# Maximizing a class of submodular utility functions with constraints

Jiajin Yu and Shabbir Ahmed*
School of Industrial & Systems Engineering,
Georgia Institute of Technology, Atlanta, GA 30332

December 30, 2014

**Abstract**

Motivated by stochastic 0-1 integer programming problems with an expected utility objective, we study the mixed-integer nonlinear set: $P = \left\{(w,x) \in \mathbb{R} \times \{0,1\}^N \,:\, w \le f(a'x+d), b'x \le B\right\}$ where $N$ is a positive integer, $f : \mathbb{R} \mapsto \mathbb{R}$ 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 $P$ by exploiting submodularity of the function $f(a'x+d)$ over $\{0,1\}^N$ and the knapsack constraint $b'x \le B$. Computational effectiveness of the proposed inequalities within a branch-and-cut framework is illustrated using instances of an expected utility capital budgeting problem.

*Keywords:* Submodularity, cutting planes, lifting, mixed integer nonlinear programming, branch-and-cut.

## 1 Introduction

Consider a stochastic 0-1 integer programming problem with $N$ variables and a finitely distributed random objective vector taking value $v_i \in \mathbb{R}^N$ under scenario $i$ with probability $\pi_i$ for $i = 1\ldots, m$. An expected utility maximization model (cf. [1, 6, 10] and references therein) for this problem is of the form

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

where $f$ is a concave and increasing utility function and $X$ is the set of feasible solutions. Problem (1) is a (convex) nonlinear integer program, i.e. without the integrality restrictions it is a convex optimization problem.

In this paper, we adopt a mixed integer *linear* programming (MILP) approach for (1). To this end, consider the reformulation

$$\max\left\{\sum_{i=1}^{m} \pi_i w_i \,:\, x \in X, (w_i, x) \in Q_i, \forall i = 1, \ldots, m\right\},$$

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

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

$a \in \mathbb{R}^N$ and $d \in \mathbb{R}$. We intend to describe $Q$ using linear inequalities so as to obtain an MILP formulation of (1). Since $Q$ is the union of a finite set of half lines with a common recession direction, its convex hull

*Corresponding Author. Email: sahmed@isye.gatech.edu

conv$(Q)$ is a polyhedron, and strong valid inequalities for conv$(Q)$ can help strengthen an MILP formulation of (1).

If the vector $a$ is nonnegative or nonpositive, the function $g(x) := f(a'x)$ is submodular over $\{0,1\}^N$ (cf. [1,8]). 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. Using submodularity, Nemhauser and Wolsey [7] provided a mixed-integer linear programming description of $Q$ using an exponential family of inequalities. In [1], Ahmed and Atamtürk give strong valid inequalities of conv$(Q)$ by lifting the inequalities developed in [7]. In this paper we further improve such a formulation by incorporating information from the constraint $x \in X$. In particular, we consider the case when $X$ is defined by a knapsack constraint and study the following mixed-integer nonlinear set

$$P = \left\{ x \in \{0,1\}^N, w \in \mathbb{R} \,:\, w \leq f(a'x + d), b'x \leq B \right\}, \tag{2}$$

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 paper 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 and replacing the knapsack constraint with a cardinality constraint. We give a polynomial time algorithm to compute this subadditive lifting function. We demonstrate the effectiveness of the proposed inequalities in a branch-and-cut framework to solve an expected utility capital budgeting problem. Compared with the cuts developed in [1], our proposed cuts significantly reduce CPU time, the number of nodes explored and the number of cuts added in the branch-and-cut process.

## 2 Valid inequalties by lifting

In this section, we study valid inequalities for the convex hull of (2):

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

where $f$ is a strictly 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. With slight abuse of notation, we 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

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

Since $f(a'x + d)$ is submodular over $\{0,1\}^n$, it has been shown in [7, p. 710] that the following two exponential families of inequalities

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

$$w \leq f(a(S)) - \sum_{i \in S} \rho_i(S \backslash i)(1 - x_i) + \sum_{i \in N \backslash S} \rho_i(\emptyset) x_i, \quad \forall S \subseteq N \tag{4}$$

can be used to define a mixed-integer linear formulation of $Q$. That is

$$Q = \left\{ x \in \{0,1\}^N, w \in \mathbb{R} \,:\, (3) \text{ or } (4) \right\}.$$

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

## 2.1 Uplifting

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

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

It follows from (4) that the following inequality

$$w \leq f(a(S)) - \sum_{i\in S} \rho_i(S\backslash i)(1 - x_i) \tag{5}$$

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 (5), consider the following lifting function

$$\begin{aligned} \zeta_1(\delta, \beta) := \max \quad & w + \sum_{i\in S} \rho_i(S\backslash i)(1 - x_i) - f(a(S)) \\ \text{s.t.} \quad & f(\sum_{i\in S} a_i x_i + \delta) \geq w \\ & \sum_{i\in S} b_i x_i + \beta \leq B \\ & x_i \in \{0,1\}, \forall i \in S. \end{aligned} \tag{6}$$

Problem (6) 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

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

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 $b_1 + \ldots + b_i \leq B - \beta$. Letting

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

a relaxation of (6) is

$$\begin{aligned} \zeta_2(\delta, k) := \max \quad & w + \sum_{i\in S} \rho_i(S\backslash i)(1 - x_i) - f(a(S)) \\ \text{s.t.} \quad & f(\sum_{i} a_i x_i + \delta) \geq w \\ & \sum_{i\in S} x_i \leq k \\ & x_i \in \{0,1\}, \forall i \in S. \end{aligned} \tag{7}$$

We now discuss some properties of the solutions of (7). 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 (7) is the set $\{i \,:\, x_i = 1\}$. Some properties may have been implicitly shown in [1], nevertheless we give the proof here for the sake of completeness.

**Lemma 1.** *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 a solution with support $T$.*

*Proof.* For case 1, we have the 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 $a(T)+\delta \leq a(S)-a_{i_2}$, we have

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

For case 2, we have the difference of objective values as

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

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

**Lemma 2.** *If there is no cardinality constraint in problem (7) (i.e. $k\geq|S|$), then we have the following properties:*

- *If $\delta\geq A(1:l)-A(j:l)$, then there exists a solution with support of size $l-j$ which is no worse than a solution with a support of larger size.*
- *If $\delta<A(1:l)-A(j+1:l)$, then there exists a solution with support of size $l-j$ which is no worse than a solution with a support of smaller size.*
- *If $A(1:l)-A(j:l)\leq\delta<A(1:l)-A(j+1:l)$, then there exists an optimal solution with support $\{j+1,\ldots,l\}$.*

*Proof.* When $\delta\geq A(1:l)-A(j:l)$, 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

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

since $a_i\geq 0, a(S)-a_i\leq A(j:l)+\delta-a_i\leq a(T)+\delta-a_i$, 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 the 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 1. 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 proved 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 1. □

**Lemma 3.** *If $\delta < A(1:l) - A(l-k+1:l)$, then let*

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

*If such a $j$ exists, then a solution with support $\{j+1,\ldots,j+k\}$ is optimal for (7). Otherwise $\delta < A(1:l) - A(1:k+1)$, then a solution with support $\{1,\ldots,k\}$ is optimal for (7).*

*Proof.* First by Lemma 2, 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 2, 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 $\delta < A(1:l) - A(l-k:l)$, 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 1. 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 the following 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 1 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 1. 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 1. Such swap cannot go on forever and eventually we will reach case 2a. □

Lemma 2 and Lemma 3 imply Algorithm 1. Given the input $\delta$, the algorithm searches for an interval that $\delta$ belongs to. If $\delta > A(1:l) - A(l-k:l)$, the resulting set $T$ is obtained from a chain of sets. Otherwise, $T$ consists $k$ consecutive items of $\{1,\ldots,l\}$. We illustrate the solutions of (7) for different $\delta$ in Figure 1.

**Proposition 4.** *Algorithm 1 solves problem (7) in $O(|S|)$ time.*

*Proof.* By Lemma 2 and Lemma 3, there are $|S|+1$ possible solutions of problem (7). Each solution corresponds to an interval that $\delta$ may belong to. These $|S|+1$ intervals are disjoint and can be ordered linearly. Algorithm 1 searches the intervals in this linear order to determine where $\delta$ belongs to. For each interval, the algorithm takes constant number of operations. Thus Algorithm 1 solves problem (7) in $O(|S|)$ time. □

## 2.2 Subadditive approximation

The lifting function defined by problem (7) is not necessarily subaddtive. In order to have sequence independent lifting, we need a subadditive lifting function [4, 9]. To achieve that, we consider the continuous relaxation of problem (7). Define

$$\begin{aligned} \gamma(\delta,k) := \max \quad & w + \sum_{i\in S} \rho_i(S\setminus i)(1-x_i) - f(a(S)) \\ \text{s.t.} \quad & f(\sum_{i\in S} a_i x_i + \delta) \geq w \\ & \sum_{i\in S} x_i \leq k \\ & 0 \leq x_i \leq 1, \forall i \in S. \end{aligned} \tag{8}$$

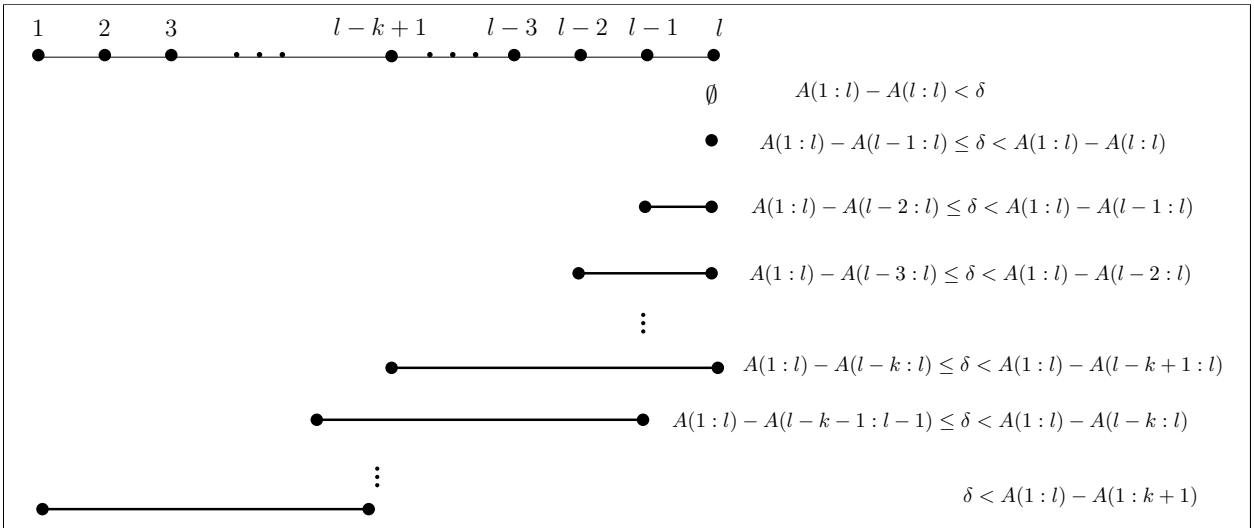

Figure 1: Solutions of (7) for different $\delta$

**Algorithm 1:** Greedy Algorithm for solving (7)

**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**
    $T\leftarrow T\backslash\{j\}$;
    $j\leftarrow j+1$;
**else**
  **while** $j>1$ *and* $a(S)-a(T)-a_{j-1}>\delta$ **do**
    $T\leftarrow T\backslash\{j+l-1\}\cup\{j-1\}$;
    $j\leftarrow j-1$;

By choosing $k$ appropriately we obtain a subadditive lifting function.

**Lemma 5.** *Let* $k_1=\max\{k(b_i,S):i\in N\backslash S\}$, $k_2=\min\{k:\gamma(0,k)\geq 0\}$, *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 of $\delta$. First $\gamma(\delta,k_0)$ is concave in $\delta$, because for each $\delta$, it is the maximum of a concave function of $\delta$ over a convex set $\{x:\sum_{i\in S}x_i\leq k_0,0\leq x_i\leq 1\}$. 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 [5, p. 239], we know that a concave function $\gamma:\mathbb{R}_+\rightarrow\mathbb{R}$ is subadditive if and only if $\gamma(0)\geq 0$. Therefore $\gamma(\delta,k_0)$ is a subadditive function of $\delta$. □

Since $\gamma(\delta,k_0)$ is subadditive in $\delta$, now we lift variables in $N\backslash S$ in a sequence independent order [4, 9] and have a family of valid inequalities.

**Theorem 6.** *For any set $S \subseteq N$, the following inequality*

$$w \leq f(a(S)) - \sum_{i \in S} \rho_i(S \backslash i)(1 - x_i) + \sum_{i \in N \backslash S} \gamma(a_i, k_0) x_i \tag{9}$$

*is valid for $P$.*

*Example 7.* We give an example comparing inequality (9) and the lifted inequality in [1]. Consider the model $P = \left\{(w, x) \in \mathbb{R} \times \{0,1\}^n : w \leq -\exp(-a'x), b'x \leq B\right\}$. 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

$$b = (0.3023, 0.1892, 0.3884, 0.1047, 0.5938, 0.6699)$$

be two vectors where each component is generated from a uniform random distribution $[0, 1]$. We calculate $k_0 = 3$ in (9). The lifted inequality in [1], which is the same as (9) 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 (9), which is tighter than the above at coefficients of $x_5, x_6$, is

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

## 2.3 Computing the subadditive lifting function

In this subsection we present a polynomial algorithm to solve problem (8) in order to compute the lifting function $\gamma(\delta, k_0)$. To achieve this, we first focus on the following more general problem

$$\max f(a'x + d) - c'x, ex \leq k, x \in [0, 1]^N, \tag{10}$$

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 (10) if it is fractional (at least one component is in $(0, 1)$) and together with Proposition 4, we can 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.

First we write out the KKT conditions for (10).

$$a_i f'(a'x + d) - c_i = \lambda + \alpha_i - \beta_i \tag{11}$$
$$\lambda(\sum_{i \in N} x_i - k) = 0 \tag{12}$$
$$\alpha_i(x_i - 1) = 0 \tag{13}$$
$$\beta_i x_i = 0 \tag{14}$$
$$\lambda, \alpha_i, \beta_i \geq 0$$
$$\sum_{i \in N} x_i \leq k$$
$$0 \leq x_i \leq 1.$$

We rewrite (11) as the following

$$f'(a'x + d) = \frac{c_i + \lambda}{a_i} + \frac{\alpha_i - \beta_i}{a_i}. \tag{15}$$

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 8.** *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 (15) 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 (13). 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 (14). □

**Lemma 9.** *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 (13) and (14), $\alpha_i = \beta_i = 0$. By Lemma 8, 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 (12). □

Before proceeding to the algorithm for solving problem (10), we need the following lemmas for solving two linear systems.

**Lemma 10.** *For $a_1 \geq a_2 \geq \cdots \geq a_n$, $h \geq 0$, and an integer $k \leq n$, the linear system*

$$\begin{aligned} a_1 x_1 + \cdots + a_n x_n &= h \\ x_1 + \cdots + x_n &\leq k \\ 0 \leq x_i \leq 1, \quad &\forall i \in \{1, \ldots, n\} \end{aligned}$$

*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

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \leq a_1 + \cdots + a_k < h.$$

That is, no solution can satisfy the condition $a_1 x_1 + \cdots + a_n x_n = h$. □

**Lemma 11.** *For $a_1 \geq a_2 \geq \cdots \geq a_n$, $h \geq 0$, and an integer $k \leq n$, the linear system*

$$\begin{aligned} a_1 x_1 + \cdots + a_n x_n &= h \\ x_1 + \cdots + x_n &= k \\ 0 \leq x_i \leq 1 \quad &\forall i \in \{1, \ldots, n\} \end{aligned}$$

*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 $a_1 x_1 + \cdots + a_n x_n = h$, we check

$$\begin{aligned} & a_1 x_1 + \cdots + a_n x_n \\ =& \sum_{i \in \{1,\ldots,k\} \setminus \{n-k+1,n\}} a_i t_1 + \sum_{i \in \{1,\ldots,k\} \cap \{n-k+1,n\}} a_i (t_1 + t_2) + \sum_{i \in \{n-k+1,n\} \setminus \{1,\ldots,k\}} a_i t_2 \\ =& (a_1 + \cdots a_k)t_1 + (a_{n-k+1} + \cdots + a_n)t_2 = h. \end{aligned}$$

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

$$a_1 + \cdots + a_k \geq a_1 x_1 + \cdots a_n x_n \geq a_{n-k+1} + \cdots + a_n.$$

Therefore if either $a_1 + \cdots + a_k < h$ or $a_{n-k+1} + \cdots a_n > h$, we cannot satisfy the condition $a_1 x_1 + \cdots a_n x_n = h$. □

To find a fractional optimal solution of problem (10), We define

$$\Lambda = \left\{ \lambda \geq 0 : \lambda = \frac{a_j c_i - a_i c_j}{a_i - a_j}, \forall i, j \in N \right\} \cup \{0\}$$

as the set of $\lambda$'s that can be part of an optimal dual solution. Then for each $\lambda \in \Lambda$, let $(\lambda, h, I)$ be a tuple such that $I \subseteq N$ and for any index $i \in I$, $h_i(\lambda) = h$. Notice here for a $\lambda$ there might exist multiple corresponding tuples. By Lemma 9, if $\lambda > 0$, then $|I| \geq 2$. Let $\mathcal{C}$ be the collection of these tuples. In Algorithm 2, for each tuple $(\lambda, h, I)$, we assume that all the fractional components are among the indices set $I$, then we use Lemma 8 to fix values of variables whose indices not in $I$. We then solve a linear system using Lemma 10 or Lemma 11 to see if there exists a solution to satisfy condition (15). If the linear system has a solution, then we find a fractional optimal solution.

**Proposition 12.** *If problem (10) has a fractional optimal solution, Algorithm 2 finds it in polynomial time.*

*Proof.* First the set $\Lambda$ includes all possible $\lambda$'s by Lemma 9. Now consider a tuple $(\lambda, h, I)$ that the algorithm chooses to construct a fractional solution. First by Lemma 10 and Lemma 11, 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 (15), (13), and (14). For any $i \notin I$, the value of its primal and dual variables are determined by Lemma 8. 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 (12) always is met. Therefore the primal solution found by Algorithm 2 is optimal.

For the time complexity, computing the tuple collection $\mathcal{C}$ takes $O(n^2)$ time since the size of $\Lambda$ is $O(n^2)$. For each tuple, solving the linear system takes $O(n)$ time by Lemma 10 and Lemma 11. Therefore the total time of Algorithm 2 is polynomial. □

**Algorithm 2:** Find a fractional optimal solution of problem (10)

Compute the tuple collection $\mathcal{C} = \{(\lambda, h, H)\}$;
**for** $(\lambda, h, I) \in \mathcal{C}$ **do**
  $T \leftarrow \{j : h_j(\lambda) < h\}, x_j = 1, \forall j \in T$;
  $T' \leftarrow \{j : h_j(\lambda) > h\}, x_j = 0, \forall j \in T'$;
  **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 $\sum_{i\in I} a_i x_i + a(T) + d) = (f')^{-1}(h), \sum_{i\in I} x_i = k - |T|, x_i \in [0,1]$;
  If the linear system has a solution, **return** $x$;
// Problem (10) has no fractional optimal solution.

**Corollary 13.** *Problem (8) can be solved in polynomial time.*

*Proof.* First we use Algorithm 2 to search for a fractional optimal solution. If it fails to find a $\lambda$ that satisfies the KKT condition, then the optimal solution of problem (8) must be an integral one. Therefore we use Algorithm 1 to solve the problem. The time complexity easily follows from Proposition 4 and Proposition 12. □

**Corollary 14.** *For a set $S$, the lifted inequality (9) can be computed in polynomial time.*

*Proof.* To compute the lifting function $\gamma(\delta, k_0)$ in (9), first we need to find $k_0$ in Lemma 5 to ensure the subadditivity of lifting function $\gamma(\delta, k_0)$ as a function of $\delta$. By Lemma 5, $k_0 = \max(k_1, k_2)$. $k_1$ 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 13. 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 (8). Notice that for all $\gamma(a_i, k_0)$, the only thing changed in problem (8) is the value of $\delta$. Therefore given a seed inequality, we first compute the collection $\mathcal{C}$ in Algorithm 2, 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 13. □

### 2.4 Downlifting

The subadditive lifting inequality for downlifting in [1] is

$$w \leq f(a(S)) + \sum_{i\in S} \omega(-a_i)(1 - x_i) + \sum_{i\in N\backslash S} \rho_i(S) x_i, \tag{16}$$

where $\omega(\cdot)$ is a subadditive lifting function. Notice if we add cardinality constraint on $x$ as a constraint of the lifting function $\omega$, 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 paper.

## 3 Computational experiments

### 3.1 Problem and its data generation

In this section, we evaluate the effectiveness of our inequalities (9) by solving a problem of expected utility capital budgeting problem, which is the benchmark used in [1]. The problem can be stated as follows:

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

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 $\pi_i$, 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 $\lambda$. We reformulate the problem into a MILP setting by introducing a $w_i$ for each scenario and rewrite the problem as following:

$$1 + \max \left\{ \pi' w \,:\, a'x \leq 1, w_i \leq -\exp\left(-\frac{-v_i'x}{\lambda}\right), 1 \leq i \leq m, x \in \{0,1\}^N \right\}. \tag{17}$$

Then for each scenario $i$, the set $\left\{ (w_i, x) \in \mathbb{R} \times \{0,1\}^n \,:\, a'x \leq 1, w_i \leq -\exp(-\frac{-v_i'x}{\lambda}) \right\}$ is of the form (2).

We use exactly the same settings 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 $\alpha_j$ is from uniform distribution $[0.05, 0.1]$, $\beta_j$ is from uniform distribution $[0, 1]$, $\ln f_i$ is from normal distribution $N(0.05, 0.0025)$ and $\epsilon_{ij}$ is from normal distribution $N(0, 0.0025)$. Finally the scenarios are equally likely to occur with $\pi_i = 1/m$ for $i \in \{1, \ldots, m\}$.

Our implementation is 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 (2) directly. Nevertheless we build a mixed-integer linear program as a relaxation of (2) 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 (5) to compute a lifted inequality (9) 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 $S$. Following [1], we approximate $\rho_i(S \backslash i)$ by $\rho_i(\emptyset)$ and solve the following problem

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

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 $k(N, 0) = \max \{|T| \,:\, \sum_{i \in T} b_i \leq B\}$. We use a parametric linear optimization algorithm proposed in [2] and find an optimal solution in $O(n^3)$ time. Now after obtaining the seeding inequality, we apply Corollary 14 to calculate inequality (9).

For downlifting, the seed inequality for inequality (16) 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

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

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.

| $n$ | $m$ | $\lambda$ | Lifted inequalities in [1] | | | Inequality (9) and (16) | | |
|---|---|---|---|---|---|---|---|---|
| | | | 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 |

Table 1: Comparing lifted inequalities

### 3.2 Experiments

We choose number of variables ($n$), scenarios ($m$) and risk tolerance factor $\lambda$ 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 $\lambda$ since most of the values of $n, m, \lambda$ used in [1] are too easy for the current solver. We use our new uplifted inequality (9) and old downlifted inequality (16) from [1] to compare against the lifted inequalities developed in [1]. We do not consider the simplest inequalities (3) and (4) 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 1. For different $n, m$ and $\lambda$, 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 1 that the number of variables $n$, scenarios $m$, and the risk tolerance factor $\lambda$ all contribute to the overall performance. There is no single parameter dominating others. This is different from the observations made in [1] where $\lambda$ 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 (17) 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 paper, we do not report the details of this comparison.

In Figure 2, we present a performance profile of CPU time. Following Dolan and Moré in [3], the performance profile is constructed as follows. We have a set $\mathcal{S}$ of two solvers using inequality (9), (16) 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 calculate its performance ratio $r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}}$ where $t_{p,s}$ is the time required by solver $s$ on instance $p$. Figure 2 plots the cumulative distribution function of the performance

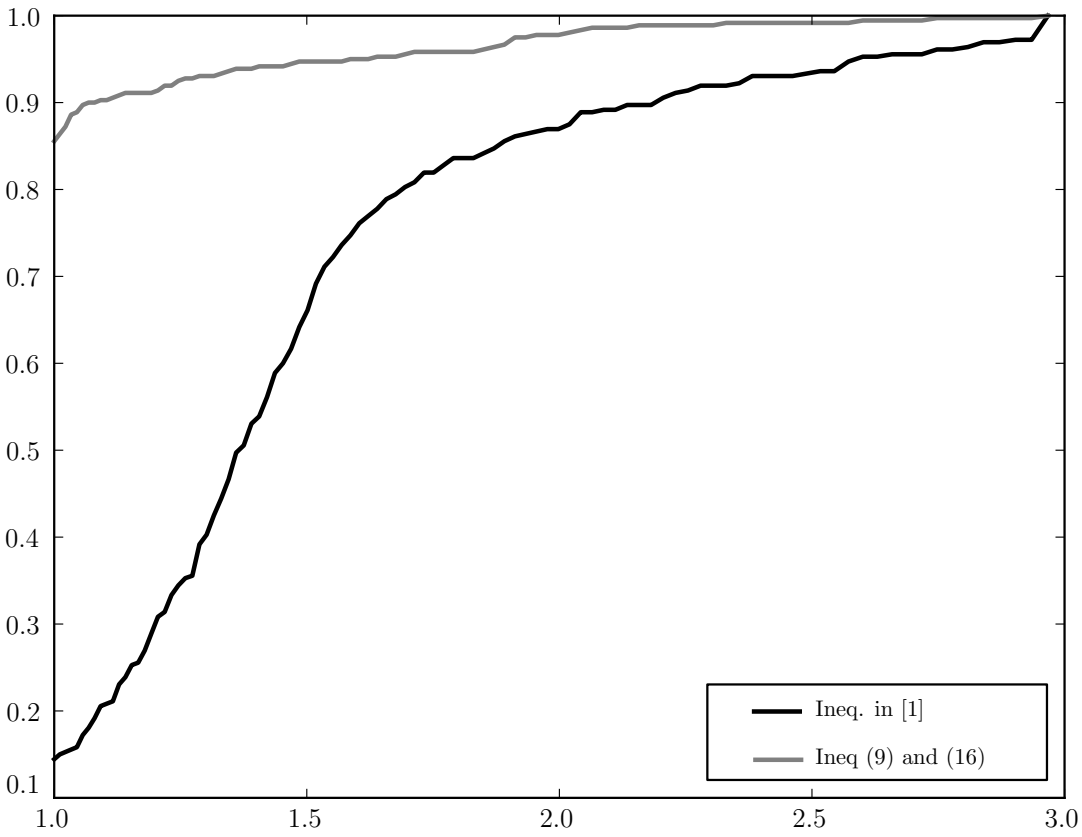

Figure 2: Performance profile for subadditive lifting function

ratio defined as $g_s(\tau) = \frac{1}{|\mathcal{P}|}\left|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}\right|$. We see the performance of inequalities (9) and (16) is significantly better than the two set of lifted inequalities in [1].

# 4 Conclusion

In this paper, we build a mixed-integer linear formulation of the mixed-integer non-linear set

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

by exploiting submodularity and knapsack structure in $P$. More specifically, we develop subadditive sequence independent lifted 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 Lagrange dual solutions 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 those proposed in [1] where the knapsack constraint is not considered.

# Acknowledgement

This research has been supported in part by the National Science Foundation grant # 1129871.

# References


[1] S. Ahmed and A. Atamtürk. Maximizing a class of submodular utility functions. *Mathematical Programming*, 128(1-2):149–169, 2011.

[2] A. Atamtürk and V. Narayanan. The submodular knapsack polytope. *Discrete Optimization*, 6(4):333 – 344, 2009.

[3] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[4] Z. Gu, G. L. Nemhauser, and M. W. Savelsbergh. Sequence independent lifting in mixed integer programming. *Journal of Combinatorial Optimization*, 4(1):109–129, 2000.

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

[6] J. Li and A. Deshpande. Maximizing expected utility for stochastic combinatorial optimization problems. In R. Ostrovsky, editor, *Proceedings of FOCS*, pages 797–806, 2011.

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

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

[9] L. A. Wolsey. Valid inequalities and superadditivity for 0-1 integer programs. *Mathematics of Operations Research*, 2(1):66–77, 1977.

[10] J. Yu and S. Ahmed. Maximizing expected utility over a knapsack constraint. Technical report, School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 2014. Available at http://www.optimization-online.org/DB_HTML/2012/10/3648.html.