Complexity analysis and a fully polynomial-time approximation scheme for the college application problem

Max Kapur[1] and Sung-Pil Hong[2]

[1,2]Department of Industrial Engineering, Seoul National University

July 5, 2022

Abstract

We introduce the college application problem, a submodular optimization problem involving the maximization of the expected maximum value of a portfolio of random variables subject to a budget constraint. We show that this problem is NP-complete and provide two optimization algorithms based on dynamic programming. A modification of the second dynamic program yields a fully polynomial-time approximation scheme. These results suggest that the college application problem is a relatively easy instance of submodular maximization.

Keywords: submodular maximization, knapsack problems, approximation algorithms

Correspondence may be addressed to Max Kapur.

Email: maxkapur@gmail.com

Address: 39-411, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Republic of Korea

Contents

1 Introduction **2**
1.1 Literature review 2
1.2 Outline 3

2 Preliminaries **3**
2.1 The objective function 3
2.2 Submodularity 4

3 NP-completeness **4**

4 Optimization algorithms **6**
4.1 Dynamic program based on application expenditures 6
4.2 Dynamic program based on portfolio valuations 7
4.3 Fully polynomial-time approximation scheme 8

5 Conclusion **10**

6 References **10**

1 Introduction

This paper provides an NP-completeness proof and three solution algorithms for the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & v(\mathcal{X}) = \mathrm{E}\Big[\max\big\{t_0, \max\{t_j Z_j : j \in \mathcal{X}\}\big\}\Big] \\ \text{subject to} \quad & \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j \in \mathcal{X}} g_j \leq H \end{aligned} \tag{1}$$

Here $\mathcal{C} = \{1 \dots m\}$ is an index set; $H > 0$ is a budget parameter; for $j = 1 \dots m$, $g_j > 0$ is a cost parameter and Z_j is a random, independent Bernoulli variable with probability f_j; and for $j = 0 \dots m$, $t_j \geq 0$ is a utility parameter.

We refer to this problem as the *optimal college application* problem, as follows. Consider an admissions market with m colleges. Consider a prospective student in this market, and let each t_j-value indicate the utility she associates with attending school j, where her utility is t_0 if she does not attend college. Let g_j denote the application fee for school j and H the student's total budget to spend on application fees. Lastly, let f_j denote the student's probability of being admitted to school j if she applies, so that Z_j equals one if she is admitted and zero if not. The student's objective is to maximize the expected utility associated with the best school she is admitted to. Therefore, her optimal college application strategy is given by the solution $\mathcal{X}$ to the problem above, where $\mathcal{X}$ represents the set of schools to which she applies.

The problem is also conformable to other competitive matching games such as job application. Here, the budget constraint may represent the time needed to complete each application, or a legal limit on the number of applications permitted.

1.1 Literature review

The college application problem is, in a sense, a static variant of the Pandora's Box problem proposed by Weitzman [14]. In a Pandora's Box formulation of college application, the student applies to schools one by one, each time paying the application fee and observing her admissions outcome after a certain time delay. The problem is to determine an optimal *stopping rule* for when Pandora should halt her college search and accept the best admissions offer she has on hand. Weitzman showed that the optimal policy is to stop searching when the value of the current best offer exceeds the maximum *reservation price,* a statistic that represents the expected value of applying to a new college.

Arguably, the static model considered in this study is more hostile to students than the Pandora's Box problem. If, for example, an unlucky Pandora is rejected from a safety school at an early round of application, then she can compensate for the unexpected loss by pivoting to a more risk-averse application strategy. By contrast, the decisionmaker in our college application problem must commit at the outset to applying to every school in her application portfolio. The admissions process used in the United States can be viewed as the concatenation of both problems: In the fall, students solve (1) and send out a batch of applications. Then, upon observing their admissions outcomes in March, they use the Pandora strategy to pursue additional offers by applying to schools that offer rolling admissions.

The objective function of the college application problem is a nondecreasing, submodular set function in the sense first described in [12]. [11] showed that a polynomial-time algorithm for maximizing such a function over a knapsack constraint cannot achieve an approximation ratio better than $1 - 1/e \approx 0.632$. Subsequent research in submodular maximization has discovered efficient algorithms that achieve this approximation ratio using a variety of relaxation techniques [3, 1, 10].

As we will show, the college application problem is NP-complete, meaning that it cannot be solved in polynomial time unless P=NP. However, the existence of a fully polynomial-time approximation scheme (FPTAS) for the college application problem suggests that it is a relatively easy instance of submodular maximization. For a maximization problem, an FPTAS is defined as an algorithm that produces a solution whose objective value is at least $(1 - \varepsilon)$ times that of the optimum, in time polynomial in m and $1/\varepsilon$. Familiar examples of problems that admit an FPTAS include the knapsack problem, the constrained spanning tree problem, a constrained parallel scheduling problem on two machines, and the replenishment storage problem [13, 8, 15, 7]. The existence of an FPTAS for an optimization problem is a desirable property because an FPTAS induces a continuum of polynomial-time algorithms with constant approximation guarantees.

To the best of our knowledge, the first systematic study of the college application problem was conducted by Kapur in his master's thesis [9]. A special case of (1) arose in an equilibrium analysis of the United States college admissions market by Fu [5]. However, her econometric task involved clustering the schools such that $m = 8$, a problem size that can be solved by enumeration. We are interested in solution techniques that scale efficiently in the number of colleges. The present article extends the results of [9] with a more precise analysis of the problem's computational complexity and a new solution algorithm.

1.2 Outline

This paper has five sections. In section 2, we establish some notation and introduce assumptions that can be imposed with little loss of generality. In section 3, we prove that the college

application problem is NP-complete. Section 4 presents three optimization algorithms, including the FPTAS. A brief conclusion follows.

2 Preliminaries

For the remainder of the paper, we assume with minimal loss of generality that $f_j \in \mathbb{Q}$, $t_j \in \mathbb{N}$, $g_j \in \mathbb{N}$, and $H \in \mathbb{N}$; that the f_j-values have the same denominator D; and that $t_0 < t_1 \leq \cdots \leq t_m$, $g_j \leq H$, and $\sum g_j > H$. Unless otherwise noted, we assume that $t_0 = 0$, a restriction that we justify below.

2.1 The objective function

This subsection derives a closed-form expression for the objective function of (1).

We refer to the set $\mathcal{X} \subseteq \mathcal{C}$ of schools to which a student applies as her *application portfolio.* The expected utility the student receives from $\mathcal{X}$ is called its *valuation.* Given an application portfolio, let $p_j(\mathcal{X})$ denote the probability that the student attends school j. This occurs if and only if she *applies* to school j, is *admitted* to school j, and is *rejected* from any school she prefers to j; that is, any school with higher index. Hence, for $j = 0 \ldots m$,

$$p_j(\mathcal{X}) = \begin{cases} f_j \prod\limits_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i), & j \in \{0\} \cup \mathcal{X} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The following proposition follows by computing $v(\mathcal{X}) = \sum_{j=0}^{m} t_j p_j(\mathcal{X})$.

Proposition 1 (Closed form of portfolio valuation function)**.**

$$v(\mathcal{X}) = \sum_{j=0}^{m} t_j p_j(\mathcal{X}) = \sum_{j \in \{0\} \cup \mathcal{X}} \Big(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \Big) \tag{3}$$

Next, we show that without loss of generality, we may assume that $t_0 = 0$.

Lemma 1. *For some $\gamma \leq t_0$, let $\bar{t}_j = t_j - \gamma$ for $j = 0 \ldots m$. Then $v(\mathcal{X}; \bar{t}_j) = v(\mathcal{X}; t_j) - \gamma$ for any $\mathcal{X} \subseteq \mathcal{C}$.*

Proof. By definition, $\sum_{j=0}^{m} p_j(\mathcal{X}) = \sum_{j \in \{0\} \cup \mathcal{X}} p_j(\mathcal{X}) = 1$. Therefore

$$\begin{aligned} v(\mathcal{X}; \bar{t}_j) &= \sum_{j \in \{0\} \cup \mathcal{X}} \bar{t}_j p_j(\mathcal{X}) = \sum_{j \in \{0\} \cup \mathcal{X}} (t_j - \gamma) p_j(\mathcal{X}) \\ &= \sum_{j \in \{0\} \cup \mathcal{X}} t_j p_j(\mathcal{X}) - \gamma = v(\mathcal{X}; t_j) - \gamma \end{aligned} \tag{4}$$

which completes the proof. □

2.2 Submodularity

The following result is primarily of taxonomical interest and may be safely skipped. Our solution algorithms for the college application problem outperform generic algorithms for submodular maximization, and the proofs of their validity do not require submodular analysis.

Definition 1 (Submodular set function)**.** Given a ground set $\mathcal{C}$ and function $v : 2^{\mathcal{C}} \mapsto \mathbb{R}$, $v(\mathcal{X})$ is called a *submodular set function* if and only if $v(\mathcal{X}) + v(\mathcal{Y}) \geq v(\mathcal{X} \cup \mathcal{Y}) + v(\mathcal{X} \cap \mathcal{Y})$ for all $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{C}$. Furthermore, if $v(\mathcal{X} \cup \{k\}) - v(\mathcal{X}) \geq 0$ for all $\mathcal{X} \subset \mathcal{C}$ and $k \in \mathcal{C} \setminus \mathcal{X}$, $v(\mathcal{X})$ is said to be a *nondecreasing* submodular set function.

Theorem 1. *The college application portfolio valuation function $v(\mathcal{X})$ is a nondecreasing submodular set function.*

Proof. See [9], § 2.3). □

3 NP-completeness

In this section, we show that the college application problem is NP-complete by transformation from the binary knapsack problem. We begin by formulating the knapsack and college application problems as decision problems.

Definition 2 (Decision form of knapsack problem (KP))**. Instance:** a set $\mathcal{B}$ of m objects, utility values $u_j \in \mathbb{N}$ and weight $w_j \in \mathbb{N}$ for each $j \in \mathcal{B}$, knapsack capacity $W \in \mathbb{N}$, and target utility $U \in \mathbb{N}$. **Question:** Is there a set $\mathcal{B}' \subseteq \mathcal{B}$ having $\sum_{j \in \mathcal{B}'} u_j \geq U$ and $\sum_{j \in \mathcal{B}'} w_j \leq W$?

Theorem 2. *KP is NP-complete.*

Proof. See [6], § 3.2.1. □

Definition 3 (Decision form of the college application problem (CAP))**. Instance:** an instance (f, t, g, H) of the college application problem and a target valuation V. **Question:** Is there a a portfolio $\mathcal{X} \subseteq \mathcal{C}$ having $v(\mathcal{X}) \geq V$ and $\sum_{j \in \mathcal{X}} g_j \leq H$?

Theorem 3. *CAP is NP-complete.*

Proof. (CAP $\in$ NP.) It suffices to show that the cost and value of a given $\mathcal{X} \subseteq \mathcal{C}$ can be computed in polynomial time. Given an oracle that multiplies integers in unit time, $v(\mathcal{X})$ can be computed in polynomial time using the following algorithm (which assumes that the elements of $\mathcal{X}$ are iterated in index order):

$v \leftarrow 0$;
for $j \in \mathcal{X}$ **do**
  $v \leftarrow (1 - f_j)\, v + f_j t_j$;
end
return v;

The denominator of $v(\mathcal{X})$ grows rapidly as schools are added to $\mathcal{X}$, so we need to show that the number of bits of information required to store $v(\mathcal{X})$ is polynomial in the size of the input data. Let $K = \max\{\max t_j, \max g_j, D\}$ denote the largest integer in the input data. Then the instance data can be written in $O(Km)$-space, and $v(\mathcal{X})$ is a rational number whose numerator is at most $m \times K \times K^m = mK^{m+1}$, and whose denominator is at most K^m. Therefore, a verifier for $\mathcal{X}$ needs

$$O\big(\log(mK^{m+1})\big) \subset O\big(m \log Km\big)$$

bits of information to store $v(\mathcal{X})$, as required.

(KP $\propto$ CAP.) Consider an instance of the knapsack problem, and we will construct an instance of Problem 3 that is a yes-instance if and only if the corresponding knapsack instance

is a yes-instance. Without loss of generality, we may assume that the objects in $\mathcal{B}$ are indexed in increasing order of u_j, that each $u_j > 0$, and that each $w_j \leq W$.

Let $U_{\max} = \sum_{j \in \mathcal{B}} u_j$ and $\delta = 1/mU_{\max} \in (0, 1)$, and construct an instance of CAP with $\mathcal{C} = \mathcal{B}$, $H = W$, all $f_j = \delta$, and each $t_j = u_j/\delta$. Clearly, $\mathcal{X} \subseteq \mathcal{C}$ is feasible for CAP if and only if it is feasible for the knapsack instance. Now, we observe that for any nonempty $\mathcal{X}$,

$$\begin{aligned}\sum_{j \in \mathcal{X}} u_j &= \sum_{j \in \mathcal{X}} f_j t_j > \sum_{j \in \mathcal{X}} \Big(f_j t_j \prod_{\substack{j' \in \mathcal{X}: \\ j' > j}} (1 - f_{j'}) \Big) = v(\mathcal{X}) \\ &= \sum_{j \in \mathcal{X}} \Big(u_j \prod_{\substack{j' \in \mathcal{X}: \\ j' > j}} (1 - \delta) \Big) \geq (1 - \delta)^m \sum_{j \in \mathcal{X}} u_j \qquad (5) \\ &\geq (1 - m\delta) \sum_{j \in \mathcal{X}} u_j \geq \sum_{j \in \mathcal{X}} u_j - m\delta U_{\max} = \sum_{j \in \mathcal{X}} u_j - 1.\end{aligned}$$

This means that the utility of an application portfolio $\mathcal{X}$ in the corresponding knapsack instance is the smallest integer strictly greater than $v(\mathcal{X})$. That is, $\sum_{j \in \mathcal{X}} u_j \geq U$ if and only if $v(\mathcal{X}) \geq U - 1$. Taking $V = U - 1$ completes the transformation and concludes the proof. □

4 Optimization algorithms

In this section, we present three optimization algorithms for the college application problem. The first two are exact dynamic programming (DP) algorithms. The third algorithm is a fully polynomial-time approximation scheme (FPTAS) based on the second DP.

4.1 Dynamic program based on application expenditures

The first DP produces an optimal solution to the college application problem in $O(Hm + m \log m)$ time. It resembles a familiar DP algorithm for the binary knapsack problem (Dantzig 1957).

For $j = 0 \ldots m$ and $h = 0 \ldots H$, let $\mathcal{X}(j, h)$ denote the optimal portfolio that uses only the schools $\{1, \ldots, j\}$ and costs no more than h, and let $V(j, h) = v\big(\mathcal{X}(j, h)\big)$. Clearly, if $j = 0$ or $h = 0$, then $\mathcal{X}(j, h) = \emptyset$ and $V(j, h) = 0$. It is also convenient to let $V(j, h) = -\infty$ for all $h < 0$.

For the remaining indices, $\mathcal{X}(j, h)$ either contains j or not. If it does not contain j, then $\mathcal{X}(j, h) = \mathcal{X}(j - 1, h)$. On the other hand, if $\mathcal{X}(j, h)$ contains j, then its valuation is $(1 - f_j)v(\mathcal{X}[j, h] \setminus \{j\}) + f_j t_j$. Therefore, $\mathcal{X}(j, h) \setminus \{j\}$ must make optimal use of the remaining budget over the remaining schools; that is, $\mathcal{X}(j, h) = \mathcal{X}(j - 1, h - g_j) \cup \{j\}$. From these observations, we obtain the following recursion for $j = 1 \ldots m$ and $h = 1 \ldots H$:

$$V(j, h) = \max\big\{V(j - 1, h), (1 - f_j)V(j - 1, h - g_j) + f_j t_j\big\} \qquad (6)$$

with the convention that $-\infty \cdot 0 = -\infty$. Given the values of $V(j, h)$ at each index, the corresponding optimal portfolios are computed by observing that $\mathcal{X}(j, h)$ contains j if and only if $V(j, h) > V(j - 1, h)$. The optimal solution is given by $\mathcal{X}(m, H)$. The algorithm below performs these computations and outputs the optimal portfolio $\mathcal{X}$.

Theorem 4 (Validity of Algorithm 1)**.** *Algorithm 1 produces an optimal application portfolio for the college application problem in $O(Hm + m \log m)$ time.*

Proof. Optimality follows from the foregoing discussion. Sorting t is $O(m \log m)$. The bottleneck step is the creation of the lookup table for $V(j, h)$ in line 2. Each entry is generated in unit

Algorithm 1: Application expenditures DP for (1).

Input: $f \in \mathbb{Q}^m$, $t \in \mathbb{N}^m$, $g \in \mathbb{N}^m$, $H \in \mathbb{N}$.

```
Sort schools by t_j ascending;
Fill a lookup table with the values of V(j, h);
h ← H;
𝒳 ← ∅;
for j = m, m − 1, ..., 1 do
    if V(j − 1, h) < V(j, h) then
        𝒳 ← 𝒳 ∪ {j};
        h ← h − g_j;
    end
end
return 𝒳;
```

time, and the size of the table is $O(Hm)$. □

Because we cannot assume that $H \leq m$, Algorithm 1 represents a pseudopolynomial-time solution (Garey and Johnson 1979, § 4.2). However, it is quite effective for typical instances in which the application costs are small integers. It is also a polynomial-time algorithm for the special case of the college application problem in which each $g_j = 1$, meaning that H is a limit on the *cardinality* $\mathcal{X}$, and $H < \sum g_j = m$ in any nontrivial instance.

4.2 Dynamic program based on portfolio valuations

As with the knapsack problem, the college application problem admits a complementary DP that iterates on the value of the cheapest portfolio instead of on the cost of the most valuable portfolio.

Let $\mathbb{Z}/D$ denote the set of integer multiples of $1/D$. For integers $0 \leq j \leq m$ and $v \in \mathbb{Z}/D^m$, let $\mathcal{W}(j, v)$ denote the least expensive portfolio that uses only schools $\{1, \ldots, j\}$ and has valuation at least v, if such a portfolio exists. Denote its cost by $G(j, v) = \sum_{j \in \mathcal{W}(j,v)} g_j$, where $G(j, v) = \infty$ if $\mathcal{W}(j, v)$ does not exist. Clearly, if $v \leq 0$, then $\mathcal{W}(j, v) = \emptyset$ and $G(j, h) = 0$, and if $j = 0$ and $v > 0$, then $(j, h) = \infty$. For the remaining indices (where $j, v > 0$), we claim that

$$G(j,v) = \begin{cases} \infty, & t_j < v \\ \min\Big\{G(j-1,v),\, g_j + G\big(j-1, v-\Delta_j(v)\big)\Big\}, & t_j \geq v \end{cases} \tag{7}$$

$$\text{where} \qquad \Delta_j(v) = \begin{cases} \frac{f_j}{1-f_j}(t_j - v), & f_j < 1 \\ \infty, & f_j = 1. \end{cases} \tag{8}$$

In the $t_j < v$ case, any feasible portfolio must be composed of schools with utility less than v, and therefore its valuation can not equal v, meaning that $\mathcal{W}(j, v)$ is undefined. In the $t_j \geq v$ case, the first argument to min{} says simply that omitting j and choosing $\mathcal{W}(j-1, v)$ is a permissible choice for $\mathcal{W}(j, v)$. If, on the other hand, $j \in \mathcal{W}(j, v)$, then

$$v(\mathcal{W}[j,v]) = (1 - f_j)\, v(\mathcal{W}[j,v] \setminus \{j\}) + f_j t_j. \tag{9}$$

Therefore, the subportfolio $\mathcal{W}(j, v) \setminus \{j\}$ must have a valuation of $v - \Delta$, where Δ satisfies $v = (1 - f_j)(v - \Delta) + f_j t_j$. When $f_j < 1$, the solution to this equation is $\Delta = \frac{f_j}{1-f_j}(t_j - v)$.

When $t_j \geq v$ and $f_j = 1$, the singleton $\{j\}$ has $v(\{j\}) \geq v$, so

$$G(j, v) = \min\big\{G(j-1, v), g_j\big\}. \tag{10}$$

Defining $\Delta_j(v) = \infty$ in this case ensures that $g_j + G\big(j-1, v-\Delta_j(v)\big) = g_j + G(j-1, v-\infty) = g_j$ as required.

To compute the optimal portfolio, we first note that its valuation cannot exceed $v(\mathcal{C})$. Therefore, under our assumptions, the optimal portfolio valuation is an element of the finite set

$$\mathcal{V} = \Big\{0, 1/D^m, 2/D^m, \cdots, v(\mathcal{C}) - 1/D^m, v(\mathcal{C})\Big\}. \tag{11}$$

Therefore, if we recursively compute the values of $G(j, v)$ for each $j \in \mathcal{C}$ and $v \in \mathcal{V}$, the optimal portfolio valuation is simply the largest achievable objective value $\max\{w : G(m, w) \leq H\}$. The corresponding portfolio can be computed portfolio associated with $G(j, v)$ can be computed by applying the observation that $\mathcal{W}(j, v)$ contains j if and only if $G(j, v) < G(j-1, v)$.

Algorithm 2: Portfolio valuations DP for (1).

Input: $f \in \mathbb{Q}^m$, $t \in \mathbb{N}^m$, $g \in \mathbb{N}^m$, $H \in \mathbb{N}$.

1 Sort schools by t_j ascending;
2 Fill a lookup table with the entries of $G(j, h)$;
3 $v \leftarrow \max\{w \in \mathcal{V} : G(m, w) \leq H\}$;
4 $\mathcal{X} \leftarrow \emptyset$;
5 **for** $j = m, m-1, \ldots, 1$ **do**
6 **if** $G(j, v) < \infty$ *and* $G(j, v) < G(j-1, v)$ **then**
7 $\mathcal{X} \leftarrow \mathcal{X} \cup \{j\}$;
8 $v \leftarrow v - \Delta_j(v)$;
9 **end**
10 **end**
11 **return** $\mathcal{X}$;

Theorem 5 (Validity of Algorithm 2)**.** *Algorithm 2 produces an optimal solution to the college application problem in* $O\big(D^m v(\mathcal{C}) m^2\big)$ *time.*

Proof. Optimality is as discussed above. The computation time is proportional to the size of the table created in line 3, which is $m \times |\mathcal{V}| = m \times D^m \times v(\mathcal{C})$. □

4.3 Fully polynomial-time approximation scheme

The computation time of Algorithm 2 formidable, but a small modification yields an FPTAS for the college application problem that computes a $(1-\varepsilon)$-optimal solution in $O(m^3/\varepsilon)$ time.

To construct the FPTAS, we approximate each portfolio's valuation using a fixed-point binary number with a precision of P, where P is the number of digits to retain after the radix point. Let $r[x] = 2^{-P}\lfloor 2^P x \rfloor$ denote the value of x rounded down to its nearest fixed-point representation. To allow for approximation error, we will use the looser upper bound $\bar{U} = \sum_{j \in \mathcal{C}} f_j t_j > v(\mathcal{C})$ on the optimal portfolio valuation. Since we will ensure that each fixed-point approximation is an *underestimate* of the portfolio's true valuation, the set $\mathcal{V}'$ of valuations observable in the fixed-point framework remains finite:

$$\mathcal{V}' = \Big\{0, 1 \times 2^{-P}, 2 \times 2^{-P}, \ldots, r\big[\bar{U} - 1 \times 2^{-P}\big], r\big[\bar{U}\big]\Big\} \tag{12}$$

The details of the FPTAS are similar to those of Algorithm 2. For integers $0 \leq j \leq m$ and $v \in \mathcal{V}'$, $\mathcal{W}[j, v]$ denotes the least expensive portfolio that uses only schools $\{1, \ldots, j\}$ and has valuation of *at least* v, if such a portfolio exists. Its cost is given by $G[j, v] = \sum_{j \in \mathcal{W}[j,v]} g_j$. The boundary conditions and recursion relation are identical to those given above (subsection 4.2), except that $\Delta_j(v)$ is replaced with

$$\Delta_j[v] \begin{cases} r\left[\frac{f_j}{1-f_j}(t_j - v)\right], & f_j < 1 \\ \infty, & f_j = 1. \end{cases} \tag{13}$$

Here we have rounded the solution of (9) down to ensure that the true valuation of $\mathcal{W}[j, v]$ is *at least* $v - \Delta$, and therefore that $G[j, v]$ meets its definition.

Intuitively, $\max\{w \in \mathcal{V}' : G[m, w] \leq H\}$ gives the optimal portfolio valuation up to a certain amount of approximation error dependent on P. In Algorithm 3 and Theorem 6, we formalize this intuition by showing that $P \geq \log_2\left(m^2/\varepsilon\bar{U}\right)$ ensures an optimality ratio of $1 - \varepsilon$ or better.

Algorithm 3: FPTAS for (1).

Input: $f \in \mathbb{Q}^m$, $t \in \mathbb{N}^m$, $g \in \mathbb{N}^m$, $H \in \mathbb{N}$.
Parameters: Tolerance $\varepsilon \in (0, 1)$.

```
1  Sort schools by t_j ascending;
2  Set precision P ← ⌈log_2 (m^2/εŪ)⌉;
3  Fill a lookup table with the entries of G[j, h];
4  v ← max{w ∈ V : G[m, w] ≤ H};
5  X ← ∅;
6  for j = m, m − 1, ..., 1 do
7  |  if G[j, v] < ∞ and G[j, v] < G[j − 1, v] then
8  |  |  X ← X ∪ {j};
9  |  |  v ← v − Δ_j[v];
10 |  end
11 end
12 return X;
```

Theorem 6 (Validity of Algorithm 3)**.** *Algorithm 3 produces a $(1 - \varepsilon)$-optimal application portfolio for the college application problem in $O(m^3/\varepsilon)$ time.*

Proof. (Optimality.) Let $\mathcal{W}$ denote the output of Algorithm 3 and $\mathcal{X}$ the true optimum. We know that $v(\mathcal{X}) \leq \bar{U}$, and because each singleton portfolio is feasible, $\mathcal{X}$ must be more valuable than the average singleton portfolio; that is, $v(\mathcal{X}) \geq \sum f_j t_j / m = \bar{U}/m$.

Because $\Delta_j(v)$ is rounded down according to (13), if $j \in \mathcal{W}[j, v]$, the true value of $(1 - f_j)v\big(\mathcal{W}[j-1, v - \Delta_j(v)]\big) + f_j t_j$ may exceed the fixed-point valuation v of $\mathcal{W}[j, v]$, but not by more than 2^{-P}. This error accumulates additively with each school added to $\mathcal{W}$, but the number of additions is at most m. Therefore, where $v'(\mathcal{W})$ denotes the fixed-point valuation of $\mathcal{W}$ recorded in line 4 of the algorithm, $v(\mathcal{W}) - v'(\mathcal{W}) \leq m2^{-P}$.

We can define $v'(\mathcal{X})$ analogously as the fixed-point valuation of $\mathcal{X}$ when its elements are added in index order and its valuation is updated and rounded down to the nearest multiple of 2^{-P} at each addition in accordance with (9). By the same logic, $v(\mathcal{X}) - v'(\mathcal{X}) \leq m2^{-P}$. The optimality of $\mathcal{W}$ in the fixed-point environment implies that $v'(\mathcal{W}) \geq v'(\mathcal{X})$.

Applying these observations, we have

$$v(\mathcal{W}) \geq v'(\mathcal{W}) \geq v'(\mathcal{X}) \geq v(\mathcal{X}) - m2^{-P} \geq \left(1 - \frac{m^2 2^{-P}}{\bar{U}}\right) v(\mathcal{X}) \geq (1-\varepsilon)\, v(\mathcal{X}) \tag{14}$$

which establishes the approximation bound.

(Computation time.) The bottleneck step is the creation of the lookup table in line 3, whose size is $m \times |\mathcal{V}'|$. Since

$$|\mathcal{V}| = \bar{U} \times 2^P + 1 = \bar{U} \times 2^{\lceil \log_2(m^2/\varepsilon\bar{U}) \rceil} + 1 \leq \frac{m^2}{\varepsilon} \times \text{const.} \tag{15}$$

is $O(m^2/\varepsilon)$, the time complexity is as promised. □

Since its time complexity is polynomial in m and $1/\varepsilon$, Algorithm 3 is an FPTAS for the college application problem.

Algorithms 1, 2, and 3 can be written using recursive functions instead of lookup tables. However, since each function references itself *twice,* the function values at each index must be recorded in the computer's memory to take advantage of overlapping indices and guarantee the runtimes promised above. [9] discusses these and other implementation issues, and demonstrates the efficiency of Algorithms 1 and 3 through a computational study.

5 Conclusion

This article has introduced a novel discrete optimization problem called the college application problem. Our results show that the college application is a relatively easy instance of maximizing a nondecreasing submodular function over a knapsack constraint: The FPTAS presented here induces a continuum of tight, polynomial-time approximation algorithms, whereas the general problem admits no efficient algorithm with an approximation factor better than $1 - 1/e$.

When the application portfolio $\mathcal{X}$ is encoded as a binary vector x, the portfolio valuation function is given by the polynomial

$$v(x) = \sum_{j=1}^{m} \Big(x_j t_j f_j \prod_{i=j+1}^{m} (1 - f_i x_i) \Big). \tag{16}$$

One interpretation of the findings presented here is that this function, though nonlinear, has similar regularity properties to a linear function. Further evidence for this view is found in [9]'s analysis of the college application problem with a cardinality constraint, which can be solved by a simple greedy algorithm. We conjecture that other results associated with maximizing linear functions may hold with respect to the objective discussed here, such the existence of a PTAS when maximizing $v(\mathcal{X})$ over the intersection of knapsack constraints, and the optimality of the greedy algorithm when maximizing $v(\mathcal{X})$ over a polymatroidal system [2, 4].

6 References

References

[1] Ashwinkumar Badanidiyuru and Jan Vondrák. "Fast Algorithms for Maximizing Submodular Functions". In: *Proceedings of the 2014 Annual ACM–SIAM Symposium on Discrete Algorithms*. 2014, pp. 1497–1514. DOI: 10.1137/1.9781611973402.110.

[2] Chandra Chekuri and Sanjeev Khanna. "A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem". In: *SIAM Journal on Computing* 35.3 (2005), pp. 713–28. DOI: 10.1137/S0097539700382820.

[3] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. "Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes". In: *SIAM Journal on Computing* 43.6 (2014), pp. 1831–79. DOI: 10.1137/110839655.

[4] Jack Edmonds. "Matroids and the Greedy Algorithm". In: *Mathematical Programming* 1 (1971), pp. 127–36. DOI: 10.1007/BF01584082.

[5] Chao Fu. "Equilibrium Tuition, Applications, Admissions, and Enrollment in the College Market". In: *Journal of Political Economy* 122.2 (2014), pp. 225–81. DOI: 10.1086/675503.

[6] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman and Company, 1979.

[7] Dorit Hochbaum and Xu Rao. "A Fully Polynomial Time Approximation Scheme for the Replenishment Storage Problem". In: *Operations Research Letters* 48.6 (2020), pp. 835–39. DOI: 10.1016/j.orl.2020.10.004.

[8] Sung-Pil Hong, Sung-Jin Chung, and Bum Hwan Park. "A Fully Polynomial Bicriteria Approximation Scheme for the Constrained Spanning Tree Problem". In: *Operations Research Letters* 32.3 (2004), pp. 233–39. DOI: doi.org/10.1016/j.orl.2003.06.003.

[9] Max Kapur. "The College Application Problem". MA thesis. Seoul National University, 2022. URL: https://github.com/maxkapur/CollegeApplication.

[10] Ariel Kulik, Hadas Shachnai, and Tami Tamir. "Approximations for Monotone and Nonmonotone Submodular Maximization with Knapsack Constraints". In: *Mathematics of Operations Research* 38.4 (2013), pp. 729–39. DOI: doi.org/10.1287/moor.2013.0592.

[11] George Nemhauser and Laurence Wolsey. "Best Algorithms for Approximating the Maximum of a Submodular Set Function". In: *Mathematics of Operations Research* 3.3 (1978), pp. 177–88. DOI: doi.org/10.1287/moor.3.3.177.

[12] George Nemhauser, Laurence Wolsey, and Marshall Fisher. "An Analysis of Approximations for Maximizing Submodular Set Functions—I". In: *Mathematical Programming* 14 (1978), pp. 265–94. DOI: 10.1007/BF01588971.

[13] Vijay Vazirani. *Approximation Algorithms*. Berlin: Springer, 2001.

[14] Martin Weitzman. "Optimal Search for the Best Alternative". In: *Econometrica* 47.3 (1979), pp. 641–54. DOI: 10.2307/1910412.

[15] Gerhard Woeginger. "A Comment on Scheduling Two Parallel Machines with Capacity Constraints". In: *Discrete Optimization* 2.3 (2005), pp. 269–72. DOI: 10.1016/j.disopt.2005.06.005.