Sell or Hold: A simple two-stage stochastic combinatorial optimization problem

Qie He, Shabbir Ahmed, George L. Nemhauser
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta, GA 30332

Aug 12, 2011

Abstract

There are n individual assets and k of them are to be sold over two stages. The first-stage prices are known and the second-stage prices have a known distribution. The sell or hold problem (SHP) is to determine which assets are to be sold at each stage to maximize the total expected revenue. We show that SHP is NP-hard when the second-stage prices are realized as a finite set of scenarios. We also identify a polynomially solvable case of SHP when the number of scenarios in the second stage is constant. A $\max\{\frac{1}{2}, \frac{k}{n}\}$-approximation algorithm is presented for the scenario-based SHP. An example shows that the bound is tight.

1 Introduction

An investor owns n indivisible assets and wants to sell k of them by the end of next year. The current market price for each asset is known. Next year's market price for each asset is random, but we assume that the distribution of the price vector is known in advance. The investor needs to decide which assets to sell this year. Then after having observed next year's price vector, he needs to decide which of the remaining assets to sell subject to at most k assets in total are sold. The investor's goal is to maximize the sum of the revenue obtained this year and the expected revenue of next year. We call this problem the sell or hold problem (SHP).

Stochastic programming [2] is widely used to deal with decision problems with uncertain data. When random parameters are introduced, many two-stage stochastic combinatorial optimization problems become NP-hard even if the original deterministic decision problems are easy. For example, the deterministic version of SHP is to decide which k out of n assets to sell to maximize the revenue, and the optimal strategy is simply to sell the k most expensive ones. Several two-stage stochastic combinatorial optimization problems have been studied in the literature, such as maximum weighted matching [8], shortest path, vertex cover, bin packing, facility location, set covering [11], steiner trees [5, 6], and spanning trees [3]. Various approximation algorithms based on techniques such as LP rounding, and the primal-dual method have been proposed.

For SHP, the first-stage decision is crucial. Once the investor decides which assets to sell this year, then next year he can adaptively sell the remaining ones according to the realized price vector. There are some special cases of SHP where the optimal strategy is trivial. If $k = n$, the investor eventually has to sell all the assets, so the optimal decision is to sell each asset this year whose current market price is greater than its expected price of next year, and sell the rest of the assets next year. Another trivial case is when the current price of every asset is less than its expected next-year price, then it is optimal to hold all assets until next year. The remainder of the paper is organized as follows. Section 2 introduces two equivalent formulations for SHP. Section 3 shows that SHP is NP-hard when the second-stage prices are discretely distributed. Section 4 identifies that SHP is polynomially solvable when the number of scenarios at the second stage is constant. A simple $\max\{\frac{1}{2}, \frac{k}{n}\}$-approximation algorithm and a tight example are presented in Section 5.

2 Two Formulations for SHP

2.1 A two-stage stochastic programming model

Let n be the total number of assets and k be the number of assets to sell. The current price for asset $i(1 \le i \le n)$ is r_i and the next-year price is $c_i(\omega)$, where $c(\omega) = (c_1(\omega), \ldots, c_n(\omega))$ is an n-dimensional random variable defined on a probability space $(\Omega, \mathcal{F}, P)$. WLOG, assume that r_i and $c_i(\omega)$ are all nonnegative. Let $x_i = 1$ ($x_i = 0$) denote the decision to sell (hold) asset i this year, and $y_i = 1$ ($y_i = 0$) denote the decision to sell (hold) asset i next year. Then SHP can be formulated as a two-stage stochastic integer programming problem:

$$\begin{array}{lll} \max & \sum_{i=1}^n r_i x_i + \mathbb{E}[Q(x, c(\omega))] & \\ \text{s.t.} & \sum_{i=1}^n x_i \le k, & \\ & x_i \in \{0,1\}, & i = 1, \ldots, n, \end{array} \tag{1}$$

where $Q(x, c(\omega))$ is the second-stage value function:

$$\begin{array}{rlll} Q(x, c(\omega)) = & \max & \sum_{i=1}^n c_i(\omega) y_i & \\ & \text{s.t.} & \sum_{i=1}^n y_i = k - \sum_{i=1}^n x_i, & \\ & & y_i \le 1 - x_i, & i = 1, \ldots, n, \\ & & y_i \in \{0,1\}, & i = 1, \ldots, n. \end{array} \tag{2}$$

Observe that in (2), the constraint matrix is totally unimodular (TU). Therefore, whenever x_i is integral for each i, the integrality restriction on y_i is redundant. Due to the constraint $y_i \le 1 - x_i$, the constraint $y_i \le 1$ is also redundant. Therefore,

$$\begin{array}{rlll} Q(x, c(\omega)) = & \max & \sum_{i=1}^n c_i(\omega) y_i & \\ & \text{s.t.} & \sum_{i=1}^n y_i = k - \sum_{i=1}^n x_i, & \\ & & 0 \le y_i \le 1 - x_i, & i = 1, \ldots, n. \end{array} \tag{3}$$

Note that $Q(x, c(\omega))$ is a monotone non-increasing concave function of x for every fixed $c(\omega)$ due to the strong duality theorem of linear programming.

2.2 A submodular maximization model

SHP can also be formulated as a non-monotone submodular maximization problem with a cardinality constraint. Let $S \subseteq N = \{1, \cdots, n\}$ denote the set of assets to sell at the first stage, the problem can be formulated as

$$\max\{f(S) : S \subseteq N, |S| \le k\}, \tag{4}$$

where $f(S) = \sum_{i \in S} r_i + \mathbb{E}[g(S, c(\omega))]$, where $g(S, c(\omega))$ is the optimal second-stage revenue when the assets in S are sold at the first stage and the second-stage price vector is $c(\omega)$.

Theorem 1. *$f(S)$ is a submodular function.*

Proof. Let $g_c(S)$ be the value of $g(S, c(\omega))$ when $c(\omega) = c$. We first show that $g_c(S)$ is a submodular function. We will show the following inequality holds for any $S \subseteq N$ and $q, r \in N \setminus S$:

$$g_c(S \cup \{r\}) - g_c(S) \ge g_c(S \cup \{r, q\}) - g_c(S \cup \{q\}). \tag{5}$$

Since no more than k items in total can be sold, $g_c(S)$ is well-defined only when $|S| \le k$. When $|S| \ge k+1$, we assign the values of $g_c(S)$ in the following way. When $|S| = k+1$, let $g_c(S) = \min_{r,q \in S}\{g_c(S \setminus \{r\}) + g_c(S \setminus \{q\}) - g_c(S \setminus \{r, q\})\}$. Then define the values of $g_c(S)$ sequentially for $|S| = k+2, k+3, \ldots, n$ in the same way. Therefore (5) is satisfied automatically when $|S| \ge k-1$ according to the way $g_c(S)$ is defined. When $|S| \le k-2$, WLOG assume that $S = \{1, \cdots, l\}$ for some $1 \le l \le k-2$. We sort the rest of the $(n-l)$ items according to their prices at the second stage in a nondecreasing order, i.e., $c_{l+1} \ge c_{l+2} \ge \ldots \ge c_n$. Consider the following four cases.

1. $q \geq k, r \geq k$. Observe that $g_c(S) = \sum_{i=l+1}^{k} c_i$ and $g_c(S \cup \{r\}) = \sum_{i=l+1}^{k-1} c_i$. Thus, $g_c(S \cup \{r\}) - g_c(S) = -c_k$. Similarly, $g_c(S \cup \{r, q\}) = \sum_{i=l+1}^{k-2} c_i$ and $g_c(S \cup \{q\}) = \sum_{i=l+1}^{k-1} c_i$. Thus, $g_c(S \cup \{r, q\}) - g_c(S \cup \{q\}) = -c_{k-1}$. Since $-c_k \geq -c_{k-1}$, (5) is satisfied.

2. $q \geq k, r \leq k$. Observe that $g_c(S \cup \{r\}) = \sum_{i=l+1, i \neq r}^{k} c_i$. Thus, $g_c(S \cup \{r\}) - g_c(S) = -c_r$. Similarly, $g_c(S \cup \{r, q\}) = \sum_{i=l+1, l \neq r}^{k-1} c_i$. Thus, $g_c(S \cup \{r, q\}) - g_c(S \cup \{q\}) = -c_r$. Therefore, (5) is satisfied.

3. $q \leq k, r \geq k$. Observe that $g_c(S \cup \{r, q\}) = \sum_{i=l+1, i \neq q}^{k-1} c_i$ and $g_c(S \cup \{q\}) = \sum_{i=l+1, i \neq q}^{k} c_i$. Thus, $g_c(S \cup \{r, q\}) - g_c(S \cup \{q\}) = -c_k$. While $g_c(S \cup \{r\}) - g_c(S) = -c_k$, (5) is satisfied.

4. $q \leq k, r \leq k$. Observe that $g_c(S \cup \{r\}) = \sum_{i=l+1, i \neq r}^{k} c_i$. Thus, $g_c(S \cup \{r\}) - g_c(S) = -c_r$. Similarly, $g_c(S \cup \{r, q\}) = \sum_{i=l+1, i \neq r, q}^{k} c_i$ and $g_c(S \cup \{q\}) = \sum_{i=l+1, i \neq q}^{k} c_i$. Thus, $g_c(S \cup \{r, q\}) - g_c(S \cup \{q\}) = -c_r$. Therefore, (5) is satisfied.

Therefore, (5) is satisfied for any $|S| \leq k-2$ and $q, r \in N \setminus S$.

Since integration preserves submodularity, $\mathbb{E}[g(S, c(\omega))] = \int_\Omega g(S, c(\omega)) dP(\omega)$ is submodular. The function $f(S)$ is the sum of a modular function and a submodular function, so it is also submodular. $\square$

Therefore, SHP can be formulated as a submodular maximization problem with a cardinality constraint. Notice that $f(S)$ is neither monotone nor symmetric. An intuitive explanation for non-monotonicity is that selling more assets at the first stage does not guarantee more or less revenue in total.

3 Complexity of SHP

Before addressing the complexity of SHP, we need to discuss how the input is represented, i.e., how to encode the uncertain information of the second-stage price $c(\omega)$. In the rest of the paper, we mainly consider the case where $c(\omega)$ is an n-dimensional discrete random variable with finite support. Assume that $c(\omega)$ could attain m values $\{c_j\}_{j=1}^m$ where $c_j = [c_{1j}, \ldots, c_{nj}]^T$ and $\Pr[c(\omega) = c_j] = p_j$ for $1 \leq j \leq m$. Let $y_{ij} = 1$ ($y_{ij} = 0$) denote the decision to sell (hold) asset i at the second stage when $c(\omega) = c_j$. We call this case the discrete sell or hold problem (DSHP). The expectation in (1) can be expressed as a finite sum and DSHP can be formulated as the integer programming problem

$$\begin{array}{lll} \max & \sum_{i=1}^n r_i x_i + \sum_{j=1}^m p_j \sum_{i=1}^n c_{ij} y_{ij} & \\ \text{s.t.} & \sum_{i=1}^n x_i + \sum_{i=1}^n y_{ij} = k, & j = 1, \ldots, m, \\ & x_i + y_{ij} \leq 1, & i = 1, \ldots, n,\ j = 1, \ldots, m, \\ & x_i \in \{0, 1\}, y_{ij} \geq 0, & i = 1, \ldots, n,\ j = 1, \cdots, m. \end{array} \tag{6}$$

Theorem 2. *DSHP is NP-hard.*

Proof. We show that DSHP is NP-hard by a polynomial reduction from the NP-hard uncapacitated facility location (UFL) problem. UFL is stated as follows. Let $F = \{1, \ldots, n\}$ be a set of facilities and $C = \{1, \ldots, m\}$ be a set of clients. Let f_i be the setup cost of opening facility i and d_{ij} be the transportation cost of assigning client j to facility i. The objective is to find a subset $I \subseteq F$ of facilities to open and a function $\phi : C \to I$ assigning clients to facilities such that the sum of setup cost and transportation cost is minimized.

Given an instance of UFL, we construct an instance of DSHP by letting $\{1, \ldots, n\} = F$ be the set of assets to sell and $\{1, \ldots, m\} = C$ be the set of scenarios at the second stage. Let $k = n-1$ be the maximum number of assets that can be sold. Set $p_j = \frac{1}{m}$, $c_{ij} = m d_{ij}$ and $r_i = f_i + \sum_{j=1}^m d_{ij}$. Observe that we will sell as many items as we can at the second stage since $c_{ij} \geq 0$, so when $k = n-1$, there will be exactly one item unsold at each scenario. Therefore, we can use a pair (I, ψ) to denote a feasible solution of DSHP, where $I \subseteq F$ and $\psi : C \to F \setminus I$. A solution (I, ψ) means that assets in I are sold at the first stage and all remaining assets but asset $\psi(j)$ are sold in scenario j at the second stage.
Claim: (I, ϕ) is an optimal solution of UFL if and only if $(F \setminus I, \phi)$ is an optimal solution of DSHP.

The objective value of $(F \setminus I, \phi)$ in DSHP is

$$
\begin{aligned}
&\sum_{i\in F\setminus I} r_i + \sum_{j=1}^{m} p_j (\sum_{i\in I} c_{ij} - c_{\phi(j)j}) \\
=&\sum_{i\in F\setminus I} (f_i + \sum_{j=1}^{m} d_{ij}) + \sum_{j=1}^{m} p_j [\sum_{i\in I} (md_{ij}) - md_{\phi(j)j}] \\
=&\sum_{i\in F} (f_i + \sum_{j=1}^{m} d_{ij}) - \sum_{i\in I} (f_i + \sum_{j=1}^{m} d_{ij}) + \sum_{j=1}^{m}\sum_{i\in I} d_{ij} - \sum_{j=1}^{m} d_{\phi(j)j} \\
=&\sum_{i\in F}(f_i + \sum_{j=1}^{m} d_{ij}) - [\sum_{i\in I} f_i + \sum_{j=1}^{m} d_{\phi(j)j}].
\end{aligned}
$$

The term $\sum_{i\in I} f_i + \sum_{j=1}^{m} d_{\phi(j)j}$ is exactly the objective value of the solution (I, ϕ) for UFL. Therefore, the objective value of the solution (I, ϕ) for UFL is minimum if and only if the objective value of the solution $(F \setminus I, \phi)$ for DSHP is maximum. $\square$

When $c(\omega)$ is a general random variable, the solution of SHP can be approximated by the solutions of a sequence of DSHP by the sample average approximation method [12]. Convergence is guaranteed as the number of samples goes to infinity. In fact, we can always aggregate scenarios such that the total number of scenarios is at most 2^n. Since the optimal second-stage decision for each scenario is to sell some of the most expensive remaining assets, its value only depends on the order of the second-stage prices $\{c_i(\omega)\}$. Therefore, we can aggregate those scenarios with the same price order into one scenario and adjust the associated parameters accordingly. Suppose that l scenarios have the same price order, then the aggregated scenario has probability $\sum_{j=1}^{l} p_j$, and the second-stage price of asset i in the aggregated scenario is $\frac{\sum_{j=1}^{l} p_j c_{ij}}{\sum_{j=1}^{l} p_j}$.

4 Polynomially solvable cases

In this section, we give some special cases of SHP that can be solved in polynomial time. We have shown that when $k = n$ the optimal decision is to compare the first-stage price and expected second-stage price of each asset and sell it at the higher price. When k is a constant, the number of possible first-stage decisions is $\binom{n}{k} = O(n^k)$, so the problem is polynomially solvable. Now we study the case when the number of scenarios m at the second stage is constant.

4.1 DSHP when $m = 2$

First consider $m = 2$, which has a special structure. By (6), the constraint is:

$$
\begin{bmatrix}
1 & & & & 1 & & & & & & & \\
& 1 & & & & 1 & & & & & & \\
& & \ddots & & & & \ddots & & & & & \\
& & & 1 & & & & 1 & & & & \\
1 & \dots & \dots & 1 & 1 & \dots & \dots & 1 & & & & \\
1 & \dots & \dots & 1 & & & & & 1 & \dots & \dots & 1 \\
1 & & & & & & & & 1 & & & \\
& 1 & & & & & & & & 1 & & \\
& & \ddots & & & & & & & & \ddots & \\
& & & 1 & & & & & & & & 1
\end{bmatrix}
\begin{bmatrix} x_1 \\ \vdots \\ x_n \\ y_{11} \\ \vdots \\ y_{n1} \\ y_{12} \\ \vdots \\ y_{n2} \end{bmatrix}
\leq
\begin{bmatrix} 1 \\ \vdots \\ 1 \\ k \\ k \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{7}
$$

Let A denote the constraint matrix. We name the constraints in (7) with right hand side k and 1 the k-constraints and 1-constraints, respectively. The fact that we have relaxed the two k-constraints to inequalities does not affect the optimal solutions due to the non-negativity of the prices. Let P_{LP} be the linear programming relaxation of (7). We first present a proposition regarding the constraint matrix of DSHP when $m = 2$.

Proposition 1. *The constraint matrix A is a network matrix.*

Proof. Construct a directed tree (N, A_1) and a digraph (N, A_2) as shown in Figure 1. The vertex set $N = \cup_{i=1}^{n} U_i \cup_{i=1}^{n} V_i \cup_{i=1}^{3} W_i$. We will show that A is an arc-dipath incidence matrix where the corresponding arcs are exactly arcs in A_1 and the paths are given by the endpoints of the arcs in A_2. Let arcs in A_1

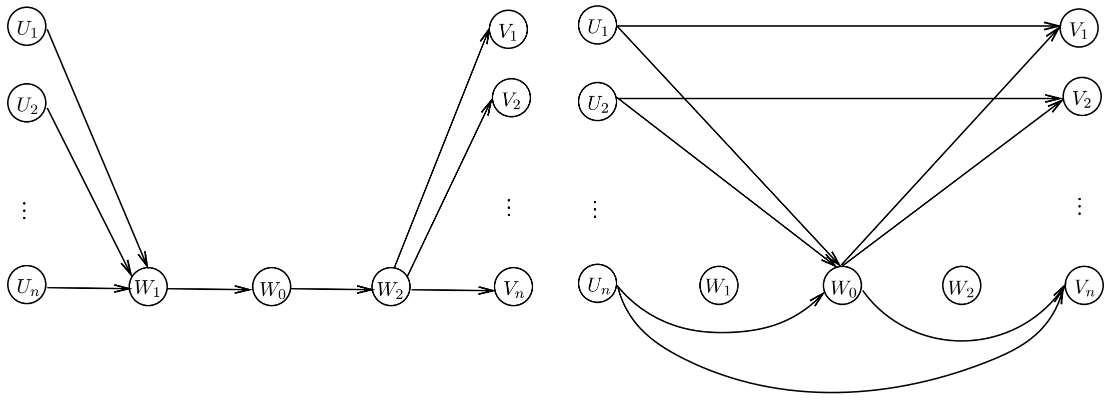

Figure 1: The digraph (N, A_1) and (N, A_2)

correspond to the rows of A. In detail, (U_i, W_1) correspond to the i-th row of A for $i = 1, \dots, n$, (W_1, W_0) and (W_0, W_2) correspond to the $(n+1)$-th and $(n+2)$-th row of A, and (W_2, V_i) correspond to $(n+2+i)$-th row of A for $i = 1, \dots, n$, respectively. Then each column of A is a characteristic vector of certain path whose endpoints are given by arcs in A_2. In detail, (U_i, V_i), (U_i, W_0) and (W_0, V_i) correspond to the i-th, $(n + i)$-th and $(2n + i)$-th columns of A for $i = 1, \dots, n$, respectively. Thus, A is a network matrix. $\square$

Corollary 1. *DSHP with $m = 2$ is solvable in polynomial time.*

Proof. Since A is a network matrix by Proposition 1 and the right-hand side of constraint (7) is integral, the polytope P_{LP} with constraints (7) with all variables in $[0, 1]$ has all of its extreme points integral. $\square$

Remark: A linear program $\max\{a^T x | Ax \le b, x \ge 0\}$ where A is a network matrix can be modeled as a network flow problem [10]. Thus we could solve DSHP when $m = 2$ by a network simplex algorithm. However, we are not aware of any simple combinatorial algorithm that directly solves DSHP when $m = 2$.

When $m \ge 3$ and $n \ge 3$, A is not TU, since it contains the following submatrix

$$\overline{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & & \\ 1 & 1 & 1 & & 1 & \\ 1 & 1 & 1 & & & 1 \\ 1 & & & 1 & & \\ & 1 & & & 1 & \\ & & 1 & & & 1 \end{bmatrix}.$$

The first three columns correspond to decision variables x_1, x_2 and x_3 at the first stage, and the last three columns correspond to decision variables y_{11}, y_{22} and y_{33} at the second stage. The first three rows correspond to the k-constraints for three different scenarios at the second stage, and the last three rows correspond to the 1-constraints for asset 1 at scenario 1, asset 2 at scenario 2 and asset 3 at scenario 3. Since $\det(\overline{A}) = 2$, A is not TU.

4.2 DSHP when m is constant

We present a technical lemma first.

Lemma 1. *The integer programming (IP) problem (P) : $\max\{a^T x | Bx \le b, x \in \{0,1\}^n\}$, where the matrix B has m rows and each entry of B is either 0, 1 or -1, can be solved in $O(n^{3^m})$ time.*

Proof. Since each entry of B is either 0 or ± 1, B has at most $M = 3^m$ different columns. Group the variables together if the corresponding columns in B are identical, then there are at most M groups. Suppose that there are n_l variables in the l-th group. Assume that in the objective function, the corresponding coefficients satisfy $a_{l_1} \ge a_{l_2} \ge \ldots \ge a_{l_{n_l}}$. Then any optimal solution of (P) satisfies the condition $x_{l_1} \ge x_{l_2} \ge \ldots \ge x_{l_{n_l}}$. Thus there are at most $n_l + 1$ possible values for these variables of any optimal solution, i.e., $x_{l_1} = \ldots = x_{l_{n_l}} = 0$, and $x_{l_1} = 1, x_{l_2} \ldots = x_{l_{n_l}} = 0$,..., and $x_{l_1} = \ldots = x_{l_{n_l}} = 1$. Therefore, the number of possible values for an optimal solution is bounded by

$$\prod_{l=1}^{M}(n_l + 1) \le (\frac{\sum_{l=1}^{M}(n_l + 1)}{M})^M = (1 + \frac{n}{M})^M.$$

The last equality follows from the fact that $\sum_{l=1}^{M} n_l = n$. Then the optimal solution can be found by enumeration in $O(n^M) = O(n^{3^m})$ time. □

Theorem 3. *DSHP is polynomially solvable when m is constant.*

Proof. The proof is divided into two steps: (a) DSHP can be decomposed into at most k^m IPs, (b) each IP is polynomially solvable.
From (6), once x is fixed, the problem can be decomposed into m optimization problems:

$$\begin{array}{rll} Q^j(x) = & \max & \sum_{i=1}^{n} c_{ij} y_{ij} \\ & \text{s.t.} & \sum_{i=1}^{n} y_{ij} = k - \sum_{i=1}^{n} x_i, \\ & & 0 \le y_{ij} \le 1 - x_i \qquad\qquad i = 1, \ldots, n. \end{array} \tag{8}$$

Problem (8) is a *fractional knapsack problem* with unit weight for each item. Suppose σ_j is a permutation of $\{1, \ldots, n\}$ such that $c_{\sigma_j(1)j} \ge c_{\sigma_j(2)j} \ge \ldots \ge c_{\sigma_j(n)j}$. Then the optimal solution of (8) is

$$y_{\sigma_j(i)j} = \begin{cases} 1 - x_{\sigma_j(i)} & i < l_j \\ k + 1 - l_j - \sum_{i=l_j}^{n} x_{\sigma_j(i)} & i = l_j \\ 0 & i > l_j \end{cases} \tag{9}$$

where $l_j = \min\{s | \sum_{i=1}^{s}(1 - x_{\sigma_j(i)}) \ge k - \sum_{i=1}^{n} x_i\}$. Therefore,

$$Q^j(x) = \sum_{i=1}^{n} c_{\sigma_j(i)j} y_{\sigma_j(i)j} = \sum_{i=1}^{l_j - 1} c_{\sigma_j(i)j}(1 - x_{\sigma_j(i)}) + c_{\sigma_j(l_j)j}(k + 1 - l_j - \sum_{i=l_j}^{n} x_{\sigma_j(i)})$$

The value of l_j depends on the value of x, and the possible values l_j could attain are $\{1, \ldots, k\}$. Hence from (8), $Q^j(x)$ is a concave piecewise linear function of x with at most k pieces, and the objective function of DSHP

$$F(x) = \sum_{i=1}^{n} r_i x_i + \sum_{j=1}^{n} p_j Q^j(x)$$

is a piecewise linear function of x with at most k^m pieces. Therefore, DSHP can be decomposed into at most k^m IPs.

The domain of each piece when $(l_1, \ldots, l_m) = (a_1, \ldots, a_m)$ is determined by

$$\begin{aligned} \sum_{i=1}^{a_j - 1}(1 - x_{\sigma_j(i)}) &< k - \sum_{i=1}^{n} x_j, \ j = 1, \ldots, m, \\ \sum_{i=1}^{a_j}(1 - x_{\sigma_j(i)}) &\ge k - \sum_{i=1}^{n} x_j, \ j = 1, \ldots, m. \end{aligned} \tag{10}$$

Since $F(x)$ is concave, it is continuous in the interior of $[0,1]^n$. Changing the sign from "$<$" to "$\leq$" in (10) will not affect the optimal value of DSHP. Then optimizing $F(x)$ over each piece is formulated as

$$\begin{array}{ll} \min & F(x) \\ \text{s.t.} & \sum_{i=1}^{a_j-1}(1-x_{\sigma_j(i)}) \leq k - \sum_{i=1}^n x_j, \quad j=1,\dots,m, \\ & \sum_{i=1}^{a_j}(1-x_{\sigma_j(i)}) \geq k - \sum_{i=1}^n x_j, \quad j=1,\dots,m, \\ & \sum_{i=1}^n x_i \leq k, \\ & x \in \{0,1\}^n. \end{array} \tag{11}$$

Problem (11) is an IP with $2m+1$ constraints, and each entry of the constraint matrix is either 0, 1 or -1. By Lemma 1, each IP can be solved in $O(n^{3^{2m+1}})$ time. Therefore, DSHP can be solved in $O(k^m n^{3^{2m+1}})$ time. When m is constant, DSHP is polynomially solvable. □

5 A $\max\{\frac{k}{n}, \frac{1}{2}\}$-approximation algorithm for DSHP

In Section 2 we have shown that SHP can be formulated as a submodular maximization problem with a cardinality constraint. Recently, there has been many results on approximation algorithm for submodular maximization [1, 4, 7, 9, 13]. These algorithms could be applied to our problem when they can deal with the additional cardinality constraint. However, these algorithms are designed for general submodular functions given by a value oracle, and therefore their performance is very weak. For example, it has been shown that non-monotone submodular maximization with a matroid independence constraint is hard to approximate to within a factor of $(\frac{1}{2}+\epsilon)$ for any fixed $\epsilon > 0$, unless P = NP [13]. In contrast, we explore the special structure of SHP and design a simple approximation algorithm which can achieve a better approximation ratio.
Let $\bar{c}_i = \mathbb{E}[c_i(\omega)]$ denote the expected price of asset i at the second stage.

Algorithm 1 Greedy heuristic 1

```
1: Set $\bar{r}_i = \max\{r_i, \bar{c}_i\}$ for each asset $i$.
2: Sort $\bar{r}_1, \dots, \bar{r}_n$ in nonincreasing order.
3: Let $\{i_1, \dots, i_k\}$ be the top $k$ assets in the list. If $r_{i_l} > \bar{c}_{i_l}$, then sell asset $i_l$ at the first stage, $l = 1, \dots, k$.
4: For the second stage of each scenario, sort the prices of the remaining assets in nonincreasing order, and
   sell the top $k - t$ assets, where $t$ is the number of assets sold at the first stage.
```

Proposition 2. *Algorithm 1 is a $\frac{k}{n}$-approximation algorithm for DSHP.*

Proof. Let $\text{OPT}(I)$ be the optimal objective value for the instance I, and $\text{ALG}_1(I)$ be the objective value of the solution produced by Algorithm 1. Then,

$$\text{ALG}_1(I) \geq \sum_{l=1}^k \bar{r}_{i_l} \geq \frac{k}{n}\sum_{i=1}^n \bar{r}_i \geq \frac{k}{n}\text{OPT}(I).$$

The last inequality follows from the fact $\sum_{i=1}^n \bar{r}_i$ is the optimal value of DSHP when $k = n$. □

A tight example for Algorithm 1 is obtained by letting the price for each asset at the first stage be $r+\epsilon$. The second stage has n scenarios, each with probability $\frac{1}{n}$. For scenario 1, the price for assets 1 to k is $\frac{n}{k}r$ and the rest of the assets all have price 0. For scenario 2, the price for assets 2 to $k+1$ is $\frac{n}{k}r$ and the rest of the assets all have price 0. For scenario j, the price for assets j to $(j+k)(\text{mod } n)$ is $\frac{n}{k}r$ and the rest of the assets have price 0. The expected price for each asset at the second stage is r, which is less than its first-stage price $r+\epsilon$. Thus the solution of Algorithm 1 is to sell k items at the first stage, and $\text{ALG}_1(I) = k(r+\epsilon)$. However, the optimal solution is to sell nothing at the first stage and sell the k most expensive assets at each scenario at the second stage. The optimal objective value $\text{OPT} = \sum \frac{1}{n}(\frac{n}{k}r \cdot k) = nr$.

Proposition 3. *Algorithm 2 is a $\frac{1}{2}$-approximation algorithm for DSHP.*

Algorithm 2 Greedy heuristic 2

```
1: Compute the profit $R_1$ of selling the $k$ most expensive assets in the first stage.
2: Compute the profit $R_2$ of selling the $k$ most expensive assets at each scenario at the second stage.
3: If $R_1 \geq R_2$, then sell the $k$ most expensive assets in the first stage, otherwise sell the $k$ most expensive
   assets at each scenario at the second stage.
```

Proof. Let $\mathrm{ALG}_2(I)$ be the objective value of the solution obtained by Algorithm 2. Then,

$$\mathrm{OPT}(I) = \text{ first-stage profit } + \text{ second-stage profit} \leq R_1 + R_2 \leq 2\mathrm{ALG}_2(I).$$

□

A tight example for Algorithm 2 is obtained by letting $k = 2$. The first-stage price of each asset is 0 except that the price for the first asset is r. The second stage has $n-1$ scenarios, each with probability $\frac{1}{n-1}$. For scenario j, the price of each asset is 0 except that the price for asset $j+1$ is r. Then in Algorithm 2, $R_1 = R_2 = r$, and $\mathrm{ALG}_2(I) = \max\{R_1, R_2\} = r$. However, the optimal solution is to sell asset 1 at the first stage, and sell asset $j+1$ at scenario j at the second stage. Then $\mathrm{OPT}(I) = 2r$.

Corollary 2. *If we take the better solution output by Algorithm 1 and Algorithm 2, the combined algorithm is a* $\max\{\frac{1}{2}, \frac{k}{n}\}$*-approximation algorithm.*

Acknowledgments

This work is supported by NSF grant CMMI-0758234 awarded to the Georgia Institute of Technology.

References

[1] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, *Maximizing a Submodular Set Function subject to a Matroid Constraint*, in IPCO XII, M. Fischetti and D. P. Williamson, eds., vol. 4513 of Lecture Notes in Computer Science, Springer, 2007, pp. 182–196.

[2] G. B. Dantzig, *Linear programming under uncertainty*, Management Science, 1 (1955), pp. 197–206.

[3] K. Dhamdhere, R. Ravi, and M. Singh, *On two-stage stochastic minimum spanning trees*, in IPCO XI, M. Jünger and V. Kaibel, eds., vol. 3509 of Lecture Notes in Computer Science, Springer, 2005, pp. 321–334.

[4] U. Feige, V. Mirrokni, and J. Vondrak, *Maximizing non-monotone submodular functions*, in Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2007, pp. 461–471.

[5] A. Gupta, R. Ravi, and A. Sinha, *An edge in time saves nine: LP rounding approximation algorithms for stochastic network design*, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2004, pp. 218–227.

[6] ——, *LP rounding approximation algorithms for stochastic network design*, Mathematics of Operations Research, 32 (2007), pp. 345–364.

[7] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar, *Constrained non-monotone submodular maximization: offline and secretary algorithms*, in Proceedings of the 6th international conference on Internet and Network Economics, Springer, 2010, pp. 246–257.

[8] N. Kong and A. J. Schaefer, *A factor 1/2 approximation algorithm for two-stage stochastic matching problems*, European Journal of Operational Research, 172 (2006), pp. 740–746.

[9] J. Lee, V. Mirrokni, V. Nagarajan, and M. Sviridenko, *Non-monotone submodular maximization under matroid and knapsack constraints*, in Proceedings of the 41st annual ACM Symposium on Theory of Computing, ACM, 2009, pp. 323–332.

[10] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*, Wiley-Interscience, New York, 1988.

[11] R. Ravi and A. Sinha, *Hedging uncertainty: Approximation algorithms for stochastic optimization problems*, Mathematical Programming, 108 (2006), pp. 97–114.

[12] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on stochastic programming: modeling and theory*, SIAM, Philadelphia, 2009.

[13] J. Vondrák, *Symmetry and approximability of submodular maximization problems*, in Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2010, pp. 651–670.