpha(B,p) = \frac{(B^{1/p}+1)^p - 1}{B-1}.$$

Since $\sum_r A_r^p = F(S^*)$ and $S^0 \subset S^*$, we may assume the minimum of $h(x,y)$ reaches when $\sum_r y_r^p = \lambda \sum_r A_r^p$ for $0 \leq \lambda < 1$ because otherwise S^0 would be a better solution than S^*. By (21), we have

$$\sum_r (x_r + y_r)^p \geq \frac{1}{B} \sum_r A_r^p + \frac{B-1}{B} \lambda \sum_r A_r^p.$$

We first argue that the minimum must be attained at equality. Assume not. Let (x,y) be a minimum solution such that $\sum_r (x_r + y_r)^p > \frac{1}{B} \sum_r A_r^p + \frac{B-1}{B} \lambda \sum_r A_r^p$. It then must be the case that there exist some $x_{r'} > 0$. Otherwise $\sum_r (x_r + y_r)^p = \sum_r y_r^p > \frac{1}{B} \sum_r A_r^p + \frac{B-1}{B} \lambda \sum_r A_r^p > \lambda \sum_r A_r^p = \sum_r y_r^p$, where the last inequality comes from that fact that $\lambda < 1$. Now decrease $x_{r'}$ (maybe for multiple r's) until we have a solution $(\tilde{x}, y)$ that reaches equality. First notice that $\sum_r (\tilde{x}_r + y_r)^p < \sum_r (x_i + y_i)^p$ and $\sum_r (A_r + \tilde{x}_r)^p < \sum_r (A_r + x_r)^p$. Therefore

$$\begin{aligned} h(\tilde{x}, y) &= \frac{\sum_{r=1}^N (A_r + \tilde{x}_r)^p - \sum_{r=1}^N (\tilde{x}_r + y_r)^p}{\sum_{r=1}^N A_r^p - \sum_{r=1}^N (\tilde{x}_r + y_r)^p} \\ &< \frac{\sum_{r=1}^N (A_r + x_r)^p - \sum_{r=1}^N (\tilde{x}_r + y_r)^p}{\sum_{r=1}^N A_r^p - \sum_{r=1}^N (\tilde{x}_r + y_r)^p} \\ &< \frac{\sum_{r=1}^N (A_r + x_r)^p - \sum_{r=1}^N (x_r + y_r)^p}{\sum_{r=1}^N A_r^p - \sum_{r=1}^N (x_r + y_r)^p}, \end{aligned}$$

which contradicts the assumption $h(x, y)$ is the minimum.

Now given that $\sum_r y_r^p = \lambda \sum_r A_r^p$ and $\sum_r (x_r + y_r)^p = \frac{1}{B} \sum_r A_r^p + \frac{B-1}{B} \lambda \sum_r A_r^p$, minimizing $h(x, y)$ is the same as minimizing $\sum_r (A_r + x_r)^p$. We use the method of Lagrange multipliers to minimize $\sum_r (A_r + x_r)^p$. In particular, we minimize the following function

$$\Lambda(x, y, \mu_1, \mu_2) = \sum_r (A_r + x_r)^p - \mu_1 \left(\sum_r (x_r + y_r)^p - \frac{1}{B} \sum_r A_r^p - \frac{B-1}{B} \lambda \sum_r A_r^p \right) - \mu_2 \left(\sum_r y_r^p - \lambda \sum_r A_r^p \right)$$

Take the partial derivative for each x_r and y_r. We have the following condition for the minimum solution.

$$p(A_r + x_r)^{p-1} - \mu_1 p (x_r + y_r)^{p-1} = 0$$

$$-\mu_1 p (x_r + y_r)^{p-1} - \mu_2 p y_r^{p-1} = 0$$

So

$$\frac{x_r + y_r}{A_r + x_r} = (\mu_1)^{p-1}, \quad \frac{y_i}{x_r + y_r} = \left(-\frac{\mu_1}{\mu_2} \right)^{p-1},$$

from which we may conclude that the ratios x_r/A_r are the same for every r, and it is also true for every y_r/A_r. Let $\lambda_1 = x_r/A_r$ and $\lambda_2 = y_r/A_r$. From the equation $\sum_r y_r^p = \lambda \sum_r A_r^p$, we solve $\lambda_2 = \lambda^{1/p}$. Then by the equation $\sum_r (x_r + y_r)^p = \frac{1}{B} \sum_r A_r^p + \frac{B-1}{B} \lambda \sum_r A_r^p$, we solve

$$\lambda_1 = \left(\frac{1}{B} + \frac{B-1}{B} \lambda \right)^{1/p} - \lambda^{1/p}.$$

Now we have

$$\sum_r (A_r + x_r)^p = \sum_r \left(1 + \left(\frac{1}{B} + \frac{B-1}{B} \lambda \right)^{1/p} - \lambda^{1/p} \right)^p A_r^p,$$

and

$$\min h(x, y) = \frac{\left(1 + \left(\frac{1}{B} + \frac{B-1}{B} \lambda \right)^{1/p} - \lambda^{1/p} \right)^p - \frac{1}{B} - \frac{B-1}{B} \lambda}{1 - \frac{1}{B} - \frac{B-1}{B} \lambda}.$$

By Proposition 14, we know that the above reaches minimum when $\lambda = 0$. By setting $\lambda = 0$, we have

$$\frac{F(S^* \cup S^1) - F(S^1)}{F(S^*) - F(S^1)} \geq \frac{(B^{1/p} + 1)^p - 1}{B - 1}.$$

□

Finally we need the following property to prove the theorem of this section.

Lemma 16. *The function $\alpha(B,p)$ is decreasing in B when $B > 1$*

Proof. The derivative

$$f'(B) = \frac{B - \left(1 + B^{1/p}\right)^{p-1}\left(B + B^{1/p}\right)}{B(B-1)^2}.$$

So $f'(B) < 0$ if and only if

$$B - \left(1 + B^{1/p}\right)^{p-1}\left(B + B^{1/p}\right) < 0.$$

Divide it by $B\left(1 + B^{1/p}\right)^{p-1}$, and we have

$$g(B) = \left(1 + B^{1/p}\right)^{1-p} - \left(1 + B^{1/p-1}\right).$$

Take the derivative of $g(B)$, we have

$$\begin{aligned} g'(B) &= B^{1/p-2}(1-1/p) - (1-1/p)B^{1/p-1}\left(1 + B^{1/p}\right)^{-p} \\ &= (1-1/p)B^{-1+1/p}\left(\frac{1}{B} - \frac{1}{\left(1 + B^{1/p}\right)^p}\right) < 0 \end{aligned}$$

where the last inequality comes from the facts $B < \left(1 + B^{1/p}\right)^p$ and $1 - 1/p < 0$. Therefore $g(B)$ is decreasing and $g(B) \le g(1) = 2^{1-p} - 2 < 0$. So $f'(B) < 0$ when $B \ge 1$. □

Proof of Theorem 12. By Lemma 16, we know that for $B > 1$ and integral, $\alpha(B,p)$ is at most $\alpha(2,p)$. Set $K = \left\lceil e^{\alpha(2,p)} \right\rceil$ and the algorithm runs in time $\mathbf{poly}\left(n, \frac{1}{\epsilon}, \log\frac{1}{\gamma}\right)$ by Corollary 11. By a close examination of the proof of inequality (14), we know that α can be replaced by the ratio $\frac{F(S^* \cup S^1) - F(S^1)}{F(S^*) - F(S^1)}$. By Lemma 15, the ratio can be lower bounded by $\alpha(B,p)$. Thus the approximation ratio of Algorithm 3 on (19) is $1 - e^{-\alpha(B,p)}$ by Theorem 6. The approximation of the stochastic problem (4) then follows from Corollary 11. □

2.4.2 Fixed number of scenarios

In this section, we consider the case where there is only fixed number of scenarios, and each scenario happens with a known probability. In particular, we focus on the following problem:

$$\max_{S \subseteq \mathcal{U}} \left\{ F(S) = \mathrm{E}\left[f(\tilde{a}(S))\right] = \sum_{i=1}^{k} q_i f(a_i(S)) \mid b(S) \le B \right\}. \tag{22}$$

Here we have k realizations of $\tilde{a}$, k is a constant, and each realization a_i happens with probability q_i. We assume that $a_i \in \mathbb{N}_+^n, 1 \leq i \leq k$. The following problem is a slight generalization of (22):

$$\max_{S \subseteq \mathcal{U}} \left\{ F(S) = \sum_{i=1}^{k} g_i(a_i(S)) \mid b(S) \leq B \right\}, \tag{23}$$

where each g_i is positively homogeneous with degree p. That is, $g_i(Rt) = R^p g_i(t)$. Notice that $q_i f(t)$ is positive homogeneous with degree p when $f(t) = t^p$. Denote $w_i = \max_j \{a_{i1}, \ldots, a_{ij}, \ldots, a_{in}\}$ and let w be the least common multiple of $w_1, \ldots, w_n$. By positive homogeneity,

$$g_i(a_i(S)) = g_i\left(\frac{w_i}{w}\frac{w}{w_i}a_i(S)\right) = \left(\frac{w_i}{w}\right)^p g_i\left(\frac{w}{w_i}a_i(S)\right),$$

we convert each a_i to $\frac{w}{w_i}a_i$ and use $(\frac{w_i}{w})^p g_i(\cdot)$ as the new g_i. Now the largest component in vector $\frac{w}{w_i}a_i$ is w for every i and $\frac{w}{w_i}a_i \in \mathbb{N}_+^n$. Therefore we will only discuss the case where every a_i has the same largest component in the following.

We first describe a pseudo-polynomial time algorithm using dynamic programming. Later we will use the usual rounding technique to convert it into a polynomial time algorithm. The idea largely follows the FPTAS that solves the classic knapsack problem. Let M be the state table in our dynamic program. For a state $(m, x_1, \ldots, x_k)$ in the table, we want to find a set $S \subseteq \{1, \ldots, m\}, m \leq n$ such that $a_i(S) = x_i, 1 \leq i \leq k$ and the weight $b(S)$ is minimized. Let the value of this state $M(m, x_1, \ldots, x_k)$ be $b(S)$. We set $M(m, x_1, \ldots, x_k) = \infty$ if there is no feasible subset of $\{1, \ldots, m\}$ to attain $a_i(S) = x_i$ for every i. For a given state $M(m+1, x_1, \ldots, x_k)$, we can either find a subset $S \subseteq \{1, \ldots, m\}$ so that $a_i(S) = x_i, 1 \leq i \leq k$, or we use item $m+1$ and a subset $S' \subseteq \{1, \ldots, m\}$ so that $a_i(S' \cup \{m+1\}) = x_i, 1 \leq i \leq k$. Initially every entry is marked as ∞. Formally, we use

the following recursion to calculate $M(m+1, x_1, \ldots, x_k)$.

$$M(m+1, x_1, \ldots, x_k) =$$
$$\begin{cases} \min\{M(m, x_1, \ldots, x_k), b_{m+1} + M(m, x_1 - a_{1m+1}, \ldots, x_k - a_{km+1})\} \\ \qquad\qquad \text{if } a_{im+1} \leq x_i, 1 \leq i \leq k. \\ M(m, x_1, \ldots, x_k) \qquad \text{otherwise.} \end{cases}$$

By induction, we can see that for each achievable state, the dynamic programming above will find a subset of $\mathcal{U}$.

Since each $x_i, 1 \leq i \leq k$ is upper bounded by nw and $a_i \in \mathbb{N}_+^n, 1 \leq i \leq k$, we have at most $O(n^{k+1}w^k)$ entries of $M(m, x_1, \ldots, x_k)$, where each entry maps to a value of $F(S)$. After the whole state table has been calculated, for each entry $M(n, x_1, \ldots, x_k)$, if it is not marked as infinity, we calculate its corresponding $F(S)$. The entry with the largest $F(S)$ is an optimal solution of the problem. The time complexity of the algorithm is $O(n^{k+1}w^k)$.

Notice that w can be very large and the algorithm above is not polynomial-time. We claim that by ignoring some insignificant bits in each a_i through scaling, the optimal solution we find by the dynamic programming above in the scaled version is very close to the optimal solution of the original problem. In particular, let ϵ' be a constant to be determined later, $R = \frac{\epsilon' w}{n}$, and $a'_{ij} = \left\lfloor \frac{a_{ij}}{R} \right\rfloor, \forall i, j$. We run the algorithm described above on the scaled version. That is, we run the dynamic programming above on the following problem.

$$\max_{S \subseteq \mathcal{U}} \left\{ F'(S) = \sum_i g_i(a'_i(S)) \mid a'_{ij} = \left\lfloor \frac{a_{ij}}{R} \right\rfloor, b(S) \leq B \right\}.$$

Let S be the optimal solution of this new problem found by the dynamic programming and S^* be an optimal solution of the original problem. The following theorem shows that for any given $\epsilon > 0$, we can set ϵ' accordingly and find a solution that is close enough to S^*.

Theorem 17. *Given an $\epsilon > 0$, by setting $\epsilon' = \left(\frac{\epsilon}{k}\right)^{1/p}$, we have an algorithm with running time $O\left(\frac{n^{2k+1}}{\epsilon^k}\right)$ that produces a solution S such that $F(S) \geq (1-\epsilon)F(S^*)$.*

Proof. Let S be the solution found by the algorithm and S^* be an optimal solution of the

problem.

$$
\begin{aligned}
F(S) &= g_1(a_1(S)) + \cdots + g_k(a_k(S)) \\
&\geq g_1(Ra'_1(S)) + \cdots + g_k(Ra'_k(S)) && \text{(rounding down, } g_i \text{ is nondecreasing)} \\
&= R^p \left(g_1(a'_1(S)) + \cdots g_k(a'_k(S))\right) && \text{(Positive homogeneity)} \\
&\geq R^p \left(g_1(a'_1(S^*)) + \cdots + g_k(a'_k(S^*))\right) && \text{(Optimality of } S\text{)} \\
&= g_1(Ra'_1(S^*)) + \cdots + g_k(Ra'_k(S^*)) && \text{(Positive homogeneity)} \\
&\geq g_1(a_1(S^*) - nR) + \cdots + g_k(a_k(S^*) - nR) && \text{(each coordinate loses at most } R\text{)} \\
&\geq g_1(a_1(S^*)) - g_1(nR) + \cdots + g_k(a_k(S^*)) - g_k(nR) && \text{(subadditivity)} \\
&= F(S^*) - \sum_i g_i(nR) \\
&= F(S^*) - \sum_i g_i(\epsilon' w) && (nR = \epsilon' w) \\
&= F(S^*) - \sum_i \epsilon'^p g_i(w) && \text{(Positive homogeneity)}
\end{aligned}
$$

Let ϵ be the approximation ratio we want to achieve. Since $\epsilon' = \left(\frac{\epsilon}{k}\right)^{1/p}$, the term $\sum_i \epsilon'^p g_i(w)$ equals to $\frac{1}{k}\epsilon \sum_i g_i(w)$, which is no greater than $\epsilon F(S^*)$, as $\sum_i g_i(w) \leq F(S^*)$.

The running time of the algorithm is

$$
O\left(n^{k+1} \left\lfloor \frac{w}{R} \right\rfloor^k\right) = O\left(n^{k+1} \left\lfloor \frac{n}{\epsilon} \right\rfloor^k\right) = O\left(\frac{n^{2k+1}}{\epsilon^k}\right). \qquad \square
$$

2.5 Conclusion

In this chapter, we present an algorithm that solves a special class of submodular function with knapsack constraint. The approximation ratio of the algorithm is better than the classic $(1 - 1/e)$ ratio by exploiting the strictly increasing property of the function and a careful analysis of the algorithm. We apply the approximation algorithm and sample average approximation technique to solve the expected utility knapsack problem with constant relative error. We also give a fully polynomial time approximation scheme (FPTAS) when the number of scenarios in the expectation is fixed.

The works by Conforti and Cornuéjols [9], and Vondràk [52] are similar to ours. The differences are that they consider only cardinality and matroid constraint and they measure

how fast a submodular function increases by the curvature of the function. The curvature and the corresponding approximation ratio achieved in [9, 52] are not directly comparable with α defined in (11) and the approximation ratio of our algorithm. We leave it as an open question whether Algorithm 3 (or any its variation) can achieve the same approximation as the one in [9, 52].

CHAPTER III

MAXIMIZING EXPECTED UTILITY OVER A KNAPSACK CONSTRAINT: POLYHEDRAL RESULTS

3.1 Introduction

In expected utility theory [41, 50], utility function serves to model the risk preferences of decision makers. Concave utility functions model risk-averse preferences where certainty of the outcome is valued higher. Contrary to the continuous setting in previous works [41, 50], we focus on decision sets that are discrete choices such as decisions of investing in bond, stock, and mutual fund. Let $v_i \in \mathbb{R}^N$ be the value vector for the investments under scenario i with probability $\pi_i, i = 1 \ldots, m$. Maximizing expected utility for a risk-averse investor can be formulated as

$$\max \left\{ \sum_{i=1}^{m} \pi_i f(v_i' x) \mid x \in X \subseteq \{0,1\}^N \right\}, \tag{24}$$

where f is a concave, increasing utility function and X is the set of feasible solutions. A canonical example of utility function is $f(t) = 1 - \exp(-t/\lambda)$ where $\lambda > 0$ is the parameter of risk tolerance, with smaller λ representing greater risk aversion.

In this chapter, we tackle Problem (24) by developing a mixed integer linear programming (MILP) based approach. We consider the following reformulation of (24):

$$\max \left\{ \sum_{i=1}^{m} \pi_i w_i \mid x \in X \subseteq \{0,1\}^n, (w_i, x) \in \operatorname{conv}(Q_i), \forall i = 1, \ldots, m \right\},$$

where each set Q_i for $i = 1, \ldots, m$ is of the form

$$Q = \left\{ x \in \{0,1\}^N, w \in \mathbb{R} \mid w \leq f(a'x + d)) \right\}$$

where $a \in \mathbb{R}^N$ and $d \in \mathbb{R}$. Since Q is the union of a finite set of half lines with a common recession direction, its convex hull $\operatorname{conv}(Q)$ is a polyhedron. If the vector a is nonnegative or nonpositive, the function $g(x) := f(a'x)$ is submodular over $\{0,1\}^N$. Recall that a function of binary variables $g : \{0,1\}^N \to \mathbb{R}$ is submodular if and only if for all $x, y \in \{0,1\}^n$ with

$x \leq y$ and for some $i \in N$ with $x_i = y_i = 0$, we have $g(x + e^i) - g(x) \geq g(y + e^i) - g(y)$, where e^i is the i-th unit vector. Nemhauser and Wolsey [35] show the mixed-integer nonlinear programming formulation of Q can be written as a mixed-integer linear program. In [1], Ahmed and Atamtürk give strong valid inequalities of conv(Q) by lifting inequalities developed by Nemhauser and Wolsey in [35] . In this chapter we improve such a formulation by incorporating information from the constraint $x \in X$. In particular, we consider the case when X is a knapsack constraint and study the following mixed-integer set

$$P = \left\{ x \in \{0,1\}^N, w \in \mathbb{R} \mid w \leq f(a'x + d), b'x \leq B \right\}, \tag{25}$$

where f is a strictly concave, increasing, differentiable function, $a, b \in \mathbb{R}^N_+$, $b \in \mathbb{R}^N_+$ and $d \in \mathbb{R}$.

The contributions of this chapter are as follows. We give a family of valid inequalities for conv(P). Specifically, starting from a valid inequality for a restriction of P, we obtain a lifting function that can be solved in polynomial time. The lifting function is then approximated by a subadditive one by dropping the integrality constraint on the solution and replacing the knapsack constraint with a cardinality constraint. We give a polynomial time algorithm to compute this subadditive lifting function. As a by-product, we give a partial characterization of the optimal solution of a continuous concave optimization problem, which we believe is interesting in its own right. We demonstrate the effectiveness of the proposed inequalities in a branch-and-cut framework to solve the expected utility problem. We develop several techniques to reduce redundant computation during cut generation. Compared with the cuts developed in [1], our proposed cuts significantly reduces CPU time of optimization and also reduces the number of nodes explored and the number of cuts added in the branch-and-cut process.

Submodular maximization has been extensively studied in recent years since its application in online auction. Because it is **NP**-hard in general case, many works [36, 46, 12, 51, 30] have been devoted to develop various kinds of algorithms to find good approximation solutions in polynomial time. This chapter tackles the problem through a mathematical programming perspective. We build effective valid inequalities of conv(P) so that an exact

optimal solution can be found fast by a MIP solver through a branch-and-cut framework. Our work is a direct extension of the work by Ahmed and Atamtürk [1], where they give valid inequalities of conv(Q). Atamtürk and Narayanan [5] study the submodular knapsack problem which extends the classic linear knapsack constraint to a submodular one. They extend the cover inequalities for knapsack problem and investigate the lifting problem. While they study a similar problem through the mathematical programming perspective, the setting is fundamentally different from ours as they have a fixed budget in the submodular constraint while we have it as a variable.

The rest of the chapter is organized as follows. In Section 3.2, we first discuss the submodular inequalities in [35] and derive a family of valid inequalities by lifting. We present a lifting problem which is a submodular maximization problem with a cardinality constraint. This extends the lifting problem in [1] and we show there is a simple polynomial time algorithm to solve the lifting problem. Then we give a polynomial time algorithm to solve the continuous relaxation of the lifting problem, which leads to the desired subadditive lifting inequalities. In Section 3.3, we present results of computational experiments showing the effectives of our new lifted inequalities.

3.2 Valid inequalties by lifting

In this section, we study the valid inequalities of the convex hull of the set (25):

$$P = \left\{ x \in \{0,1\}^N, w \in \mathbb{R} \mid w \leq f(a'x + d), b'x \leq B \right\},$$

where f is a strict concave, increasing, differentiable function, $a, b \in \mathbb{R}^N_+$, $d \in \mathbb{R}$ and B is a positive real number. The following notation will be used throughout. Let $N = \{1, \ldots, n\}$. For a vector $v \in \mathbb{R}^N$, let $v(S) = \sum_{i \in S} v_i$. Let $\rho_i(S) = f(a(S) + a_i) - f(a(S))$.

The unconstrained version of P is set

$$Q = \left\{ x \in \{0,1\}^N, w \in \mathbb{R} \mid w \leq f(a'x + d) \right\}.$$

Since $f(a'x + d)$ is submodular over $\{0,1\}^n$, it has been shown in [35, p. 710] that the

following two exponential families of inequalities

$$w \leq f(a(S)) - \sum_{i \in S} \rho_i(N \backslash i)(1 - x_i) + \sum_{i \in N \backslash S} \rho_i(S) x_i, \quad \forall S \subseteq N \tag{26}$$

$$w \leq f(a(S)) - \sum_{i \in S} \rho_i(S \backslash i)(1 - x_i) + \sum_{i \in N \backslash S} \rho_i(\emptyset) x_i, \quad \forall S \subseteq N \tag{27}$$

can be used as a mixed-integer linear formulation of Q. That is

$$Q = \left\{ x \in \{0,1\}^N, w \in \mathbb{R} \mid (26) \text{ or } (27) \right\}.$$

Ahmed and Atamtürk already show in [1] that such a formulation is rather ineffective and they develop strong lifted inequalities for $\text{conv}(Q)$. Inspired by their work, we consider inequalities for the constrained set $\text{conv}(P)$ in this chapter.

3.2.1 Uplifting

Given a set $S \subseteq N$, consider the restriction of P by setting $x_i = 0$ for all $i \in N \backslash S$:

$$P(N \backslash S, \emptyset) = \left\{ x \in \{0,1\}^S, w \in \mathbb{R} \mid f(\sum_{i \in S} a_i x_i) \geq w, \quad \sum_{i \in S} b_i x_i \leq B \right\}.$$

It follows from (27) that the following inequality

$$w \leq f(a(S)) - \sum_{i \in S} \rho_i(S \backslash i)(1 - x_i) \tag{28}$$

is valid for $P(N \backslash S, \emptyset)$. Furthermore it is facet-defining for $\text{conv}(P(N \backslash S, \emptyset))$ if $b(S) \leq B$. To lift variable x_i, $i \in N \backslash S$ into (28), consider the following lifting function

$$\begin{aligned} \zeta_1(\delta, \beta) := \max \quad & w + \sum_{i \in S} \rho_i(S \backslash i)(1 - x_i) - f(a(S)) \\ \text{s.t.} \quad & f(\sum_{i \in S} a_i x_i + \delta) \geq w \\ & \sum_{i \in S} b_i x_i + \beta \leq B \\ & x_i \in \{0,1\}, \forall i \in S. \end{aligned} \tag{29}$$

Problem (29) is hard because of the knapsack constraint, therefore we solve a relaxation with a cardinality constraint that is derived from the knapsack constraint. Toward this end, we define cardinality

$$k(\beta, S) = \max \left\{ |T| \mid \sum_{i \in T} b_i + \beta \leq B, T \subseteq S \right\}.$$

To calculate $k(\beta, S)$, we first sort b_i so that $b_1 \leq \cdots \leq b_{|S|}$, then $k(\beta, S)$ is the largest index i such that $b_1 + \ldots + b_i \leq B - \beta$.

Given

$$k = \max \{k(b_i, S) \mid i \in N \backslash S\},$$

then a relaxation of (29) is

$$\begin{aligned} \zeta_2(\delta, k) := \max \quad & w + \sum_{i \in S} \rho_i(S \backslash i)(1 - x_i) - f(a(S)) \\ \text{s.t.} \quad & f(\sum_i a_i x_i + \delta) \geq w \\ & \sum_{i \in S} x_i \leq k \\ & x_i \in \{0, 1\}, \forall i \in S. \end{aligned} \tag{30}$$

We now discuss some properties of the solutions of (30). We use the following notation throughout the discussion. For any two integers i_1, i_2, denote $A(i_1 : i_2) = \sum_{i=i_1}^{i_2} a_i$. Let $l = |S|$, and we sort a_i for $i \in S$ so that $a_1 \geq \cdots \geq a_l$. The *support* of a solution x of Problem (30) is the set $\{i \mid x_i = 1\}$. Some properties may have been implicitly shown in [1], nevertheless we give the proof here for the sake of completeness.

Lemma 18. *For any nonempty set $T \subseteq S$, consider an item $i_1 \in T$ and an item $i_2 \in S \backslash T$. If they satisfy either one of the following conditions:*

1. *$a_{i_1} \leq a_{i_2}$ and $a(T) + \delta \leq a(S) - a_{i_2}$,*
2. *$a_{i_1} \geq a_{i_2}$ and $a(T) + \delta \geq a(S) - a_{i_2}$,*

then the objective value of the solution with support $T \backslash \{i_1\} \cup \{i_2\}$ is no worse than that of the one with T.

Proof. For case 1, we have difference of objective values as

$$\begin{aligned} & f(a(T) + a_{i_2} - a_{i_1} + \delta) - \rho_{i_2}(S \backslash i_2) - f(a(T) + \delta) + \rho_{i_1}(S \backslash i_1) \\ =& f(a(T) + a_{i_2} - a_{i_1} + \delta) - f(a(T) + \delta) - (f(a(S) - a_{i_1}) - f(a(S) - a_{i_2})) \end{aligned}$$

Consider $a(T)+\delta$ and $a(S)-a_{i_2}$. If both terms increase by amount $a_{i_2}-a_{i_1}$, which is nonnegative, then the smaller one of these two terms will have a larger increase of function value by the concavity of f. Since $a(T)+\delta \le a(S)-a_{i_2}$, we have

$$f(a(T)+a_{i_2}-a_{i_1}+\delta)-f(a(T)+\delta)-(f(a(S)-a_{i_1})-f(a(S)-a_{i_2}))\ge 0.$$

For case 2, we have difference of objective values as

$$\begin{aligned}&f(a(T)+\delta+a_{i_2}-a_{i_1})-\rho_{i_2}(S\backslash i_2)-f(a(T)+\delta)+\rho_{i_1}(S\backslash i_1)\\ =&f(a(T)+\delta-a_{i_1}+a_{i_2})-f(a(T)+\delta)-(f(a(S)-a_{i_1})-f(a(S)-a_{i_2}))\ge 0,\end{aligned}$$

since $a_{i_2}\le a_{i_1}, a(T)+\delta\ge a(S)-a_{i_2}$, and f is concave. □

Lemma 19. *If there is no cardinality constraint in Problem* (30) *(i.e.* $k\ge|S|$*), then we have the following properties:*

- *If* $\delta\ge A(1:l)-A(j:l)$*, then there exists a solution with support of size* $l-j$ *no worse than any solution with support of larger size.*
- *If* $\delta<A(1:l)-A(j+1:l)$*, then there exists a solution with support of size* $l-j$ *no worse than any solution with support of smaller size.*
- *If* $A(1:l)-A(j:l)\le\delta<A(1:l)-A(j+1:l)$*, then there exists an optimal solution with support* $\{j+1,\dots,l\}$.

Proof. When $\delta\ge A(1:l)-A(j:l)$, for any solution T of size larger than $l-j$, if we remove an item i from T, the objective function will not be worse. In particular, the difference of the objective function is

$$\begin{aligned}&f(a(T)+\delta-a_i)+\rho_i(S\backslash i)-f(a(T)+\delta)\\ =&f(a(S))-f(a(S-a_i))-(f(a(T)+\delta)-f(a(T)+\delta-a_i))\ge 0,\end{aligned}$$

since $a_i\ge 0$, $a(S)-a_i\le A(j:l)+\delta-a_i\le a(T)+\delta-a_i$, and f is concave. This process can continue until T is of size $l-j$.

When $\delta < A(1:l) - A(j+1:l)$, for any solution T of size smaller than $l-j$, there are two cases to consider. (1) If $\delta + a(T) \leq A(1:l) - a_i$ for some $i \notin T$, adding the item i to T will not worsen the solution, because the difference of the objective function is

$$\begin{aligned} & f(a(T) + a_i + \delta) - \rho_i(S \backslash i) - f(a(T) + \delta) \\ =& f(a(T) + a_i + \delta) - f(a(T) + \delta) - (f(a(S)) - f(a(S - a_i)) \geq 0, \end{aligned}$$

since $a_i \geq 0, a(T) + \delta \leq a(S) - a_i$ and f is concave. (2) $\delta + a(T) > A(1:l) - a_i$ for all $i \notin T$. Then we say there must exist $i_1 \in T \backslash \{j+1, \ldots, l\}, i_2 \in \{j+1 \ldots, l\} \backslash T$ such that $a_{i_1} \geq a_{i_2}$. Otherwise since size of T is smaller than $l-j$, $T \subset \{j+1, \ldots, l\}$, and then $a(T) + a_{i_2} \leq A(j+1:l)$. In addition, since $\delta < A(1:l) - A(j+1:l)$, we have $\delta < A(1:l) - a(T) - a_{i_2}$, which is a contradiction to the assumption. Then we say the value of the solution $T \backslash \{i_1\} \cup \{i_2\}$ will not be worse because of case 2 of Lemma 18. Such swap can continue until T contains the smallest $|T|$ items among $\{a_1, \ldots, a_l\}$. Then there exists $i \in \{j+1, \ldots, n\}$ such that $a(T) + a_i \leq A(j+1:l)$. Therefore $\delta + a(T) \leq a(S) - a_i$ by the assumption $\delta < A(1:l) - A(j+1:l)$. Then we go to case 1 until T reaches size of $l-j$.

When $A(1:l) - A(j:l) \leq \delta < A(1:l) - A(j:l)$, we already prove that a set of size $l-j$ is optimal. Now consider a solution T of size $l-j$ but it is not $\{j+1, \ldots, l\}$. Let $i_1 \in T \backslash \{j+1, \ldots, l\}, i_2 \in \{j+1, \ldots, l\} \backslash T$. Solution $T \backslash \{i_1\} \cup \{i_2\}$ will not be worse by case 2 of Lemma 18. □

Lemma 20. *If $\delta < A(1:l) - A(l-k+1:l)$, then let*

$$j = \operatorname{argmax} \left\{ A(1:l) - A(j':j'+k) \mid \delta \geq A(1:l) - A(j':j+k), 1 \leq j' \leq l-k \right\}.$$

If such a j exists, then a solution with support $\{j+1, \ldots, j+k\}$ is optimal for (30)*. Otherwise $\delta < A(1:l) - A(1:k+1)$, then a solution with support $\{1, \ldots, k\}$ is optimal for* (30)*.*

Proof. First by Lemma 19, for $A(1:l) - A(l-k:l) \leq \delta < A(1:l) - A(l-k+1:l)$, the support $\{l-k+1, \ldots, l\}$ is optimal.

Also by Lemma 19, we know that for any $\delta < A(1:l) - A(l-k+1:l)$, there exist a set T of size k that is better than any set of size less than k. In the following, we show that for

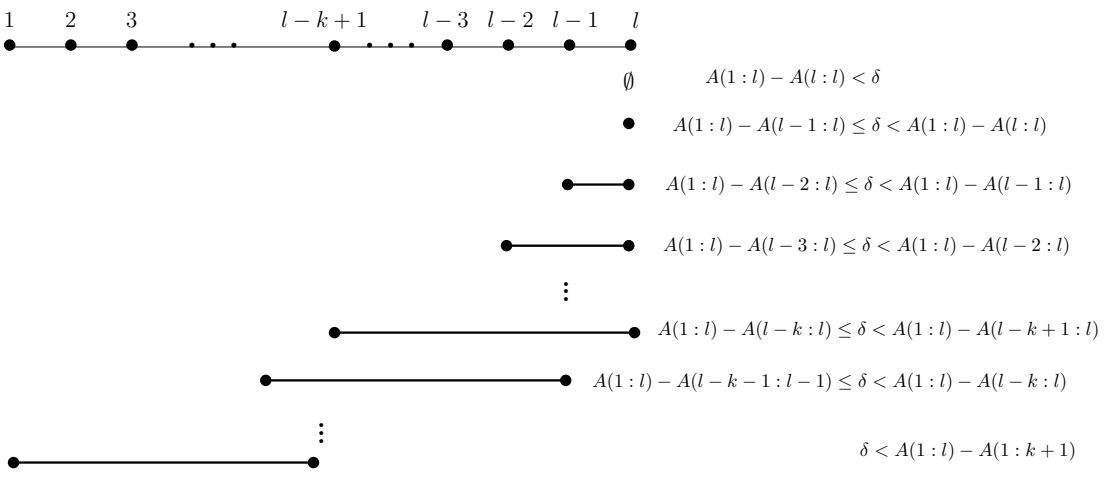


Figure 4: Solutions of (30) for different δ

$\delta < A(1:l) - A(l-k:l)$, if we start from set T of size k not as desired, we may transform it to the desired one without decreasing the objective value.

If $\delta < A(1:l) - A(1:k+1)$ and $T \neq \{1,\ldots,k\}$. Then there must exist an $i_1 \in T, i_1 > k$ and $i_2 \notin T, i_2 \leq k$ such that $a_{i_2} \geq a_{i_1}$ and $\delta < A(1:l) - a(T) - a_{i_1}$. We claim that $T\setminus\{i_1\}\cup\{i_2\}$ is no worse than T by case 1 of Lemma 18. Such swap can go on until $T = \{1,\ldots,k\}$ as desired.

If $A(1:l) - A(j:j+k) \leq \delta < A(1:l) - A(j+1:j+k+1)$ and $T \neq \{j+1,\ldots,j+k\}$, we have several cases to consider.

1. $a_1,\ldots,a_j \notin T$, then again we can always find $i_1 \in T, i_1 > j+k$ and $i_2 \notin T, i_2 \in \{j+1,\ldots,j+k\}$. To see that $a(T)+\delta \geq a(S) - a_{i_2}$, we notice that $a(T)+a_{i_2} \leq A(j+1:j+k+1)$ since $a_1,\ldots,a_j \notin T$ and $A(1:l) - A(j+1:j+k+1) > \delta$. Therefore we know that $T\setminus\{i_1\}\cup\{i_2\}$ is no worse than T by case 1 of Lemma 18 and such swap can continue until $T = \{j+1,\ldots,j+k\}$.

2. There exists at least one $i_1 \leq j, i_1 \in T$. Pick an $i_2 \notin T, i_2 \in \{j+1,\ldots,j+k\}$. There are two cases to consider.

   (a) $\delta \geq A(1:l) - a(T) - a_{i_2}$. Then we replace i_1 by i_2. Such swap will not worsen the solution by case 2 of Lemma 18. After replacing all $i_1 \leq j, i_1 \in T$, if the

solution still does not equal to $\{j+1,\ldots,j+k\}$, we are in case 1.

(b) $\delta < A(1:l)-a(T)-a_{i_2}$. Then we claim that there exists $i_3 \in \{j+k+1,\ldots,l\}\cap T$. Otherwise $a(T)+a_{i_2} \geq A(j:j+k)$ and then since $\delta < A(1:l)-a(T)-a_{i_2}$ we may conclude $\delta < A(1:l)-A(j:j+k)$ which is a contradiction. Therefore we may replace i_3 by i_2 and not worsen the solution because of case 1 of Lemma 18. Such swap cannot go on forever and eventually we will reach case 2a. □

Lemma 19 and Lemma 20 imply Algorithm 4. Given the input δ, the algorithm searches an interval that δ belongs to. If $\delta > A(1:l)-A(l-k:l)$, the result set T comes from a chain of sets. Otherwise, T consists k consecutive items of $\{1,\ldots,l\}$. We illustrate the solutions of (30) for different δ in Figure 4.

Algorithm 4: Greedy Algorithm for solving (30)

Result: A set $T \subseteq S$ so that $x_i = 1, \forall i \in T$ and $x_i = 0$ otherwise.
$l \leftarrow |S|$;
$k \leftarrow k(S,\beta)$;
Sort a_i so that $a_1 \geq \cdots \geq a_l$;
$T = \{l-k+1,\ldots,l\}$;
$j \leftarrow l-k+1$;
if $\delta \geq a(S) - A(l-k+1:l)$ **then**
  while $j \leq l$ *and* $a(S)-a(T) < \delta$ **do**
    $T \leftarrow T\setminus\{j\}$;
    $j \leftarrow j+1$;
  end
else
  while $j > 1$ *and* $a(S)-a(T)-a_{j-1} > \delta$ **do**
    $T \leftarrow T\setminus\{j+l-1\}\cup\{j-1\}$;
    $j \leftarrow j-1$;
  end
end

Proposition 21. *Algorithm 4 solves Problem* (30) *in* $O(|S|)$ *time.*

Proof. By Lemma 19 and Lemma 20, there are $|S|+1$ possible solutions of Problem (30). Each solution corresponds to an interval that δ may belong to. These $|S|+1$ intervals are disjoint and can be ordered linearly. Algorithm 4 searches the intervals in this linear order to determine where δ belongs to. For each interval, the algorithm takes constant number of operations. Thus Algorithm 4 solves Problem (30) in $O(|S|)$ time. □

3.2.2 Subadditive approximation

In order to have sequence independent lifting, we need a subadditive lifting function [54, 21]. To achieve that, we consider the continuous relaxation of Problem (30). Define

$$\begin{aligned} \gamma(\delta, k) := \max \quad & w + \sum_{i \in S} \rho_i(S \backslash i)(1 - x_i) - f(a(S)) \\ \text{s.t.} \quad & f(\sum_{i \in S} a_i x_i + \delta) \geq w \\ & \sum_{i \in S} x_i \leq k \\ & 0 \leq x_i \leq 1, \forall i \in S, \end{aligned} \tag{31}$$

and we will have a subadditive lifting function by choosing k in the following way.

Lemma 22. *Let* $k_1 = \max\{k(b_i, S) \mid i \in N \backslash S\}$, $k_2 = \min\{k \mid \gamma(0, k) \geq 0\}$, *and* $k_0 = \max\{k_1, k_2\}$. *Then* $\gamma(\delta, k_0)$ *is a subadditive lifting function.*

Proof. We first show $\gamma(\delta, k_0)$ is a valid lifting function for all $i \in N \backslash S$ by proving that $\gamma(\delta, k_0)$ is larger than $\zeta_1(\delta, b_i)$ for all $i \in N \backslash S$. To see this, notice $\gamma(\delta, k_0) \geq \gamma(\delta, k_1)$ and $\gamma(\delta, k_1)$ is a continuous relaxation of the lifting function $\zeta_2(\delta, k_1)$. Therefore we have $\gamma(\delta, k_0) \geq \zeta_1(\delta, b_i)$ for every $i \in N \backslash S$.

Now we show $\gamma(\delta, k_0)$ is a subadditive function in δ. First $\gamma(\delta, k_0)$ is concave in δ, because for each δ, it is the maximum of a concave function of δ over a convex set $\left\{x \mid \sum_{i \in S} x_i \leq k_0, 0 \leq x_i \leq 1\right\}$. Then since $\gamma(0, k_2) \geq 0$ and $k_0 \geq k_2$, we know that $\gamma(0, k_0) \geq 0$. Then by [24, p. 239], we know that a concave function $\gamma : \mathbb{R}_+ \to \mathbb{R}$ is subadditive if and only if $\gamma(0) \geq 0$. Therefore $\gamma(\delta, k_0)$ is a subadditive function in δ. □

Since $\gamma(\delta, k_0)$ is subadditive in δ, now we lift variables in $N \backslash S$ in a sequence independent order [54, 21] and have a family of valid inequalities.

Theorem 23. *For any set* $S \subseteq N$, *the following inequality*

$$w \leq f(a(S)) - \sum_{i \in S} \rho_i(S \backslash i)(1 - x_i) + \sum_{i \in N \backslash S} \gamma(a_i, k_0) x_i \tag{32}$$

is valid for P.

Example 24. We give an example comparing inequality (32) and the lifted inequality in [1]. Consider the model $P = \{(w, x) \in \mathbb{R} \times \{0, 1\}^n \mid w \leq -\exp(-a'x), b'x \leq B\}$. Let $n = 6$, $B = 1$ and $S = \{1, 2, 3, 4\}$. Let

$$a = (0.3008, 0.3621, 0.4233, 0.6395, 0.1164, 0.0448),$$

and

$$b = (0.3023, 0.1892, 0.3884, 0.1047, 0.5938, 6699)$$

be two vectors where each component is generated from a uniform random distribution $[0, 1]$. We calculate $k_0 = 3$ in (32). The lifted inequality in [1], which is the same as (32) except $k_0 = |S| = 4$, is

$$w \leq -0.5714 + 0.06251x_1 + 0.0777x_2 + 0.0938x_3 + 0.1594x_4 + 0.0417x_5 + 0.0238x_6;$$

Inequality (32), which is tighter than the above at coefficients of x_5, x_6, is

$$w \leq -0.5714 + 0.06251x_1 + 0.0777x_2 + 0.0938x_3 + 0.1594x_4 + \mathbf{0.0374}x_5 + \mathbf{0.0169}x_6.$$

3.2.3 Computing the subadditive lifting function

To compute the lifting function $\gamma(\delta, k_0)$ fast, we need to solve problem (31) in polynomial time. To achieve this, we first focus on the following more general problem

$$\max f(a'x + d) - c'x, ex \leq k, x \in [0, 1]^N, \tag{33}$$

where $a \in \mathbb{R}^N_+, c \in \mathbb{R}^N$, e is the all-one vector, and $f : \mathbb{R} \to \mathbb{R}$ is strictly concave. We will show that we may find an optimal solution of (33) if it is fractional (at least one component is in $(0, 1)$) and together with Proposition 21, we may compute $\gamma(\delta, k_0)$ in polynomial time.

Remark. We do not assume that $a_i/c_i, i \in N$ are distinct, which is the assumption made in [1], because the reduction used to transform from non-distinct case to distinct case may break the cardinality constraint we impose here.

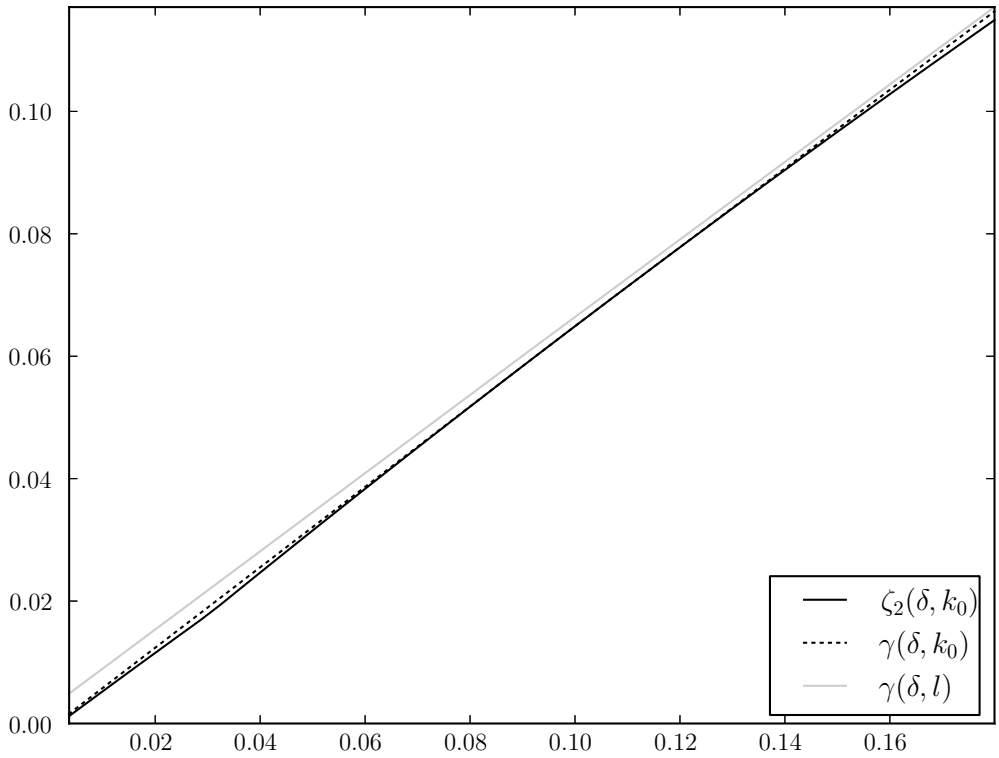


Figure 5: Comparison of lifting functions for $\delta < A(1:l) - A(l-k+1:l)$

Figure 5 is an example of differences among $\zeta_2(\delta, k_0), \gamma(\delta, k_0)$ and $\gamma(\delta, l)$ when $l = 5, k = 3$. Notice that since the number of variables in (31) is l, thus $\gamma(\delta, l)$ means that there is no cardinality constraint. Therefore $\gamma(\delta, l)$ is the subadditive lifting function used in [1]. For $\delta \geq A(1:l) - A(l-k_0+1:l)$, Problem (30) without cardinality constraint has an optimal solution of size no greater than k_0. This means that there is almost no difference between $\gamma(\delta, l)$ and $\gamma(\delta, k)$. Thus in Figure 5, we plot the functions in the range $\delta < A(1:l) - A(l-k_0+1:l)$.

First we write out the KKT conditions for (33).

$$a_i f'(a'x + d) - c_i = \lambda + \alpha_i - \beta_i \tag{34}$$

$$\lambda(\sum_{i \in N} x_i - k) = 0 \tag{35}$$

$$\alpha_i(x_i - 1) = 0 \tag{36}$$

$$\beta_i x_i = 0 \tag{37}$$

$$\lambda, \alpha_i, \beta_i \geq 0$$

$$\sum_{i \in N} x_i \leq k$$

$$0 \leq x_i \leq 1.$$

We rewrite (34) as the following

$$f'(a'x + d) = \frac{c_i + \lambda}{a_i} + \frac{\alpha_i - \beta_i}{a_i}. \tag{38}$$

Since f is strictly concave and increasing, the inverse function of f' exists. The above can also be written as

$$a'x + d = (f')^{-1}\left(\frac{c_i + \lambda}{a_i} + \frac{\alpha_i - \beta_i}{a_i}\right).$$

To simplify the notation, we define

$$h_i(\lambda) = \frac{c_i + \lambda}{a_i}.$$

From the KKT conditions, we have the following properties of the optimal solution.

Lemma 25. *For a component x_i with $\alpha_i = \beta_i = 0$, if $h_j(\lambda) < h_i(\lambda)$, then $x_j = 1$; if $h_j(\lambda) > h_i(\lambda)$, then $x_j = 0$.*

Proof. For any other index $j \neq i$, by (38) we have

$$h_i(\lambda) = h_j(\lambda) + \frac{\alpha_j - \beta_j}{a_j}.$$

Therefore if $h_j(\lambda) < h_i(\lambda)$, $\frac{\alpha_j - \beta_j}{a_j} = h_i(\lambda) - h_j(\lambda) > 0$. Since $a_j > 0$, we must have $\alpha_j > 0$ and $x_j = 1$ by condition (36). Similarly, if $h_j(\lambda) > h_i(\lambda)$, $\frac{\alpha_j - \beta_j}{a_j} = h_i(\lambda) - h_j(\lambda) < 0$. Since $a_j > 0$, we must have $\beta_j > 0$ and $x_j = 0$ by condition (37). □

Lemma 26. *For a fractional optimal solution, if its dual variable $\lambda > 0$, then there must exist at least two different indices i, j such that $h_i(\lambda) = h_j(\lambda)$.*

Proof. Consider a fractional component x_i and suppose that there is no index j such that $h_i(\lambda) = h_j(\lambda)$. By condition (36) and (37), $\alpha_i = \beta_i = 0$. By Lemma 25, for any j with $h_j(\lambda) < h_i(\lambda)$, $x_j = 1$; for any j with $h_j(\lambda) > h_i(\lambda)$, $x_j = 0$. Thus x_i is the only fractional component and so $\sum_i x_i < k$. If $\lambda > 0$, it then violates condition (35). □

Before proceeding to the algorithm for solving Problem (33), we need the following lemmas for solving two linear systems.

Lemma 27. *For $a_1 \geq a_2 \geq \cdots \geq a_n$, $h \geq 0$, and an integer $k \leq n$, the linear system*

$$\begin{aligned} a_1x_1 + \cdots + a_nx_n &= h \\ x_1 + \cdots + x_n &\leq k \\ 0 \leq x_i \leq 1, \quad \forall i \in \{1, \ldots, n\} & \end{aligned}$$

has a solution if and only if $a_1 + \cdots a_k \geq h$. Moreover,

$$x_i = \frac{h}{a_1 + \cdots + a_k}, \forall i \leq k; \quad x_i = 0, \forall i > k,$$

is a solution.

Proof. If $a_1 + \cdots + a_k \geq h$, then $x_i = \frac{h}{a_1+\cdots+a_k} \in [0, 1]$ for $i \leq k$ and $\sum_{i \leq k} x_i \leq k$. So the solution is feasible.

If $a_1 + \cdots + a_k < h$, then for any $x = (x_1, \ldots, x_n) \in [0, 1]^n$ with $\sum_i x_i \leq k$, we have

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq a_1 + \cdots + a_k < h.$$

That is, no solution can satisfy the condition $a_1x_1 + \cdots + a_nx_n = h$. □

Lemma 28. *For $a_1 \geq a_2 \geq \cdots \geq a_n$, $h \geq 0$, and an integer $k \leq n$, the linear system*

$$\begin{aligned} a_1x_1 + \cdots + a_nx_n &= h \\ x_1 + \cdots + x_n &= k \\ 0 \leq x_i \leq 1 \quad \forall i \in \{1, \ldots, n\} & \end{aligned}$$

has a solution if and only if $a_1 + \cdots a_k \geq h$ and $a_{n-k+1} + \cdots + a_n \leq h$. Moreover, a solution can be computed in linear time.

Proof. If $a_1 + \cdots a_k \geq h$ and $a_{n-k+1} + \cdots + a_n \leq h$, we first solve the following linear system:

$$\begin{aligned} (a_1 + \cdots + a_k)t_1 + (a_{n-k+1} + \cdots + a_n)t_2 &= h \\ t_1 + t_2 &= 1. \end{aligned}$$

After getting the solution $t_1, t_2 \in [0, 1]$, we set

$$x_i = \begin{cases} t_1 & i \in \{1, \ldots, k\} \setminus \{n - k + 1, n\} \\ t_1 + t_2 = 1 & i \in \{1, \ldots, k\} \cap \{n - k + 1, n\} \\ t_2 & i \in \{n - k + 1, n\} \setminus \{1, \ldots, k\} \\ 0 & \text{otherwise.} \end{cases}$$

To verify $x_1 + \cdots + x_n = k$, we check

$$\begin{aligned} & x_1 + \cdots + x_n \\ =& \sum_{i \in \{1,\ldots,k\} \setminus \{n-k+1,n\}} t_1 + \sum_{i \in \{1,\ldots,k\} \cap \{n-k+1,n\}} (t_1 + t_2) + \sum_{i \in \{n-k+1,n\} \setminus \{1,\ldots,k\}} t_2 \\ =& kt_1 + kt_2 = k. \end{aligned}$$

To verify $a_1x_1 + \cdots + a_nx_n = h$, we check

$$\begin{aligned} & a_1x_1 + \cdots + a_nx_n \\ =& \sum_{i \in \{1,\ldots,k\} \setminus \{n-k+1,n\}} a_it_1 + \sum_{i \in \{1,\ldots,k\} \cap \{n-k+1,n\}} a_i(t_1 + t_2) + \sum_{i \in \{n-k+1,n\} \setminus \{1,\ldots,k\}} a_it_2 \\ =& (a_1 + \cdots a_k)t_1 + (a_{n-k+1} + \cdots + a_n)t_2 = h. \end{aligned}$$

We now prove the other direction. When $\sum_i x_i = k$, notice that

$$a_1 + \cdots + a_k \geq a_1x_1 + \cdots a_nx_n \geq a_{n-k+1} + \cdots + a_n.$$

Therefore if either $a_1 + \cdots + a_k < h$ or $a_{n-k+1} + \cdots a_n > h$, we cannot satisfy the condition $a_1x_1 + \cdots a_nx_n = h$. □

To find a fractional optimal solution of Problem (33), We define

$$\Lambda = \left\{ \lambda \geq 0 \mid \lambda = \frac{a_jc_i - a_ic_j}{a_i - a_j}, \forall i, j \in N \right\} \cup \{0\}$$

as the set of λs that can be part of an optimal dual solution. Then for each $\lambda \in \Lambda$, let (λ, h, I) be a tuple such that $I \subseteq N$ and for any index $i \in I$, $h_i(\lambda) = h$. Notice here for a λ there might exist multiple corresponding tuples. By Lemma 26, if $\lambda > 0$, then $|I| \geq 2$. Let $\mathcal{C}$ be the collection of these tuples. In Algorithm 5, for each tuple (λ, h, I), we assume that all the fractional components are among the indices set I, then we use Lemma 25 to fix values of variables whose indices not in I. We then solve a linear system using Lemma 27 or Lemma 28 to see if there exists a solution to satisfy condition (38). If the linear system has a solution, then we find a fractional optimal solution.

Proposition 29. *If Problem* (33) *has a fractional optimal solution, Algorithm 5 finds it in polynomial time.*

Algorithm 5: Find a fractional optimal solution of Problem (33)

Compute the tuple collection $\mathcal{C} = \{(\lambda, h, H)\}$;
for $(\lambda, h, I) \in \mathcal{C}$ **do**
  $T \leftarrow \{j \mid h_j(\lambda) < h\}, x_j = 1, \forall j \in T$;
  $T' \leftarrow \{j \mid h_j(\lambda) > h\}, x_j = 0, \forall j \in T'$;
  if $\lambda = 0$ **then**
    Solve $\sum_{i \in I} a_i x_i + a(T) + d) = (f')^{-1}(h), \sum_{i \in I} x_i \leq k - |T|, x_i \in [0, 1]$;
  else
    Solve $\sum_{i \in I} a_i x_i + a(T) + d) = (f')^{-1}(h), \sum_{i \in I} x_i = k - |T|, x_i \in [0, 1]$;
  end
  If the linear system has a solution, **return** x;
end
// Problem (33) has no fractional optimal solution.

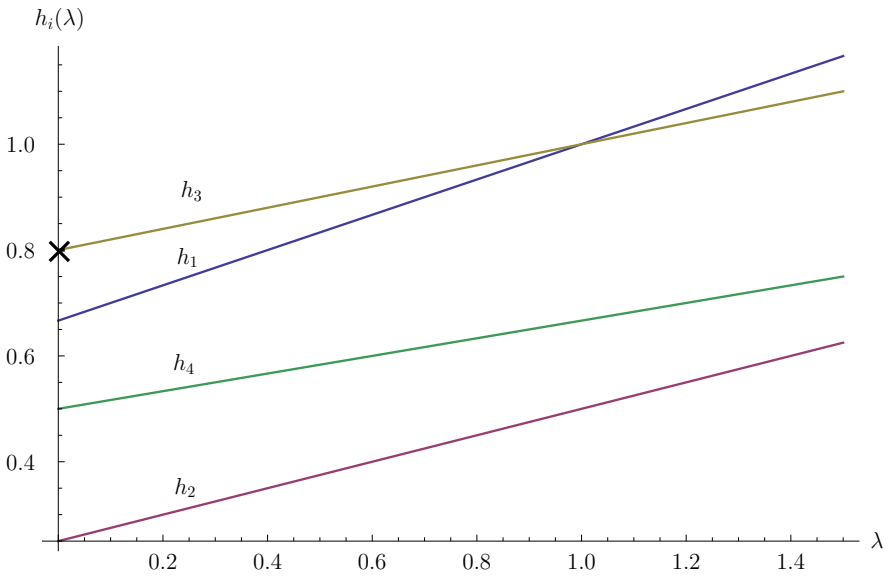
Proof. First the set Λ includes all possible λs by Lemma 26. Now consider a tuple (λ, h, I) that the algorithm chooses to construct a fractional solution. First by Lemma 27 and Lemma 28, we know that if the linear system to solve has a solution, we will find it correctly. Then we show that we can construct a dual solution so that the primal and dual solutions satisfy the KKT conditions. For $i \in I$, we set $\alpha_i = \beta_i = 0$, therefore we satisfy conditions (38), (36), and (37). For any $i \notin I$, the value of its primal and dual variables are determined by Lemma 25. Finally if $\lambda > 0$, we use the constraint $\sum_{i \in I} x_i = k - |T|$ in the linear system; otherwise we use the one with $\sum_{i \in I} x_i \leq k - |T|$ so condition (35) always is met. Therefore the primal solution found by Algorithm 5 is optimal.

For the time complexity, computing the tuple collection $\mathcal{C}$ takes $O(n^2)$ time since the size of Λ is $O(n^2)$. For each tuple, solving the linear system takes $O(n)$ time by Lemma 27 and Lemma 28. Therefore the total time of Algorithm 5 is polynomial. □

Corollary 30. *Problem* (31) *can be solved in polynomial time.*

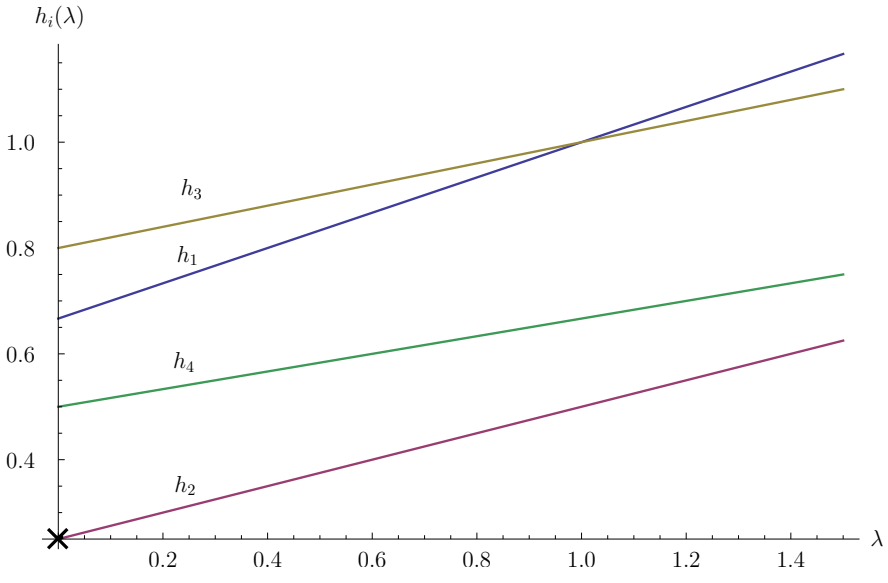
Proof. First we use Algorithm 5 to search for a fractional optimal solution. If it fails to find a λ that satisfies the KKT condition, then the optimal solution of Problem (31) must be an integral one. Therefore we use Algorithm 4 to obtain to solve the problem. The time complexity easily follows from Proposition 21 and Proposition 29. □

Corollary 31. *For a set S, the lifted inequality* (32) *can be computed in polynomial time.*



(a) Infeasible solution

If x_3 is a fractional component, then by Lemma 25, $x_1 = x_2 = x_4 = 1$. Then the constraint $\sum_i x_i \leq k$ is not satisfied.



(b) Optimal solution

If x_2 is a fractional component, then by Lemma 25, $x_3 = x_1 = x_2 = 0$. We check that $f'(ax + d) = h_2(\lambda) = 0.25$ and every other KKT conditions are met. So this is an optimal solution.

Figure 6: An example of searching optimal λ in Algorithm 5 when $n = 4, k = 3$

The function $f(a'x) = -\exp(-a'x + d)$, $n = 4$, $k = 3$, and $d = 1$. The vector $a = (3, 4, 5, 6), c = (2, 1, 4, 3)$. We start from the tuple $(0, h_3(0), \{3\})$ in Algorithm 5, and we find an optimal solution at tuple $(0, h_2(\lambda), \{2\})$.

Proof. To compute the lifting function $\gamma(\delta, k_0)$ in (32), first we need to find k_0 in Lemma 22 to ensure the subadditivity of lifting function $\gamma(\delta, k_0)$ as a function of δ. By Lemma 22, $k_0 = \max(k_1, k_2)$. k_1 is easy to calculate and we now discuss how to reduce time of calculating k_2. To find k_2, a simple approach would be to evaluate $\gamma(0, k)$ starting from $k = 1$ until $\gamma(0, k) \geq 0$. Notice here $\gamma(0, k) \leq \gamma(0, k')$ if $k < k'$. Therefore we use binary search for k over $\{1, \ldots, |S|\}$ and reduce the times to evaluate $\gamma(0, k)$ from $O(|S|)$ to $O(\log(|S|))$. Each evaluation will take polynomial time by Corollary 30. Then we we need to evaluate $\gamma(a_j, k_0)$ for every $i \in N \backslash S$. For each of them, we need to solve an instance of Problem (31). Notice that for all $\gamma(a_i, k_0)$, the only thing changed in Problem (31) is the value δ. Therefore given a seed inequality, we first compute the collection $\mathcal{C}$ in Algorithm 5, then for each $a_i, i \in N \backslash S$, we just look up the precomputed collection $\mathcal{C}$ instead of computing it every time. The total time to compute $\gamma(a_i, k_0)$ again is polynomial by Corollary 30. □

3.2.4 Downlifting

The subadditive lifting inequality for downlifting in [1] is

$$w \leq f(a(S)) + \sum_{i \in S} \omega(-a_i)(1 - x_i) + \sum_{i \in N \backslash S} \rho_i(S) x_i, \tag{39}$$

where $\omega(\cdot)$ is a subadditive lifting function. Notice if we add cardinality constraint on x as a constraint of the lifting function ω, it would decrease the value of $\omega(-a_i)$ and thus increases the value of coefficient of variable x_i. This defeats the purpose of tightening inequalities to have a better relaxation. Thus we do not consider downlifting with constraints in this chapter.

3.3 Computational experiments

3.3.1 Problem and its data generation

In this section, we evaluate the effectiveness of our inequalities (32) by solving a problem of expected utility maximization with capital budgeting, which is the benchmark used in [1]. The problem can be stated as following:

$$\max \left\{ \sum_{i=1}^{m} \pi_i \left(1 - \exp\left(-\frac{v_i' x}{\lambda} \right) \right) \mid a' x \leq 1, x \in \{0, 1\}^N \right\}.$$

Here for a set N of investment options, $a_j, j \in N$ are the capital requirements and we normalize the available budget to be 1. Each scenario $i, 1 \leq i \leq m$ happens with probability π_i, and the value of investments in the future under scenario i is denoted as $v_i \in \mathbb{R}_+^N$. The utility function is modeled as an exponential function $f(t) = 1 - \exp(t/\lambda)$ with risk tolerance parameter λ. We reformulate the problem into a MILP setting by introducing a w_i for each scenario and rewrite the problem as following:

$$1 + \max \left\{ \pi' w \mid a'x \leq 1, w_i \leq -\exp\left(-\frac{-v_i'x}{\lambda}\right), 1 \leq i \leq m, x \in \{0,1\}^N \right\}. \tag{40}$$

Then for each scenario i, the set $\left\{(w_i, x) \in \mathbb{R} \times \{0,1\}^n \mid a'x \leq 1, w_i \leq -\exp(-\frac{-v_i'x}{\lambda})\right\}$ is the form of (25).

We use exactly the same setting of [1] to generate problem instances. For capital requirements a_i, they are uniformly generated from $[0, 0.2]$. For investment valuation, we use the lognormal return distribution. In particular, the value of investment j under scenario i is

$$v_{ij} = r_{ij} a_j,$$

where

$$\ln r_{ij} = \alpha_j + \beta_j \ln f_i + \epsilon_{ij}, j \in N, i \in \{1, \ldots, m\}.$$

Here α_j is from uniform distribution $[0.05, 0.1]$, β_j is from uniform distribution $[0, 1]$, $\ln f_i$ is from normal distribution $N(0.05, 0.0025)$ and ϵ_{ij} is from normal distribution $N(0, 0.0025)$. Finally the scenarios are equally likely to occur with $\pi_i = 1/m$ for $i \in \{1, \ldots, m\}$.

We implement the computation, written in Python, using the MILP solver of Gurobi 5.6.3 on a 2.3 GHz x86 Linux workstation with 7GB memory restriction. Gurobi's internal cut parameters are in default setting. We disable multithreading, heuristics, and the concurrent MIP solver; and set the relative MIP optimality gap as 0.01% and the time limit of the computation to 30 minutes.

Since f is an exponential function, there is no algorithm in Gurobi that can solve the continuous relaxation of (25) directly. Nevertheless we build a mixed-integer linear program as a relaxation of (25) and let it be the initial model for Gurobi to solve. In particular, we

approximate $f(a'x)$ by its gradient at zero and add constraint $w_i \leq f(0) + f'(0)(a'x - 0)$ for each $i = 1, \ldots, m$ to the model.

During the branch-and-cut process, we need a seed inequality (28) to compute a lifted inequality (32) during the branch-and-cut process. Recall the seed inequality for uplifting is

$$w \leq f(a(S)) - \sum_{i \in S} \rho_i(S \backslash i)(1 - x_i).$$

To compute the seed inequality, we need a set S. Given a point $\overline{x}$, if it is integral, we use the support of x as the set S; otherwise, we use a heuristic approach to find a set S. Toward this, we approximate $\rho_i(S \backslash i)$ by $\rho_i(\emptyset)$, which follows [1], solve the following problem

$$\min \left\{ f(az) - \sum_{i \in N} \rho_i(\emptyset)(1 - \overline{x_i})z_i \mid z \in \{0,1\}^N, \sum_{i \in N} z_i \leq k(N, 0) \right\},$$

and use the support of the solution as the set S. To make the above problem solvable, here we replace the original knapsack constraint by a cardinality constraint where $k(N, 0) = \max\left\{|T| \mid \sum_{i \in T} b_i \leq B\right\}$. We use a parametric linear optimization algorithm proposed in [5] and find an optimal solution in $O(n^3)$ time. Now after knowing the seeding inequality, we apply Corollary 31 to calculate inequality (32).

For downlifting, the seed inequality for inequality (39) is

$$w \leq h(S) + \sum_{i \in N \backslash S} \rho_i(S) x_i.$$

For an integral $\overline{x}$, the approach is again to find the support of the solution as S. For a fractional one, similar to uplifting, we solve the following problem

$$\min \left\{ f(az) + \sum_{i \in N} \rho_i(\emptyset)\overline{x_i}(1 - z_i) \mid z \in \{0,1\}^N, bz \leq k(N, 0) \right\},$$

to find the desired S, which also takes $O(n^3)$ time. After knowing the seed inequality, we use the same approach as in [1] to compute an inequality by downlifting.

3.3.2 Experiments

We choose number of variables (n), scenarios (m) and risk tolerance factor λ as the parameters of our experiments, which is the same as the setting in [1]. Here we increase the

difficulty of the problems by increasing n, m or decreasing λ since most of the values of n, m, λ used in [1] are too easy for the current solver. We use our new uplifted inequality (32) and old downlifted inequality (39) from [1] to compare against the lifted inequalities developed in [1]. We do not consider the simplest inequalities (26) and (27) since they are too weak and do not result in termination of the branch-and-cut procedure in the desired period of time in most cases.

We present a summary of results in Table 2. For different n, m and λ, we report the CPU time spent in optimization in seconds (time), the number of branch-and-cut nodes explored (nodes), and the number of cuts added (cuts). Each row of the table presents average over twenty instances.

First, we observe from Table 2 that the number of variables n, scenarios m, and the risk tolerance factor λ all contribute to the overall performance. There is no single parameter dominating others. This is different from the observations made in [1] where λ dominates the other two factors. Second, we observe that by using our new uplifted inequalities the time spent in optimization decreases most, compared with the number of nodes and cuts. In particular, the average CPU time is 118 seconds when using inequalities developed in [1], while it is 80 seconds with our new uplifted inequality. The average number of nodes and cuts are 522, and 1173 for inequalities in [1], respectively; and they are 502, and 1136 for the new inequality. So the reduction in average CPU time is 32% and the reductions in the numbers of nodes and cuts are 4% and 6% respectively. Finally we remark that our initial linear approximation of constraint $w \leq f(a'x)$, replacing $f(a'x)$ with its gradient at zero, is useful. Compared with experiments only using trivial upper bounds $w \leq 0$ for (40) at the start, we observe a 33% reduction in the number of nodes and cuts in the linear approximation. Since it is not the main focus of this chapter, we do not report the details of this comparison.

In Figure 7, we present a performance profile of CPU time in . Following Dolan and Moré in [10], the performance profile is constructed as follows. We have a set $\mathcal{S}$ of two solvers using inequality (32), (39) and the lifted inequalities in [1], respectively. We denote the set of problem instances as $\mathcal{P}$. For a solver $s \in \mathcal{S}$ and a problem instance $p \in \mathcal{P}$, we

Table 2: Comparing lifted inequalities

n	m	λ	Lifted inequalities in [1]			Inequality (32) and (39)		
			Time	Node	Cuts	Time	Node	Cuts
100	50	2	15	115	274	12	114	267
		1	27	218	512	21	215	515
		0.8	29	266	558	23	264	576
	100	2	27	109	539	23	115	603
		1	53	257	1044	37	245	1020
		0.8	67	352	1361	46	332	1257
150	50	2	39	219	385	29	204	344
		1	103	576	908	61	548	837
		0.8	132	793	1268	86	749	1151
	100	2	48	137	541	89	135	531
		1	197	362	1593	85	336	1428
		0.8	215	532	1852	124	489	1784
200	50	2	73	375	465	54	368	465
		1	147	936	1149	100	879	1023
		0.8	182	1416	1601	143	1382	1638
	100	2	146	293	874	57	296	894
		1	254	907	2396	187	878	2308
		0.8	365	1528	3797	273	1489	3808

calculate its performance ratio $r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} \mid s \in \mathcal{S}\}}$ where $t_{p,s}$ is the time required by solver s on instance p. Figure 7 plots the cumulative distribution function of the performance ratio defined as $g_s(\tau) = \frac{1}{|\mathcal{P}|} |\{p \in \mathcal{P} \mid r_{p,s} \leq \tau\}|$. We see the performance of inequalities (32) and (39) is significantly better than the two set of lifted inequalities in [1].

3.4 Conclusion

In this chapter, we build a mixed-integer linear formulation of the mixed-integer non-linear program $P = \{(w, x) \in \mathbb{R} \times \{0, 1\}^n \mid w \geq f(a'x), b'x \leq B\}$. More specifically, we develop subadditive sequence independent lifting inequalities for $\text{conv}(P)$. The subadditive lifting function is computed by a greedy algorithm when the optimal solution is integral, and is computed by searching the Lagrange dual solution when the optimal is fractional. The latter is done by a partial characterization of the optimal solution of a continuous concave maximization problem. Computational experiments show that the proposed inequalities are better than these proposed in [1] where the knapsack constraint is not considered.

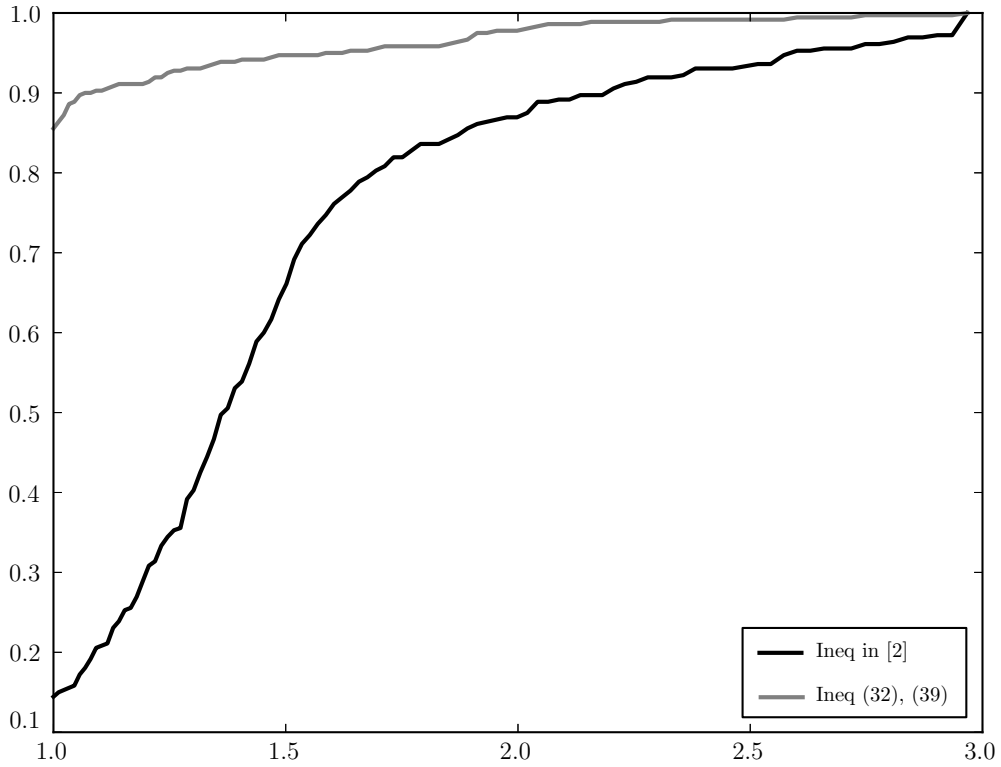


Figure 7: Performance profile for subadditive lifting function

It is an open question whether the subadditive lifting function (33) we proposed is the concave envelope of the original lifting function. Moreover, it would be interesting if another family of valid inequalities not relying on submodular inequalities (27) or (26) could be developed, even though we do not hope to get a polynomial time separation algorithm since a linear optimization problem over P is **NP**-hard [1].

CHAPTER IV

MINIMIZING A CLASS OF SUBMODULAR FUNCTIONS OVER A CARDINALITY CONSTRAINT: POLYHEDRAL RESULTS

4.1 Introduction

Optimization problems in various applications involving economies of scale or risk averse behavior can be formulated as concave cost combinatorial optimization problems of the form

$$\text{(CCO)}: \ \min\left\{\sum_{\ell=1}^{L} f_\ell(a_\ell' x): \ x \in X \subseteq \{0,1\}^n\right\},$$

where $f_\ell : \mathbb{R} \mapsto \mathbb{R}$ is concave and $a_\ell \in \mathbb{R}^n$ for $\ell = 1, \ldots, L$. The set X denotes combinatorial constraints, for example, those modeling feasible paths, assignments or knapsack solutions. Some specific examples of (CCO) are mentioned next.

Concave cost facility location: The concave cost facility location problem is an extension of the usual facility location problem to model economies of scale that can be achieved by connecting multiple customers to the same facility. With m customers and n facilities, the problem can be formulated as

$$\min\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j(\sum_{i=1}^{m} d_i x_{ij}): \ \sum_{j=1}^{n} x_{ij} = 1 \ \forall \ i, \ x_{ij} \in \{0,1\} \ \forall \ i,j\right\},$$

where c_{ij} is the cost of assigning customer i to facility j, d_i is the demand of customer i, and f_j is a nondecreasing concave cost function modeling the economies of scale of serving demand from facility j. This (CCO) problem generalizes the well-known fixed-charge facility location problem and has been studied in [14, 22, 40, 45].

Mean-risk combinatorial optimization: Consider a stochastic combinatorial optimization problem $\min\{\tilde{c}'x: \ x \in X \subseteq \{0,1\}^n\}$ where the cost vector has independently distributed components $\tilde{c}_i$ with mean μ_i and variance ν_i for $i = 1, \ldots, n$. A typical deterministic

equivalent formulation is to optimize a weighted combination of the mean and standard deviation of the cost, i.e.

$$\min\left\{\mu' x + \lambda\sqrt{\nu' x} :\ x \in X \subseteq \{0,1\}^n\right\},$$

where λ is a nonnegative weight. Such mean-risk combinatorial optimization problems in various settings have been studied in [4, 26, 37, 39].

Approximate submodular minimization: Various combinatorial optimization problems with nondecreasing submodular cost functions of the form

$$\min\{F(x) :\ x \in X \subseteq \{0,1\}^n\}$$

have been considered (cf. [6, 17, 19, 28]). Often the cost function F is only available through a value oracle. It has been shown [18] that a general nondecreasing submodular function F can be approximated up to a factor of $\sqrt{n}\log n$ by a function of the form $f(x) = \sqrt{c'x}$ using only a polynomial number of function value queries, i.e. $f(x) \leq F(x) \leq O(\sqrt{n}\log n) f(x)$ for all $x \in \{0,1\}^n$. Thus we can write an explicit formulation for an approximate version of the above submodular cost combinatorial optimization problem as the (CCO) problem:

$$\min\left\{\sqrt{c'x} :\ x \in X \subseteq \{0,1\}^n\right\}.$$

With a view towards developing mixed integer linear programming (MILP) based approaches for (CCO) we consider the following reformulation

$$\text{(CCO')}:\ \min\left\{\sum_{\ell=1}^{L} w_\ell :\ x \in X \subseteq \{0,1\}^n, (w_\ell, x) \in \text{conv}(\mathcal{Q}_\ell) \forall\ \ell = 1,\ldots,L\right\},$$

where each set $\mathcal{Q}_\ell$ for $\ell = 1,\ldots,L$ is of the form

$$\mathcal{Q} = \{(w,x) \in \mathbb{R} \times \{0,1\}^n :\ w \geq f(a'x)\}, \tag{41}$$

and f is a concave function. Note that since $\mathcal{Q}$ is the union of a finite set of half lines with a common recession direction, its convex hull, denoted by $\text{conv}(\mathcal{Q})$, is a polyhedron. When the vector a is nonnegative (or nonpositive) , the function $F(x) := f(a'x)$ is submodular over the binary hypercube. A function of binary variables $F : \{0,1\}^n \to \mathbb{R}$ is submodular

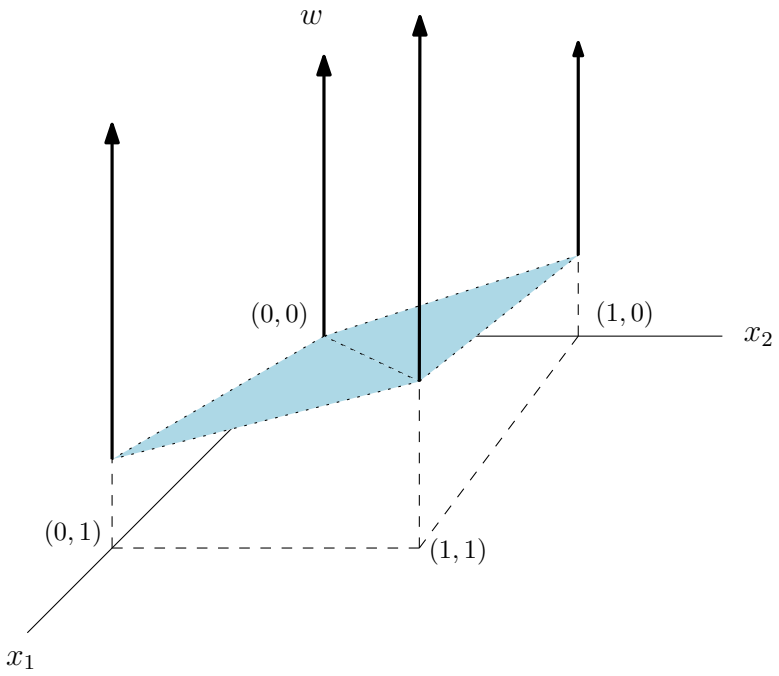
if for all $x, y \in \{0,1\}^n$ with $x \leq y$ (component-wise) and for some $i \in \{1,\ldots,n\}$ with $x_i = y_i = 0$, we have $F(x + e^i) - F(x) \geq F(y + e^i) - F(y)$, where $e^i \in \mathbb{R}^n$ is the i-th unit vector, i.e. the marginal values are diminishing [49]. By the classical results of Edmonds [11] and Lovasz [32] on submodular functions, we know an explicit inequality description of conv($\mathcal{Q}$). Thus (CCO') provides a mixed integer linear programming formulation of the mixed integer nonlinear program (CCO). In this chapter we improve such a formulation by incorporating information from the constraints $x \in X$ in the set $\mathcal{Q}$. In particular we study the following mixed-integer set

$$\mathcal{P} = \{(w, x) \in \mathbb{R} \times \{0,1\}^n : w \geq f(a'x), e'x \leq k\}, \tag{42}$$

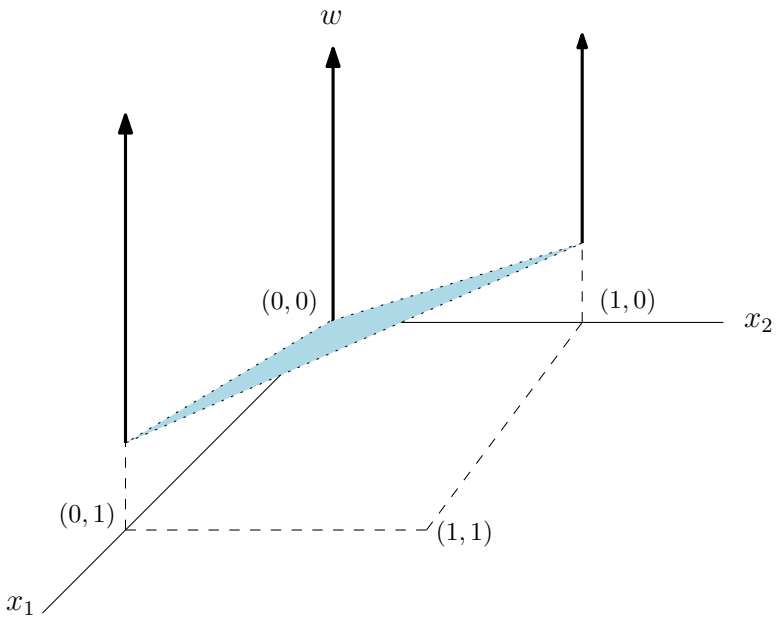
where f is a concave function, $a \in \mathbb{R}^n_+, k \in \mathbb{Z}_+$ and e is a vector of ones, and the cardinality constraint $e'x \leq k$ is assumed to be implied by the constraints $x \in X$. We give an illustration of difference between $\mathcal{Q}$ and $\mathcal{P}$ in Figure 8.

The contributions of this chapter are as follows. First, we give a complete description of conv($\mathcal{P}$) when the vector a in (42) has identical components. Similar to the classical work in [11], we write a pair of primal-dual linear programs corresponding to the convex lower envelope of $f(a'x)$ and valid inequalities of conv($\mathcal{P}$) respectively. Then we construct a pair of optimal primal and dual solutions that provide the explicit inequality description of conv($\mathcal{P}$). Second, we give a family of facet-defining inequalities of conv($\mathcal{P}$) for general nonnegative a. We obtain such inequalities through sequence dependent exact lifting and present a polynomial time algorithm to find the lifting coefficients corresponding to any given sequence. We give another family of approximately lifted inequalities that can be weaker than the exactly lifted facet-defining inequalities but much faster to compute within a branch-and-cut procedure. Finally, we demonstrate the effectiveness of using the proposed inequalities in a branch-and-cut framework to solve mean-risk knapsack problems. Our computational results show significant decrease in both time and the number of nodes over standard methods.

We close this section by a brief discussion of related literature. As mentioned earlier, an inequality description of the convex hull of the epigraph of a general submodular function



(a) $\mathcal{Q} = \{(w, x) \mid w \geq f(x)\}$



(b) $\mathcal{P} = \{(w, x) \mid w \geq f(x), x_1 + x_2 \leq 1\}$

Figure 8: Comparison of $\mathcal{Q}$ and $\mathcal{P}$.

Figure 8 gives an example of $\mathcal{Q}$ and $\mathcal{P}$ for $n = 2$. The two sets are denoted by black lines with directions. The shadow area represents non-trivial supporting hyperplanes of the convex hulls of two sets.

follows from the classical works [11, 32]. In the presence of constraints, submodular minimization is in general **NP**-hard and in most cases very hard to approximate [17, 28, 47]. Even for the cardinality constraint, the special case of size constrained minimum cut problem is **NP**-hard since it can be reduced to graph partitioning problem [15]. This suggest that there is little hope of obtaining a tractable inequality description of the convex hull of the epigraph of a submodular function under a cardinality constraint. However the submodular function considered here is of the special form $F(x) = f(a'x)$ where f is a concave function. Hassin and Tamir [23] solve the problem $\min\{c'x + f(a'x) : \ e'x \leq k, x \in \{0,1\}^n\}$ where f is a concave function, in polynomial time by reducing it to a two-dimension parametric linear programming problem. Onn and Rothblum [38] generalize the univariate concave function to multivariate case. From the equivalence of separation and optimization this suggests that for such a special submodular function a tractable description of the convex hull of the epigraph under a cardinality constraint is possible. However, to the best of our knowledge an explicit inequality description of such a set has not been presented before. Recently a number of works have used valid inequalities exploiting the submodularity of underlying functions within mixed integer linear programming approaches for various classes of mixed integer nonlinear programs [4, 1, 2, 6]. In particular, the work of Atamtürk and Narayanan [4] is most related to our work. They use the inequalities describing the convex hull of the epigraph of a submodular function without constraints, i.e. the set $\text{conv}(\mathcal{Q})$, to strengthen a second order cone programming (SOCP) relaxation of a mean-risk optimization problem in a branch-and-cut procedure and show that facet-defining inequalities of $\text{conv}(\mathcal{Q})$ significantly improves the performance. Our work extends this approach to include information from a cardinality constraint.

4.2 Valid inequalities for $\text{conv}(\mathcal{P})$

In this section, we study valid inequalities of the convex hull of the set (42):

$$\mathcal{P} = \left\{(w,x) \in \mathbb{R} \times \{0,1\}^n \ \mid \ w \geq f(a'x), \ e'x \leq k\right\},$$

where f is a concave function and a is a nonnegative vector. The following notation will be used throughout. Let $N = \{1, \ldots, n\}$. Denote $a(S) := \sum_{i \in S} a_i$ for any $S \subseteq N$. Given a

permutation $\sigma := ((1), \ldots, (n))$ of N, let $A_{(i)} = \sum_{j=1}^{i} a_{(j)}$ for all $(i) \in \sigma$.

Recall the unconstrained version of $\mathcal{P}$, i.e. the set $\mathcal{Q} = \{(w, x) \in \mathbb{R} \times \{0,1\}^n \mid w \geq f(a'x)\}$. Using the fact that $f(a'x)$ is submodular over $\{0,1\}^n$ it is known [11, 32] that every facet of $\text{conv}(\mathcal{Q})$, except the trivial bounds on x, is of the form

$$f(0) + \sum_{(i) \in \sigma} \rho_{(i)} x_{(i)} \leq w. \tag{43}$$

where

$$\rho_{(i)} = f(A_{(i-1)} + a_{(i)}) - f(A_{(i-1)}) \tag{44}$$

for $i \geq 1$ and σ is some permutation of N. Thus a complete inequality description of $\text{conv}(\mathcal{Q})$ is given by the inequalities (43) corresponding to all permutations. Although there is an exponential number of such inequalities, they can be separated in polynomial time. Following [4] we refer to the inequalities (43) as *extended polymatroid inequalities* (EPI). Clearly EP inequalities are valid for $\text{conv}(\mathcal{P})$. Next we strengthen the EP inequalities using the information from the cardinality constraint in $\mathcal{P}$.

Given a $(w, x) \in \mathbb{R} \times [0,1]^n$ with $x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(n)}$ we can check if $(w, x) \in \text{conv}(\mathcal{P})$ by solving the the following (dual) linear program:

$$\begin{aligned} \max \quad & \pi_0 + \sum_{i=1}^{n} \pi_{(i)} x_{(i)} \\ \text{s.t.} \quad & \pi_0 + \sum_{(i) \in S} \pi_{(i)} \leq f(a(S)) \quad \forall S, |S| \leq k. \end{aligned} \tag{45}$$

It follows that, given any permutation $\sigma = \{(1), \ldots, (n)\}$, an inequality $\pi_0 + \sum \pi_{(i)} x_{(i)} \leq w$ is valid for $\text{conv}(\mathcal{P})$ if and only if π_0, π is a feasible solution of (45). Also notice that coefficients of the EP inequality defined by (44) form a feasible solution of (45) since the EP inequality is valid for $\text{conv}(\mathcal{P})$.

Thus we can get an inequality description of $\text{conv}(\mathcal{P})$ if we obtained all optimal solutions to (45) corresponding to all vectors x. The construction is complicated by the fact that, unlike the unconstrained case, an optimal solution to (45) depends on the specific value of the vector x rather than just the permutation implied by its components. In the following subsection we show that we can do this construction and obtain a description of $\text{conv}(\mathcal{P})$

when all components of a are identical which we refer to as the *unweighted* case. In Section 4.2.2 we use a different approach, lifting, to derive valid inequalities for the *weighted* case where the vector a has nonidentical components.

4.2.1 Unweighted case: identical components

In this section we consider the case where $a_1 = \cdots = a_n = a$. Without loss of generality, unless otherwise specified, we consider the permutation $(1, \ldots, n)$. We simplify the notation by denoting $f(A_i) = f(a \cdot i)$ as $f(i)$ and write the coefficients (44) of the EP inequality as $\rho_i = f(i) - f(i-1)$ for $i \in N$. We also denote $f(a(S)) = f(|S|)$ for any $S \subseteq N$.

Given a vector $x \in \{x \in [0,1]^n : \ e'x \leq k\}$ such that $x_1 \geq x_2 \cdots \geq x_n$ we will next construct an optimal solution to (45). First we define a *critical* index $i_0 \in \{0, 1, \ldots, k-1\}$ corresponding to x as follows. Let $x_0 := 1$ and $z_i := (k-i)x_i - \sum_{j=i+1}^{k-1} x_j$ for $i \in \{0, 1, \ldots, k\}$ so that $z_k = 0, z_{k-1} = x_{k-1}, z_{k-2} = 2x_{k-2} - x_{k-1}, \ldots, z_0 = k - \sum_{j=1}^{k-1} x_j$. Let $y := \sum_{j=k}^{n} x_j$ and

$$i_0 := \operatorname{argmax} \{0 \leq i \leq k-1 \mid z_{i+1} \leq y \leq z_i\}. \tag{46}$$

Note that i_0 is well defined since the range $[z_{i+1}, z_i]$ is valid for any $i \in \{0, 1, \ldots, k-1\}$ as $z_i - z_{i+1} = (k-i)(x_i - x_{i+1}) \geq 0$, and y will be in some interval $[z_{i+1}, z_i]$. The latter follows from the fact that $y \geq z_k = 0$ and $y \leq z_0$ since $z_0 - y = k - \sum_{i=1}^{n} x_i \geq 0$ because x satisfies $e'x \leq k$. Given $i_0 \in \{0, \ldots, k-1\}$, we construct the following solution to (45):

$$\pi_j := \begin{cases} f(0) & j = 0 \\ \rho_j & j \leq i_0 \\ \dfrac{f(k) - f(i_0)}{k - i_0} & j > i_0. \end{cases} \tag{47}$$

We will first show that the solution constructed above corresponding to any $i_0 \in \{0, \ldots, k-1\}$ is feasible to (45) . Then we will show that if i_0 is constructed as in (46) then it is optimal. We will need the following result.

Lemma 32. *Given a concave function $f : \mathbb{R} \to \mathbb{R}$, for positive integers $t_1 > t_2, t_3 > t_4$, $t_2 \geq t_4$ and $t_1 - t_2 \geq t_3 - t_4$, we have*

$$\frac{f(t_1) - f(t_2)}{t_1 - t_2} \leq \frac{f(t_3) - f(t_4)}{t_3 - t_4}.$$

Proof. For any integer i, we define $\rho_i = f(i) - f(i-1)$. Then we have

$$\begin{aligned}\frac{f(t_3) - f(t_4)}{t_3 - t_4} &= \frac{\sum_{j \geq t_4}^{t_3 - 1} \rho_j}{t_3 - t_4} \\ &\geq \frac{\sum_{j=t_2}^{t_2+t_3-t_4-1} \rho_j}{t_3 - t_4} \\ &\geq \frac{\sum_{j=t_2}^{t_2+t_3-t_4-1} \rho_j + \sum_{j=t_2+t_3-t_4}^{t_1-1} \rho_j}{t_3 - t_4 + t_1 - t_2 - t_3 + t_4} \\ &= \frac{f(t_1) - f(t_2)}{t_1 - t_2},\end{aligned}$$

where the first inequality follows from the fact $\rho_j \geq \rho_{j'}$ when $j \leq j'$ (by concavity) and $t_2 \geq t_4$; and the second inequality follows from the same fact and $t_1 - t_2 \geq t_3 - t_4$. □

Proposition 33. *The solution π in* (47) *corresponding to any $i_0 \in \{0, \ldots, k-1\}$ is feasible for problem* (45).

Proof. For an $i_0 \in \{0, \ldots, k-1\}$, consider an arbitrary set S such that $|S| \leq k$. Let $S_1 = S \cap \{1, \ldots, i_0\}$ and $S_2 = S \backslash S_1$. Let $i_1 = |S_1|, i_2 = |S_2|$. Then from the construction (47), π satisfies

$$\pi_0 + \sum_{i \in S} \pi_i = \pi_0 + \sum_{i \in S_1} \pi_i + \sum_{i \in S_2} \pi_i \leq f(i_1) + \sum_{i \in S_2} \pi_i = f(i_1) + i_2 \cdot \frac{f(k) - f(i_0)}{k - i_0}.$$

The first inequality in the above chain follows from the fact that for $i \in S_1$, π_is are coefficients of EPI and hence satisfies $\pi_0 + \sum_{i \in S_1} \pi_i \leq f(S_1) = f(i_1)$. The second equality follows from the definition of the dual solution in (47). If $i_2 = 0$, then we already know it is feasible. Assume $i_2 > 0$, we need to show

$$f(i_1) + i_2 \cdot \frac{f(k) - f(i_0)}{k - i_0} \leq f(i_1 + i_2),$$

which is equivalent to

$$\frac{f(k) - f(i_0)}{k - i_0} \leq \frac{f(i_1 + i_2) - f(i_1)}{i_2}. \tag{*}$$

We consider two cases:

$i_2 \leq k - i_0$: Inequality (*) follows from Lemma 32 by setting $t_1 = k$, $t_2 = i_0$, $t_3 = i_1 + i_2$ and $t_4 = i_1$. Note that $t_2 \geq t_4$ since $i_0 \geq i_1$ and $t_1 - t_2 \geq t_3 - t_4$ since $k - i_0 \geq i_2$.

$i_2 > k - i_0$**:** First by concavity, we know

$$i_2(f(k) - f(i_1 + i_2)) \leq i_2(f(i) - f(i_1 + i_2 - (k - i_0))).$$

We also have

$$(i_2 + i_0 - k)(f(i_1 + i_2) - f(i_1)) \leq i_2(f(i_1 + i_2 - (k - i_0)) - f(i_1))$$

from Lemma 32 by setting $t_1 = i_1 + i_2, t_2 = i_1, t_3 = i_1 + i_2 - k + i_0$ and $t_4 = i_1$. To check that the condition of Lemma 32 is satisfied, first note $t_3 = i_1 + i_2 - (k - i_0) \geq 0$, then $t_2 = t_4$, and finally $t_1 - t_2 = i_2 \geq t_3 - t_4 = i_2 + i_0 - k$ since $i_0 \leq k$. Add these two inequalities together, and after rearrangement, we get inequality (*). □

Next we show that when i_0 is constructed as in (46) then π constructed as in (47) is an optimal solution to problem (45). We proceed by constructing a complementary solution to the following primal problem corresponding to (45):

$$\begin{aligned} \min \quad & \sum_{|S| \leq k} P(S) f(|S|) \\ \text{s.t.} \quad & \sum_{S: i \in S, |S| \leq k} P(S) = x_i \quad \forall i \in N \\ & \sum_{|S| \leq k} P(S) = 1 \\ & P(S) \geq 0, \quad \forall S, |S| \leq k. \end{aligned} \tag{48}$$

We construct a solution to (48) as follows:

$$P(\emptyset) = 1 - x_1;\ P(\{1, \dots j\}) = x_j - x_{j+1}\ \forall j < i_0;\ \text{and}\ P(\{1, \dots, i_0\}) = \frac{z_{i_0} - y}{k - i_0}. \tag{49}$$

Note that if $i_0 = 0$, then

$$P(\emptyset) = \frac{z_0 - y}{k}.$$

Let $\mathcal{S} := \{S \mid S \subset \{i_0 + 1, \dots, n\}, |S| = k - i_0\}$. For sets $S \in \mathcal{S}$ we set $P(S)$ according to Lemma 34 below. All remaining sets $S \subseteq N$ with $|S| \leq k$ are assigned $P(S) = 0$. The sets whose probabilities we are going to calculate are illustrated at Figure 9.

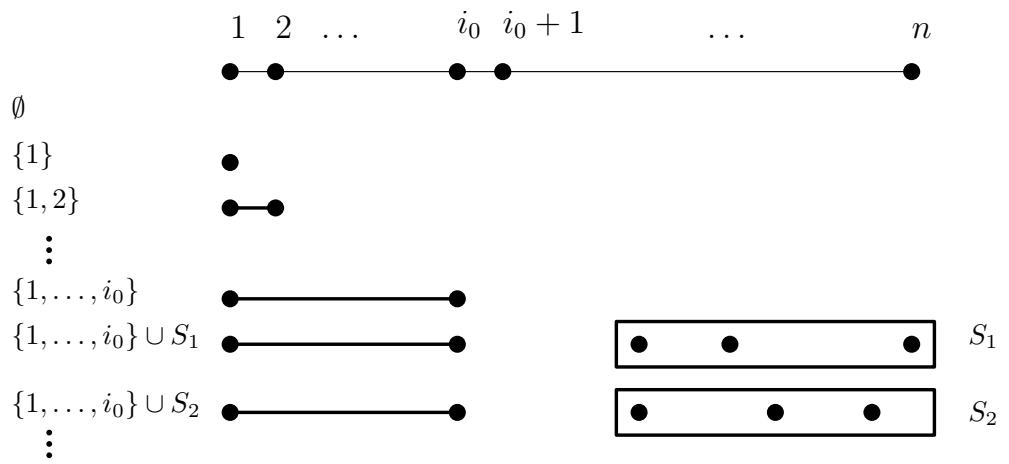


Figure 9: An illustration of an solution of (48), $S_i \subseteq \{i_0+1,\ldots,n\}, |S_i| = k - i_0$

A solution of (48) consists a chain of sets: $\emptyset, \{1\}, \ldots, \{1,\ldots,i_0\}$ and sets $\{1,\ldots,i_0\} \cup S_i$ where $S_i \subseteq i_0+1,\ldots,n$ are determined by Algorithm 6.

Lemma 34. *Suppose i_0 is constructed as in* (46) *then the following linear system in $P(S)$ for $S \in \mathcal{S}$*

$$\sum_{S:i\in S} P(\{1,\ldots,i_0\}\cup S) = x_i,\ i \in \{i_0+1,\ldots,n\}$$

has a nonnegative solution and

$$\sum_{S\in\mathcal{S}} P(\{1,\ldots,i_0\}\cup S) = \frac{\sum_{i=i_0+1}^{n} x_i}{k-i_0}.$$

Proof. To avoid confusion with the original notation, we change the notation as follows. Set $m = n - i_0, l = k - i_0,\ v_1 = x_{i_0+1}, v_2 = x_{i_0+2}, \ldots, v_m = x_n$. We will prove for $m, l \in \mathbb{Z}_+, m \geq l+1,\ v_1 \geq v_2 \geq \cdots v_m \geq 0$, and $\mathcal{S} = \{S \mid S \subset \{1,\ldots,m\}, |S| = l\}$, the following linear system in $q(S), S \in \mathcal{S}$

$$\sum_{S:j\in S} q(S) = v_j, j \in \{1,\ldots,m\},$$

has a nonnegative solution and $\sum_{S\in\mathcal{S}} q(S) = \frac{\sum_{j=1}^{m} v_j}{l}$.

We will use Algorithm 6 to construct a feasible solution. The main idea is for any set S, the associated variable $q^t(S)$ will remain nondecreasing in the iteration count t. For any j, v_j^t is the right-hand-side value that has not been satisfied yet, and it will remain

Algorithm 6: Recursive procedure to construct a feasible solution

$t \leftarrow 0, m^t \leftarrow m$;
$v_j^t \leftarrow v_j, 1 \le j \le m^t \quad q^t(S) \leftarrow 0, \forall S \in \mathcal{S}$;
while $m^t > l+1$ **do**
C1 **if** $lv_l^t \le \sum_{j=1}^{m^t} v_j^t - lv_{m^t}^t$ **then**
$q^{t+1}(\{1,\dots,l-1,m^t\}) \leftarrow q(\{1,\dots,l-1,m^t\}) + v_{m^t}^t$;
$v_j^{t+1} \leftarrow v_j^t - v_{m^t}^t, j \in \{1,\dots,l-1\}$;
$m^{t+1} \leftarrow m^t - 1$;
else
C2 $\Delta = \sum_{j=1}^{m^t} v_j^t/l - v_{l+1}^t$;
$q^{t+1}(\{1,\dots,l\}) \leftarrow q^t(\{1,\dots,l\}) + \Delta$;
$v_j^{t+1} \leftarrow v_j^t - \Delta, j \in \{1,\dots,l\}$;
$m^{t+1} \leftarrow m^t$
end
$q^{t+1}(S) \leftarrow q^t(S)$ for all the other S;
SRT sort $v_1^{t+1},\dots,v_{m^t}^{t+1}$ and index them so that $v_1^{t+1} \ge \cdots v_{m^{t+1}}^{t+1}$;
$t \leftarrow t+1$;
end
$q^t(\{1,\dots,l+1\}\setminus\{j\}) \leftarrow q^t(\{1,\dots,l+1\}\setminus\{j\}) + \frac{\sum_{\tau=1}^{l+1} v_\tau^t}{l} - v_j^t, j \in \{1,\dots,l+1\}$

nonincreasing in t. We will keep $v_j^{t+1} + \sum_{S,j\in S,S\in\mathcal{S}} q^{t+1}(S) = v_j$ along the way. At the end of the procedure, every item j will have its $v_j^t = 0$, and we will find the solution as desired.

We now exploit some properties of v_j^t and $q^t(S)$. Initially, we have

1. $v_j^0 \ge 0$ for every j
2. $q^0(S) = 0$ and $v_j^0 + \sum_{S,j\in S,S\in\mathcal{S}} q^0(S) = v^j$
3. Since $y \ge z_{i_0+1}$, we have $\sum_{j=k}^n x_j \ge (k-i_0-1)x_{i_0+1} - \sum_{j=i_0+2}^{k-1} x_j$. Then $\sum_{j=1}^m v_j = \sum_{j=i_0+1}^n x_j \ge (k-i_0)x_{i_0+1} = lv_1$.

We will show that v_ls satisfy the following invariance after finishing line SRT.

1. $v_j^{t+1} \ge 0$ for $j \in \{1,\dots,m^{t+1}\}$.
2. $v_j^{t+1} + \sum_{S,j\in S,S\in\mathcal{S}} q^{t+1}(S) = v_j$.
3. $lv_1^{t+1} \le \sum_{j=1}^{m^{t+1}} v_j^{t+1}$.

First we prove $v_j^{t+1} \geq 0$. Case C1 is easy since $v_j^t \geq v_{m^t}^t$ for all j. For Case C2 where $lv_l^t > \sum_{j=1}^{m^t} v_j^t - lv_{m^t}^t$, the smallest v_j^{t+1} among the ones updated is

$$v_l^t - \Delta = v_l^t - \frac{\sum_{j=1}^{m^t} v_j^t}{l} + v_{l+1}^t \geq \frac{\sum_{j=1}^{m^t} v_j^t}{l} - v_{m^t}^t - \frac{\sum_{j=1}^{m^t} v_j}{l} + v_{l+1}^t = v_{l+1}^t - v_{m^t}^t \geq 0.$$

Then we show $v_j^{t+1} + \sum_{S, j \in S, S \in \mathcal{S}} q^{t+1}(S) = v_j$. Since in the loop each iteration we only consider one set S, and for any item $j \in S$, $v_j^t - v_j^{t+1} = q^{t+1}(S) - q^t(S)$, the claim is true.

Then we prove that $lv_1^{t+1} \leq \sum_{j=1}^{m^{t+1}} v_j^{t+1}$ after line SRT. In Case C1, v_1^{t+1} will be either

- $v_1^t - v_{m^t}^t$. Then $\sum_{j=1}^{m^{t+1}} v_j^{t+1} = \sum_{j=1}^{m^t} v_j^t - lv_{m^t}^t \geq lv_1^t - lv_{m^t}^t$ since the claim holds for the previous iteration, or
- v_l^t. Then again $\sum_{j=1}^{m^{t+1}} v_j^{t+1} = \sum_{j=1}^{m^t} v_j^t - lv_{m^t}^t \geq lv_l^t$ because we are at Case C1.

In the second case C2, first notice that after line SRT, $v_1^{t+1} = v_{l+1}^t$ since the other choice $v_1^t - \Delta = v_1^t - \frac{\sum_{j=1}^{m^t} v_j^t}{l} + v_{l+1}^t \leq v_{l+1}^t$ because $lv_1^t \leq \sum_{j=1}^{m^t} v_j^t$. Then

$$\sum_{j=1}^{m^{t+1}} v_j^{t+1} = \sum_{j=1}^{m^t} v_j^t - l\Delta = \sum_{j=1}^{m^t} v_j^t - l\left(\frac{\sum_{j=1}^{m^t} v_j^t}{l} - v_{l+1}^t\right) = lv_{l+1}^t.$$

If Algorithm 6 terminates, we say that $q^t(S), \forall S \in \mathcal{S}$ is the solution desired. We consider two cases after the while loop. For any $j > l+1$, we have $v_j^t = 0$ which is equivalent to $\sum_{S, j \in S, S \in \mathcal{S}} q^t(S) = v_j$. For any $j \leq l+1$, before the last line, we have $\sum_{S, j \in S, S \in \mathcal{S}} q^t(S) = v_j - v_j^t$, and after last line, we have

$$\sum_{S, j \in S, S \in \mathcal{S}} q^t(S) = v_j - v_j^t + \sum_{j' \neq j} \left(\frac{\sum_{\tau=1}^{l+1} v_\tau^t}{l} - v_{j'}^t\right) = v_j - v_j^t + \sum_{\tau=1}^{l+1} v_\tau^t - \sum_{j' \neq j} v_{j'}^t = v_j.$$

If it does not terminate, we claim that $q(S) = \lim_{t \to \infty} q^t(S)$ for all $S \in \mathcal{S}$ is the solution desired. To prove that, we show that $\lim_{t \to \infty} v_j^t = 0$. If this is true, then we get $\lim_{t \to \infty} \sum_{S, j \in S, S \in \mathcal{S}} q^t(S) = v_j$. Notice that each time after we finish Case (C2), $\sum_{j=1}^{m^{t+1}} v_j^{t+1} \leq \frac{l}{l+1} \sum_{j=1}^{m^t} v_j^t$ because

$$\sum_{j=1}^{m^t} v_j^t \geq \sum_{j=1}^{l+1} v_j^t \geq (l+1) v_{l+1}^t, \text{ and } \sum_{j=1}^{m^{t+1}} v_j^{t+1} = \sum_{j=1}^{m^t} v_j^t - l\Delta = lv_{l+1}.$$

Since the algorithm does not terminate, Case C2 happens infinitely many times. Notice that initially $\sum_{j=1}^{m^0} v_j \leq m$, the value

$$\lim_{t\to\infty} \sum_{j=1}^{m^t} v_j^t \leq \lim_{t\to\infty} (\frac{l}{l+1})^t m = 0.$$

Since v_j^t is always nonnegative, $\lim_{t\to\infty} \sum_{j=1}^{m^t} v_j^t = 0$ and $\lim_{t\to\infty} v_j^t = 0$.

Now we calculate the sum of all variables. Given that

$$\sum_{S:j\in S} q(S) = v_j, j \in \{1, \ldots, m\},$$

we have

$$\sum_{j=1}^{m} \sum_{S:j\in S} q(S) = \sum_{j=1}^{m} v_j.$$

Since each $q(S)$ appears on the left side exactly l times,

$$l \sum_{S\in\mathcal{S}} q(S) = \sum_{j=1}^{m} v_j.$$

□

Lemma 35. *Suppose i_0 is constructed as in* (46) *then the solution $P(S)$ defined by* (49) *and Lemma 34 is a feasible solution to the primal problem* (48).

Proof. First we check that $\sum_{S:i\in S,|S|\leq k} P(S) = x_i$ for all $i \in N$. If $i > i_0$, this holds by Lemma 34. For $i \leq i_0$, apply (49), Lemma 34, and the definitions of z_{i_0} and y. We have:

$$\begin{aligned}
&\sum_{j=i}^{i_0} P(\{1,\ldots,j\}) + \sum_{S\in\mathcal{S}} P(S \cup \{1,\ldots,i_0\}) \\
=&x_i - x_{i_0} + \frac{z_{i_0} - y}{k - i_0} + \frac{\sum_{j=i_0+1}^{n} x_j}{k - i_0} \\
=&x_i - x_{i_0} + \frac{(k-i_0)x_{i_0} - \sum_{j=i_0+1}^{n} x_j}{k - i_0} + \frac{\sum_{j=i_0+1}^{n} x_j}{k - i_0} \\
=&x_i.
\end{aligned}$$

Next we check that the $P(S)$'s sum up to one. For $i_0 > 0$, we have the following by the same steps as above for $i = 1$ and the construction of $P(\emptyset)$:

$$P(\emptyset) + \sum_{j=1}^{i_0} P(\{1,\ldots,j\}) + \sum_{S\in\mathcal{S}} P(S \cup \{1,\ldots,i_0\}) = 1 - x_1 + x_1 = 1.$$

For $i_0 = 0$, we have

$$P(\emptyset) + \sum_{S \in \mathcal{S}} P(S) = \frac{k - \sum_{j=1}^{n} x_j}{k} + \frac{\sum_{j=1}^{n} x_j}{k} = 1.$$

□

Proposition 36. *Suppose i_0 is constructed as in (46) then the solution π constructed as in (47) is an optimal solution for problem (45).*

Proof. By Lemma 33 and Lemma 35, we already know we have a pair of primal and dual feasible solutions. Now we verify that the objective value of the primal solution is the same as the dual solution. For $i_0 > 0$, we have

$$\begin{aligned} & f(\emptyset)P(\emptyset) + \sum_{l=1}^{i_0} P(\{1, \ldots, l\}) f(l) + \sum_{S \in \mathcal{S}} P(S \cup \{1, \ldots, i_0\}) f(k) \\ =& f(0) + \sum_{j=1}^{i_0} (f(j) - f(j-1)) x_j + \sum_{j=i_0+1}^{n} \frac{f(k) - f(i_0)}{k - i_0} x_j. \end{aligned}$$

For $i_0 = 0$, we have

$$\begin{aligned} & f(\emptyset)P(\emptyset) + \sum_{S \in \mathcal{S}} P(S) f(k) \\ =& \frac{k - \sum_{j=1}^{n} x_j}{k} f(0) + \frac{\sum_{j=1}^{n} x_j}{k} f(k) \\ =& f(0) + \sum_{j=1}^{n} \frac{f(k) - f(0)}{k} x_j \end{aligned}$$

□

Theorem 37. *When $a_1 = \cdots = a_n = a$ then $\mathrm{conv}(\mathcal{P})$ is defined by the trivial inequalities $x \in [0,1]^n$, $e'x \leq k$ and the following inequalities*

$$f(0) + \sum_{j=1}^{i_0} (f(j) - f(j-1)) x_{(j)} + \sum_{j=i_0+1}^{n} \frac{f(k) - f(i_0)}{k - i_0} x_{(j)} \leq w \tag{50}$$

corresponding to every permutation $\sigma = \{(1), \ldots, (n)\}$ of N and $i_0 \in \{0, 1, \ldots, k-1\}$, where $f(j) = f(a \cdot j)$ for $j \in N$. Moreover given a $x \in [0,1]^n$, we can decide whether $x \in \mathrm{conv}(\mathcal{P})$ and find a violated inequality in $O(n \log n)$ time.

Proof. The first part follows from Propositions 33 and 36. Given $x \in [0,1]^n$ we can first check in $O(n)$ if $e'x \leq k$. If yes, then to compute the coefficients of (50), first we sort the components of x, which takes $O(n \log n)$ time. Then finding the desired i_0 takes $O(n)$ time. Once i_0 is found, the coefficients can be computed according to (47) in $O(n)$ time. Therefore we can check for a violated inequality of the form (50) in $O(n \log n)$ time. □

We refer to the inequalities (50) as *separation inequalities* (SI) since they can be exactly separated.

Example 38. Consider an example of $\mathcal{P} = \left\{(w,x) \in \mathbb{R} \times \{0,1\}^n \mid w \geq \sqrt{e'x}, e'x \leq k\right\}$. Let $n = 6, k = 3$. Let $\delta = (1,2,3,4,5,6)$ be a permutation of $\{1,\ldots,6\}$. Its corresponding extended polymatroid inequality (43) is

$$w \geq x_1 + 0.4142x_2 + 0.3178x_3 + 0.2679x_4 + 0.2361x_5 + 0.2134x_6;$$

and its corresponding separation inequality (50) for $i_0 = 2$ is

$$w \geq x_1 + 0.4142x_2 + 0.3178x_3 + \mathbf{0.3178}x_4 + \mathbf{0.3178}x_5 + \mathbf{0.3178}x_6.$$

We see the latter is tighter than the former at coefficients of x_4, x_5 and x_6.

4.2.2 Weighted case: nonindentical components

Now we consider the more general case where the components of the nonnegative vector a are not necessarily identical. We derive a family of facet-defining inequalities of $\text{conv}(\mathcal{P})$ through lifting. Consider a set $S \subset N$ such that $|S| = k$. Without loss of generality, we consider the set $S = \{1,\ldots,k\}$. The restriction of $\mathcal{P}$ by setting $x_i = 0$ for all $i \in N \backslash S$ is denoted by:

$$\mathcal{P}^0 = \left\{(w,x) \in \mathbb{R} \times \{0,1\}^S \mid w \geq f(a'x)\right\}.$$

Since there is no constraint on x in $\mathcal{P}^0$, we know that the extend polymatroid inequality (43) is facet-defining for $\mathcal{P}^0$:

$$f(0) + \sum_{i=1}^{k} \rho_i x_i \leq w \tag{51}$$

We then lift variables $x_{k+1}, \ldots, x_n$ sequentially in that order. The intermediate set of feasible points are:

$$\mathcal{P}^i = \left\{(w,x) \in \mathbb{R} \times \{0,1\}^{\{1,\ldots,k,\ldots,i\}} \mid w \geq f(a'x),\ e'x \leq k\right\}\ \forall\ i = k+1,\ldots,n.$$

Given a facet-defining inequality

$$f(0) + \sum_{j=1}^{k} \rho_j x_j + \sum_{j>k}^{i-1} \zeta_j x_j \leq w$$

for $\text{conv}(\mathcal{P}^{i-1})$, the lifting problem associated with $\mathcal{P}^i$ is:

$$\begin{aligned} \zeta_i := \min \quad & w - f(0) - \sum_{j=1}^{k} \rho_i x_j - \sum_{j>k}^{i-1} \zeta_j x_j \\ \text{s.t.} \quad & f\left(\sum_{j<i} a_j x_j + a_i\right) \leq w \\ & \sum_{j<i} x_j \leq k-1,\ x_j \in \{0,1\}\ \forall\ j = 1,\ldots,i-1 \end{aligned} \tag{52}$$

From [21], We have the following result.

Lemma 39. *If the lifting coefficients ζ_j for $j = k+1,\ldots,i$ are computed as in* (52) *then the inequality*

$$f(0) + \sum_{j=1}^{k} \rho_j x_j + \sum_{j>k}^{i} \zeta_j x_j \leq w$$

is facet-defining for $\text{conv}(\mathcal{P}^i)$.

Lemma 40. *The lifting coefficient ζ_i in* (52) *can be computed in $O(i^3)$ time.*

Proof. The optimization problem (52) is equivalent to the following problem of minimizing concave function over cardinality constraint

$$\min\ \left\{ f\left(\sum_{j<i} a_j x_j + a_i\right) - \sum_{j=1}^{k} \rho_j x_j - \sum_{j>k}^{i-1} \zeta_j x_j,\ \sum_{j<i} x_j \leq k-1, x_j \in \{0,1\}, \forall j = 1,\ldots,i-1 \right\}.$$

This problem can be reduced to a two-dimension parametric linear programming problem and solved by enumerating all its $O(i^2)$ extreme points. Atamtürk and Narayanan give such an algorithm in [5] that finds an optimal solution in $\text{O}(i^3)$. □

The following theorem directly follows from Lemma 39 and Lemma 40.

Theorem 41. *For a permutation* $((1), \ldots, (n))$*, the inequality*

$$f(0) + \sum_{i \leq k} \rho_{(i)} x_{(i)} + \sum_{i > k} \zeta_{(i)} x_{(i)} \leq w \tag{53}$$

is facet-defining for $\operatorname{conv}(\mathcal{P})$*, and can be computed in time* $O(n^4)$*.*

Next we relate the above inequality to the separation inequalities (50) when a has identical components.

Proposition 42. *If the components of* a *are identical, then* $\zeta_i = f(k) - f(k-1)$ *for* $i > k$*. Moreover for all* $i > k$*, the lifted coefficient* $\zeta_i = \pi_i$*, where* π_i *is given by* (47) *corresponding to* $i_0 = k - 1$*.*

Proof. Denote $f(\sum_i a_i x_i)$ as $f(\sum_i x_i)$ and $f(a \cdot i)$ as $f(i)$. We have

$$\begin{aligned}
\zeta_i &= \min_{x \in \{0,1\}^i} \left\{ f\left(\sum_{j<i} x_j + 1\right) - f(0) - \sum_{j=1}^{k} \rho_j x_j - \sum_{j>k}^{i-1} \zeta_j x_j \;\middle|\; \sum_{j<i} x_j \leq k-1 \right\} \\
&\geq \min_{x \in \{0,1\}^i} \left\{ f\left(\sum_{j<i} x_j + 1\right) - f\left(\sum_{j<i} x_j\right) \;\middle|\; \sum_{j<i} x_j \leq k-1 \right\} \\
&= f(k) - f(k-1),
\end{aligned}$$

where the inequality comes from the validity of the coefficients ρ_j for $1 \leq j \leq k$ and ζ_j for $j > k$, i.e., $f(0) + \sum_{j=1}^{k} \rho_j x_j + \sum_{j>k}^{i-1} \zeta_j x_j \leq f(\sum_{j<i} x_j)$ and the equality comes from the concavity of f. However $\zeta_i \leq f(k) - f(k-1)$ since $x_j = 1$ for $j \in \{1, \ldots, k-1\}$ is a solution of (52). Finally we observe that $\zeta_i = \pi_i$ defined in (47) when $i_0 = k - 1$ and $i > k$. □

The $O(n^4)$ time complexity of the exact lifted inequality (53) make it computationally ineffective in a branch-and-cut framework. Next, we derive another set of approximately lifted valid inequalities that will be used in our computational experiments.

Proposition 43. *For* $i > k$*, let* $T_{(i)} = \operatorname{argmax}\{a(T) \mid T \subseteq \{(1), \ldots, (i-1)\}, |T| = k-1\}$*, and* $\gamma_{(i)} = f(a(T_{(i)}) + a_{(i)}) - f(a(T_{(i)}))$*. The inequality,*

$$f(0) + \sum_{i \leq k} \rho_{(i)} x_{(i)} + \sum_{i > k} \gamma_{(i)} x_{(i)} \leq w \tag{54}$$

is valid for $\operatorname{conv}(\mathcal{P})$ *and can be computed in* $O(n \log n)$ *time.*

Proof. Without loss of generality, we fix the sequence as $(1, \ldots, n)$ in the proof. Let $T_i^* \subseteq \{1, \ldots, i-1\}$ denote the support of an optimal solution of (52), i.e. $x_j = 1$ for $j \in T_i^*$ is an optimal solution. We prove the statement by showing that for every $i > k$, $\gamma_i \leq \zeta_i$:

$$\begin{aligned}
\gamma_i &= f(a(T_i) + a_i) - f(a(T_i)) \\
&\leq f(a(T_i^*) + a_i) - f(a(T_i^*)) \\
&\leq f(a(T_i^*) + a_i) - f(0) - \sum_{j \in T_i^*, j \leq k} \rho_j - \sum_{j \in T_i^*, j > k} \zeta_j \\
&= \zeta_i.
\end{aligned}$$

Above the first inequality is by concavity of f and the fact that $a(T_i) \geq a(T_i^*)$. The second inequality comes from the validity of the coefficients ρ_j for $1 \leq j \leq k$ and ζ_j for $j > k$, i.e., $f(0) + \sum_{j \in T_i^*, j \leq k} \rho_j + \sum_{j \in T_i^*, j > k} \zeta_j \leq f(a(T_i^*))$. Finally the last equality is by definition of ζ_i.

For time complexity, the ρ_is can be computed in time $O(k)$. Then for γ_i, we maintain a sorted list of $a_1, \ldots, a_{i-1}$ and the sum of its $k-1$ largest items. For a new a_i, we insert it into the sorted list and update the sum, which takes $O(\log n)$ time. Therefore the total time will be $O(n \log n)$. □

We refer to the approximately lifted inequality (54) as a *lifted inequality* (LI) in the remainder of the chapter.

Example 44. Consider an example of $\mathcal{P} = \left\{(w, x) \in \mathbb{R} \times \{0,1\}^n \mid w \geq \sqrt{a'x}, e'x \leq k\right\}$ where $n = 6, k = 3$, and

$$a = (0.7522, 0.0212, 0.8534, 0.4937, 0.8597, 0.3100)$$

whose components are generated from uniform distribution $[0, 1]$. Let $\delta = (1, 2, 3, 4, 5, 6)$ be a permutation of $\{1, \ldots, 6\}$. Its corresponding extended polymatroid inequality (43) is

$$w \geq 0.8673x_1 + 0.0121x_2 + 0.3960x_3 + 0.1807x_4 + 0.2701x_5 + 0.0876x_6.$$

Its corresponding lifted inequality (54) is

$$w \geq 0.8673x_1 + 0.0121x_2 + 0.3960x_3 + \mathbf{0.1818}x_4 + \mathbf{0.3030}x_5 + \mathbf{0.1135}x_6.$$

We see that the latter is tighter than the former at coefficients of x_4, x_5 and x_6.

Remark 45. If the components of a are identical then $\gamma_i = f(k) - f(k-1)$, and therefore $\gamma_i = \zeta_i$ in this case.

We also have the following property that the (LI) inequality is at least as strong as the (EP) inequality.

Proposition 46. *$\gamma_{(i)} - \rho_{(i)} \geq 0$, and is positive if f is strictly monotone.*

Proof. Since

$$\gamma_{(i)} - \rho_{(i)} = f(a(T_{(i)} + a_{(i)}) - f(a(T_{(i)}) - \left(f\left(\sum_{j<i} a_{(j)} + a_{(i)} \right) - f\left(\sum_{j<i} a_{(j)} \right) \right),$$

and we know from the definition in (54) that $a(T_{(i)}) < \sum_{j<i} a_{(j)}$ if $i > k$, therefore by concavity $\gamma_{(i)} \geq \rho_{(i)}$. It is then positive if f is strictly monotone. □

4.3 Computational results

In this section we demonstrate the effectiveness of the proposed separation inequalities (50) and the lifted inequalities (54) for solving the following class of mean-risk knapsack problems [4]:

$$\min \left\{ -\lambda' x + \Omega(\epsilon)\sqrt{\nu' x} : \ b'x \leq B, \ x \in \{0,1\}^n \right\}. \tag{55}$$

As discussed in Section 4.1, problem (55) involves a weighted combination of the mean and standard deviation of the total value of a knapsack, where individual item values are independent stochastic with mean μ_i and variance ν_i for $i = 1, \ldots, n$. Following[4, 16, 7] the tradeoff between the mean and standard deviation of the objective is set as $\Omega(\epsilon) = \sqrt{\frac{1-\epsilon}{\epsilon}}$ where $\epsilon \in (0,1)$ represents a risk aversion parameter. The vector of item weights and knapsack capacity are denoted by b and B, respectively. Problem (55) can be formulated as a mixed-integer second order cone program (SOCP):

$$\min \left\{ -\lambda' x + \Omega(\epsilon) w : \ w \geq \sqrt{\sum_i \nu_i x_i^2}, \ b'x \leq B, \ x \in \{0,1\}^n \right\}. \tag{56}$$

We incorporate the proposed inequalities in a branch-and-cut framework for the above formulation. Note that inequalities derived in the previous section are for the cardinality

constraint $e'x \leq k$, while the constraint in (56) is a knapsack. We derive a cardinality constraint from the knapsack constraint as a simple cover inequality as follows. Sort b_i such that $b_1 \leq \cdots \leq b_n$, and find an index k such that $b_1 + \cdots b_k \leq B$ and $b_1 + \ldots + b_{k+1} > B$. Then the constraint $e'x \leq k$ is valid for $b'x \leq B$.

The implementation, written in python, is based on the mixed integer second order cone programming solver of Gurobi 5.6. The continuous relaxation of (56) at each node is solved using the Barrier method. We restrict the number of submodular inequalities added to at most five. Gurobi's internal cut parameters are in default setting. We disable multithreading, heuristics, and the concurrent MIP solver; and set the relative MIP optimality gap as 0.01% and the time limit of the computation to 30 minutes. All computations are on an Intel Xeon server with 16 cores and 7GB memory restriction.

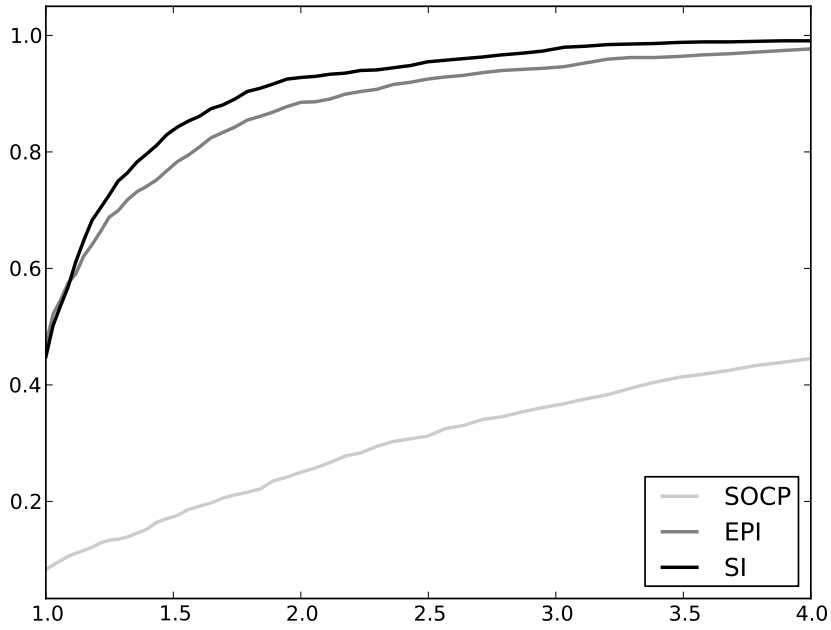
Our computational experiments use randomly generated instances of (56). Our parameters' setting generally follow the ones in [4]. For the *weighted case*, each component of λ and b is generated from a uniform distribution with range $[0, 100]$. The risk aversion parameter ϵ is from $\{0.01, 0.02, 0.03\}$. It is varied to observe the relationship between the weight on the nonlinear term in the objective and running time. For each component ν_i of the variance vector, its square root is uniformly generated in the range $[0, \alpha\lambda_i]$ where α is from $\{0.5, 0.75, 1\}$. Such generation ensure that $\sqrt{\nu_i}/\lambda_i \leq \alpha$. Finally we set $B = \lfloor \sum_i b_i/r \rfloor$ where r is also a varied parameter. For a fixed $b_1, \ldots, b_n$, the larger r is, the smaller B is, thus the smaller k is. Therefore the parameter r controls the right-hand-side of the cardinality constraint. For the *unweighted case*, the uniform distribution of components in λ is changed to $[1, 5]$ and every component of ν_i is identical and equal to a number whose square root is generated according to the uniform distribution with range $[1, \alpha \cdot \min_i \{\lambda_i\}]$. The reasons for the changes are the following. First we need the upper bound $\min_i \{\lambda_i\}$ to ensure $\sqrt{\nu_i} \leq \alpha\lambda_i$ for every i. Second if we used a lower bound of 0 for λ, it sometimes generates very small λ_i which then makes the non-linear part of the problem negligible. Thirdly, if we used the original upper bound of 100 for λ, instances for the unweighted case could be solved in less than one second making them too trivial.

The experiments are performed over all combinations of the parameters n, α, ϵ and

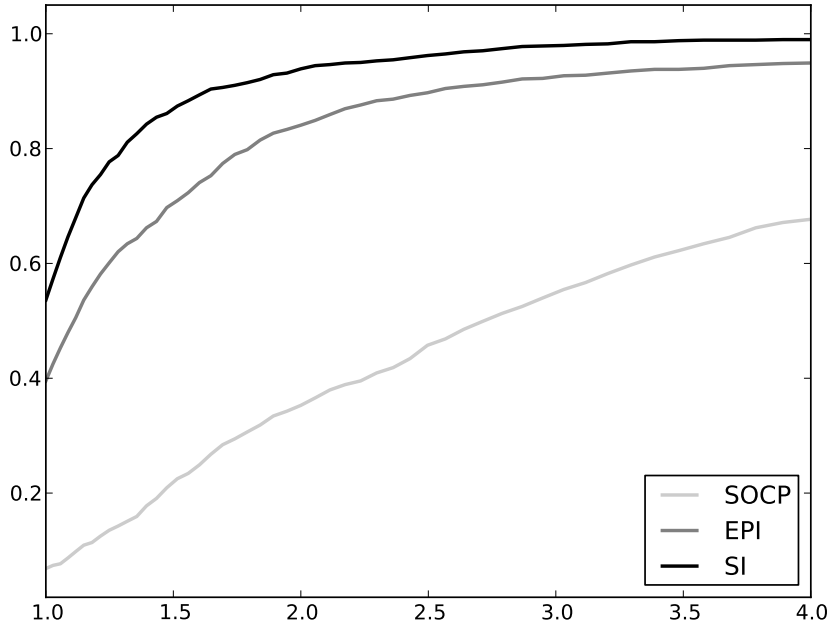
r; and for each combination we generate 20 instances. For each instance, we solve the problem using three different approaches: the SOCP formulation, SOCP formulation with the extended polymatroid inequalities (43) denoted by EPI, and the SOCP formulation with lifted inequalities (54) denoted by LI (or separation inequalities (47) if unweighted, denoted by SI).

Figure 10 presents the performance profiles of time for the weighted and unweighted cases. Following Dolan and Moré in [10], the performance profiles are constructed as follows. We have a set $\mathcal{S}$ of three solvers: SOCP, EPI, LI/SI, and a set of problem instances $\mathcal{P}$. For a solver $s \in \mathcal{S}$ and a problem instance $p \in \mathcal{P}$, we calculate its performance ratio $r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} \mid s \in \mathcal{S}\}}$ where $t_{p,s}$ is the time required by solver s on instance p. Figure 10 plots the cumulative distribution function of the performance ratio defined as $g_s(\tau) = \frac{1}{|\mathcal{P}|} |\{p \in \mathcal{P} \mid r_{p,s} \leq \tau\}|$. Observe that in both cases the proposed inequalities are uniformly better than SOCP formulation and SOCP with EPI. The improvement in the unweighted cases is very significant since in this case we have a complete description of the convex hull.

In Tables 3 and 4, we provide more details of the performance improvement. These tables report, for both the weighted and unweighted cases, the node counts, the solution times, and the corresponding percentage of improvements over EPI across various parameter settings. We note that with LI or SI, performance is better in almost every parameter group. We also observe that the weighted case takes much more time and nodes than the unweighted case. When the parameter α , which controls the range of the variance ν_i, gets smaller, the time and node needed decrease most significantly, compared with other parameters. This is because when α is small, the non-linear part of the constraint is less import. For both weighted and unweighted case, smaller α means that LI/SI cuts lead to better improvements. For ϵ, a smaller value puts more weight on the difficult nonlinear part of the objective and therefore the running time is longer. For the parameter r, we see that in the weighted case, larger r results in better improvement in case of the LI cuts. This observation matches our theoretical results since larger r indicates smaller k which implies that there will be more different coefficients in the LI cuts compared with EPI cuts.



(a) Weighed case



(b) Unweighted case

Figure 10: Performance profiles

Table 3: Performance comparison for weighted case

	SOCP		EPI		LI			
	Nodes	Time	Nodes	Time	Nodes	Improvement	Time	Improvement
$n = 50$	4480	15.18	1416	8.17	1422	-0.46%	6.60	19.28%
$n = 100$	150638	662.04	26093	148.49	24209	7.22%	140.98	5.05%
$\alpha = 0.5$	17678	78.07	2145	20.10	2018	5.92%	12.03	40.16%
$\alpha = 0.75$	74241	335.07	12571	76.20	12412	1.27%	75.51	0.90%
$\alpha = 1$	140758	602.69	26547	138.68	24018	9.53%	133.83	3.50%
$\epsilon = 0.03$	39773	159.74	4041	23.84	3732	7.67%	19.01	20.27%
$\epsilon = 0.02$	75951	307.15	12245	70.27	11754	4.01%	64.93	7.60%
$\epsilon = 0.01$	116953	548.94	24977	140.88	22962	8.07%	137.44	2.44%
$r = 7.5$	64116	276.29	8004	55.48	8007	-0.03%	48.97	11.73%
$r = 5$	95961	420.91	12622	71.94	11194	11.31%	68.09	5.36%
$r = 2$	72599	318.63	20637	107.57	19247	6.74%	104.32	3.03%

4.4 Conclusion

In this chapter, we develop a mixed-integer linear formulation of the epigraph of submodular function $f(a'x)$ together with a cardinality constraint. For special cases when every component of a is identical, we give a complete description of the convex hull and provide a fast separation algorithm. For the general case, we develop a family of facet-defining inequalities through lifting and another family of valid inequalities that are easily computable in a branch-and-cut framework. Computational experiments show significant improvement over the standard formulation without considering the cardinality constraint.

The question remains whether for a general vector a the convex hull of the epigraph of submodular function $f(a'x)$ with cardinality constraint may have a simple polynomial time separation algorithm similar to Algorithm 2 for the epigraph of submodular function without constraints. Linear optimization over the epigraph of $f(a'x)$ with cardinality constraint can be solved by an algorithm in [5], but a separation algorithm similar to Algorithm 2 needs much more structural information on the inequalities describing epigrah, which we do not know yet.

Table 4: Performance comparison for unweighted case

	SOCP		EPI		SI			
	Nodes	Time	Nodes	Time	Nodes	Improvement	Time	Improvement
$n = 50$	3923	12.22	2962	8.92	2074	29.96%	7.85	12.02%
$n = 100$	21133	77.27	4248	22.10	3482	18.01%	16.27	26.41%
$\alpha = 0.5$	1643	5.04	773	5.55	731	5.33%	2.56	53.91%
$\alpha = 0.75$	6180	19.99	1600	5.95	1460	8.72%	5.51	7.47%
$\alpha = 1$	29761	109.20	8442	35.03	6144	27.22%	28.10	19.77%
$\epsilon = 0.03$	2287	7.78	905	3.43	838	7.35%	3.24	5.53%
$\epsilon = 0.02$	4631	15.31	1331	6.86	1262	5.17%	4.76	30.65%
$\epsilon = 0.01$	30666	111.13	8579	36.24	6235	27.32%	28.17	22.27%
$r = 7.5$	10300	35.35	4883	20.57	3153	35.43%	12.67	38.42%
$r = 5$	13785	52.23	4613	17.74	3871	16.09%	18.44	-3.96%
$r = 2$	13498	46.65	1319	8.22	1312	0.52%	5.06	38.47%

REFERENCES

[1] Ahmed, S. and Atamtürk, A., "Maximizing a class of submodular utility functions," *Mathematical Programming*, vol. 128, no. 1, pp. 149–169, 2011.

[2] Ahmed, S. and Papageorgiou, D. J., "Probabilistic set covering with correlations," *Operations Research*, vol. 61, no. 2, pp. 438–452, 2013.

[3] Asadpour, A., Nazerzadeh, H., and Saberi, A., "Stochastic submodular maximization," in *Proceedings of the 4th International Workshop on Internet and Network Economics*, WINE, pp. 477–489, 2008.

[4] Atamtürk, A. and Narayanan, V., "Polymatroids and mean-risk minimization in discrete optimization," *Operations Research Letters*, vol. 36, no. 5, pp. 618–622, 2008.

[5] Atamtürk, A. and Narayanan, V., "The submodular knapsack polytope," *Discrete Optimization*, vol. 6, no. 4, pp. 333 – 344, 2009.

[6] Baumann, F., Berckey, S., and Buchheim, C., "Exact algorithms for combinatorial optimization problems with submodular objective functions," in *Facets of Combinatorial Optimization* (Jünger, M. and Reinelt, G., eds.), pp. 271–294, Springer Berlin Heidelberg, 2013.

[7] Bertsimas, D. and Popescu, I., "Optimal inequalities in probability theory: A convex optimization approach," *SIAM Journal on Optimization*, vol. 15, no. 3, pp. 780–804, 2005.

[8] Charikar, M., Chekuri, C., and Pál, M., "Sampling bounds for stochastic optimization," in *Proceedings of the 9th International Workshop on Randomization and Computation*, RANDOM, pp. 257–269, 2005.

[9] Conforti, M. and Cornuéjols, G., "Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem," *Discrete Applied Mathematics*, vol. 7, no. 3, pp. 251–274, 1984.

[10] Dolan, E. D. and Moré, J. J., "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.

[11] Edmonds, J., "Submodular functions, matroids, and certain polyhedra," in *Combinatorial Structures and Their Applications*, pp. 68–87, Gordon and Breach, 1971.

[12] Feige, U., Mirrokni, V., and Vondrak, J., "Maximizing non-monotone submodular functions," in *Foundations of Computer Science, 2007. FOCS '07. 48th Annual IEEE Symposium on*, pp. 461–471, Oct 2007.

[13] Feige, U., "A threshold of $\ln n$ for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[14] Feldman, E., Lehrer, F., and Ray, T., "Warehouse location under continuous economies of scale," *Management Science*, vol. 12, pp. 670–684, 1966.

[15] Garey, M. R. and Johnson, D. S., *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[16] Ghaoui, L. E., Oks, M., and Oustry, F., "Worst-case value-at-risk and robust portfolio optimization: A conic programming approach," *Operations Research*, vol. 51, no. 4, pp. 543–556, 2003.

[17] Goel, G., Karande, C., Tripathi, P., and Wang, L., "Approximability of combinatorial problems with multi-agent submodular cost functions," in *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, pp. 755–764, IEEE, 2009.

[18] Goemans, M. X., Harvey, N. J. A., Iwata, S., and Mirrokni, V., "Approximating submodular functions everywhere," in *SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, (Philadelphia, PA, USA), pp. 535–544, Society for Industrial and Applied Mathematics, 2009.

[19] Goemans, M. X. and Soto, J. A., "Symmetric submodular function minimization under hereditary family constraints," *CoRR*, vol. abs/1007.2140, 2010.

[20] Grötschel, M., Lovász, L., and Schrijver, A., "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1981.

[21] Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W., "Sequence independent lifting in mixed integer programming," *Journal of Combinatorial Optimization*, vol. 4, no. 1, pp. 109–129, 2000.

[22] Hajiaghayi, M., Mahdian, M., and Mirrokni, V., "The facility location problem with general cost functions," *Networks*, vol. 42, no. 1, pp. 42–47, 2003.

[23] Hassin, R. and Tamir, A., "Maximizing classes of two-parameter objectives over matroids," *Mathematics of Operations Research*, vol. 14, no. 2, pp. 362–375, 1989.

[24] Hille, E. and Phillips, R. S., *Functional analysis and semi-groups*, vol. 31. American Mathematical Soc., 1957.

[25] Ibarra, O. H. and Kim, C. E., "Fast approximation algorithms for the knapsack and sum of subset problems," *Journal of the ACM*, vol. 22, no. 4, pp. 463–468, 1975.

[26] Ishii, H., Shiode, S., Nishida, T., and Namasuya, Y., "Stochastic spanning tree problem," *Discrete Applied Mathematics*, vol. 3, no. 4, pp. 263 – 273, 1981. Special Copy.

[27] Iwata, S., Fleischer, L., and Fujishige, S., "A combinatorial strongly polynomial algorithm for minimizing submodular functions," *Journal of the ACM*, vol. 48, no. 4, pp. 761–777, 2001.

[28] Iwata, S. and Nagano, K., "Submodular function minimization under covering constraints," in *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, pp. 671–680, IEEE, 2009.

[29] Klastorin, T., "On a discrete nonlinear and nonseparable knapsack problem," *Operations Research Letters*, vol. 9, no. 4, pp. 233–237, 1990.

[30] Lee, J., Mirrokni, V. S., Nagarajan, V., and Sviridenko, M., "Non-monotone submodular maximization under matroid and knapsack constraints," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 323–332, ACM, 2009.

[31] Li, J. and Deshpande, A., "Maximizing expected utility for stochastic combinatorial optimization problems," in *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*, FOCS, pp. 797 –806, 2011.

[32] Lovász, L., "Submodular functions and convexity," in *Mathematical Programming The State of the Art*, pp. 235–257, Springer, 1983.

[33] Mehrez, A. and Sinuany-Stern, Z., "Resource allocation to interrelated risky projects using a multiattribute utility function," *Management Science*, vol. 29, no. 4, pp. 430–439, 1983.

[34] Meyer, R. R., "On the existence of optimal solutions to integer and mixed-integer programming problems," *Mathematical Programming*, vol. 7, no. 1, pp. 223–235, 1974.

[35] Nemhauser, G. L. and Wolsey, L. A., *Integer and combinatorial optimization*, vol. 18. Wiley New York, 1988.

[36] Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L., "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[37] Nikolova, E., "Approximation algorithms for reliable stochastic combinatorial optimization," in *Proceedings of the 13th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, APPROX, pp. 338–351, Springer, 2010.

[38] Onn, S. and Rothblum, U. G., "Convex combinatorial optimization," *Discrete & Computational Geometry*, vol. 32, no. 4, pp. 549–566, 2004.

[39] Ozsen, L., Coullard, C. R., and Daskin, M. S., "Capacitated warehouse location model with risk pooling," *Naval Research Logistics (NRL)*, vol. 55, no. 4, pp. 295–312, 2008.

[40] Romeijn, H. E., Sharkey, T. C., Shen, Z.-J. M., and Zhang, J., "Integrating facility location and production planning decisions," *Networks*, vol. 55, no. 2, pp. 78–89, 2010.

[41] Schoemaker, P. J., "The expected utility model: Its variants, purposes, evidence and limitations," *Journal of Economic Literature*, vol. 20, no. 2, pp. 529–563, 1982.

[42] Schrijver, A., "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," *Journal of Combinatorial Theory, Series B*, vol. 80, no. 2, pp. 346–355, 2000.

[43] Shapiro, A., "Monte carlo sampling methods," in *Stochastic Programming* (Ruszczyński, A. and Shapiro, A., eds.), vol. 10 of *Handbooks in Operations Research and Management Science*, pp. 353 – 425, Elsevier, 2003.

[44] Shapiro, A., Dentcheva, D., and Ruszczyński, A., *Lectures on Stochastic Programming: Modeling and Theory*. Society for Industrial and Applied Mathematics, 2009.

[45] Stratila, D., *Combinatorial Optimization Problems with Concave Costs*. PhD thesis, MIT, 2009.

[46] Sviridenko, M., "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letter*, vol. 32, no. 1, pp. 41–43, 2004.

[47] Svitkina, Z. and Fleischer, L., "Submodular approximation: Sampling-based algorithms and lower bounds," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1715–1737, 2011.

[48] Swamy, C. and Shmoys, D. B., "Sampling-based approximation algorithms for multistage stochastic," in *Proceedings of 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, pp. 357–366, 2005.

[49] Topkis, D. M., *Supermodularity and complementarity*. Princeton University Press, 1998.

[50] Von Neumann, J. and Morgenstern, O., *Theory of games and economic behavior*. Princeton University Press, 1944.

[51] Vondrák, J., "Optimal approximation for the submodular welfare problem in the value oracle model," in *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC, pp. 67–74, ACM, 2008.

[52] Vondràk, J., "Submodularity and curvature: the optimal algorithm," *RIMS Kokyuroku Bessatsu B23, Kyoto*, pp. 253–266, 2010.

[53] Weingartner, H. M., "Capital budgeting of interrelated projects: survey and synthesis," *Management Science*, vol. 12, no. 7, pp. 485–516, 1966.

[54] Wolsey, L. A., "Valid inequalities and superadditivity for 0-1 integer programs," *Mathematics of Operations Research*, vol. 2, no. 1, pp. 66–77, 1977.

[55] Wolsey, L. A., "Maximising real-valued submodular functions: Primal and dual heuristics for location problems," *Mathematics of Operations Research*, vol. 7, no. 3, pp. 410–425, 1982.

[56] Yu, J. and Ahmed, S., "Maximizing a class of submodular utility functions with constraints," 2014. `http://www.optimization-online.org/DB_HTML/2014/12/4712.html`.

[57] Yu, J. and Ahmed, S., "Maximizing expected utility over a knapsack constraint," 2014. `http://www.optimization-online.org/DB_HTML/2012/10/3648.html`.

[58] Yu, J. and Ahmed, S., "Polyhedral results for a class of cardinality constrained submodular minimization problems," 2014. http://www.optimization-online.org/DB_HTML/2014/08/4522.html.