The College Application Problem

Max Kapur

Seoul National University
Department of Industrial Engineering
Management Science/Optimization Lab
Correspondence: maxkapur@gmail.com
Advisor: Sung-Pil Hong

June 13, 2022

Abstract

This paper considers the maximization of the expected maximum value of a portfolio of random variables subject to a budget constraint. We refer to this as the optimal college application problem. When each variable's cost, or each college's application fee, is identical, we show that the optimal portfolios are nested in the budget constraint, yielding an exact polynomial-time algorithm. When colleges differ in their application fees, we show that the problem is NP-complete. We provide four algorithms for this more general setup: a branch-and-bound routine, a dynamic program that produces an exact solution in pseudopolynomial time, a different dynamic program that yields a fully polynomial-time approximation scheme, and a simulated-annealing heuristic. Numerical experiments demonstrate the algorithms' accuracy and efficiency.

Keywords: submodular maximization, knapsack problems, approximation algorithms

요약

본 논문은 다수의 확률 변수로 구성된 포트폴리오의 기대 최댓값을 예산 조건 하에서 최대화하는 문제를 고려한다. 이를 대학 입학 지원 최적화 문제라고 부른다. 각 확률 변수의 비용, 즉 각 대학의 지원 비용이 동일한 경우, 최적 포트폴리오는 예산 제약식으로 결정된 포함 사슬 관계 성질을 가짐을 보이고 이를 바탕으로 다항 시간 해법을 제시한다. 대학의 지원 비용이 서로 다른 경우, 문제가 NP-complete함을 증명한다. 일반적인 문제를 위해 4가지 해법을 제시하며, 분지한계 기반 해법, 의사 다항 시간에 정확한 해를 출력하는 동적 계획 해법, 다른 동적 계획 기반으로 완전 다항 시간 근사 해법(fully polynomial-time approximation scheme), 그리고 모의 담금질(simulated annealing)을 이용한 휴리스틱 해법을 포함한다. 수리적 실험을 통해 알고리즘의 정확도와 효율성을 보인다.

본 논문은 https://github.com/maxkapur/CollegeApplication에서 전문을 한글로 공유한다.

Contents

1 Introduction

This paper considers 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. The jth college is named c_j. Consider a single prospective student in this market, and let each t_j-value indicate the utility she associates with attending c_j, where her utility is t_0 if she does not attend college. Let g_j denote the application fee for c_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 c_j if she applies, so that Z_j equals one if she is admitted and zero if not. It is appropriate to assume that the Z_j are statistically independent as long as f_j are probabilities estimated specifically for this student (as opposed to generic acceptance rates). Then the student's objective is to maximize the expected utility associated with the best school she gets into within this budget. 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 college application problem is not solely of theoretical interest. Due to the widespread perception that career earnings depend on college admissions outcomes, the solution of (1) holds monetary value. In the American college consulting industry, students pay private consultants an average of $200 per hour for assistance in preparing application materials, estimating their admissions odds, and identifying target schools (Sklarow 2018). In Korea, an important revenue stream for admissions consulting firms such as Megastudy (megastudy.net) and Jinhak (jinhak.com) is "mock application" software that claims to use artificial intelligence to optimize the client's application strategy. However, although predicting admissions outcomes on a school-by-school basis is a standard benchmark for logistic regression models (Acharya et al. 2019), we believe our study is the first to focus on the overall application strategy as an optimization problem.

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 Methodological orientation

The college application problem straddles several methodological universes. Its stochastic nature recalls classical portfolio allocation models. However, the knapsack constraint renders the problem NP-complete, and necessitates combinatorial solution techniques. We also observe that the objective function is also a submodular set function, although our approximation results suggest that college application is a relatively easy instance of submodular maximization.

A problem similar to (1) arose in an equilibrium analysis of the American college market by Fu (2014), who described college application as a "nontrivial portfolio problem" (226). In computational finance, traditional portfolio allocation models weigh the sum of expected profit across all assets against a risk term, yielding a concave maximization problem with linear constraints

(Markowitz 1952; Meucci 2005). But college applicants maximize the expected value of their *best* asset: If a student is admitted to her jth choice, then she is indifferent as to whether she gets into her $(j+1)$th choice. As a result, student utility is *convex* in the utility associated with individual applications. Risk management is implicit in the college application problem because, in a typical admissions market, college preferability correlates inversely with competitiveness. That is, students negotiate a tradeoff between attractive, selective "reach schools" and less preferable "safety schools" where admission is a safer bet (Jeon 2015). Finally, the combinatorial nature of the college application problem makes it difficult to solve using the gradient-based techniques associated with continuous portfolio optimization. Fu estimated her equilibrium model (which considers application as a *cost* rather than a constraint) by clustering the schools so that $m = 8$, a scale at which enumeration is tractable. We pursue a more general solution.

The integer formulation of the college application problem can be viewed as a kind of binary knapsack problem with a polynomial objective function of degree m. Our branch-and-bound and dynamic programming algorithms closely resemble existing algorithms for knapsack problems (Martello and Toth 1990, § 2.5–6). In fact, by manipulating the admissions probabilities, the objective function can be made to approximate a linear function of the characteristic vector to an arbitrary degree of accuracy, a fact that we exploit in our NP-completeness proof. Previous research has introduced various forms of stochasticity to the knapsack problem, including variants in which each item's utility takes a known probability distribution (Steinberg and Parks 1979; Carraway et al. 1993) and an online context in which the weight of each item is observed after insertion into the knapsack (Dean et al. 2008). Our problem superficially resembles the preference-order knapsack problem considered by Steinberg and Parks and Carraway et al., but these models lack the college application problem's singular "maximax" form. Additionally, unlike those models, we do not attempt to replace the real-valued objective function with a preference order over *outcome distributions,* which introduces technical issues concerning competing notions of stochastic dominance (Sniedovich 1980). We take for granted the student's preferences over *outcomes* (as encoded in the t_j-values), and focus instead on an efficient computational approach to the well-defined problem above.

We take special interest in the validity of greedy optimization algorithms, such as the algorithm that iteratively adds the school that elicits the greatest increase in the objective function until the budget is exhausted. Greedy algorithms produce a *nested* family of solutions parameterized by the budget H: If $H \leq H'$, then the greedy solution for budget H is a subset of the greedy solution for budget H'. As Rozanov and Tamir (2020) remark, the knowledge that the optima are nested aids not only in computing the optimal solution, but in the implementation thereof under uncertain information. For example, in the United States, many college applications are due at the beginning of November, and it is typical for students to begin working on their applications during the prior summer because colleges reward students who tailor their essays to the target school. However, students may not know how many schools they can afford to apply to until late October. The nestedness property—or equivalently, the validity of a greedy algorithm—implies that even in the absence of complete budget information, students can begin to carry out the optimal application strategy by writing essays for schools in the order that they enter the optimal portfolio.

For certain classes of optimization problems, such as maximizing a submodular set function over a cardinality constraint, a greedy algorithm is known to be a good approximate solution and exact under certain additional assumptions (Fisher et al. 1978; Assad 1985). For other problems, notably the binary knapsack problem, the most obvious greedy algorithm can be made to perform arbitrarily poorly (Vazirani 2001). We show results for the college application

problem that mirror those for the knapsack problem: When each $g_j = 1$, the optimal portfolios are nested. This special case mirrors the centralized college application process in Korea, where there is no application fee, but students are allowed to apply to only three schools during the main admissions cycle. Unfortunately, the nestedness property does not hold in the general case, nor does the greedy algorithm offer any performance guarantee. Instead, we identify a fully polynomial-time approximation scheme (FPTAS) based on fixed-point arithmetic.

Finally, we remark that the objective function of (1) is a nondecreasing submodular function. However, our research employs more elementary analytical techniques, and our approximation results are tighter than those associated with generic submodular maximization algorithms. For example, a well-known result of Fisher et al. (1978) implies that the greedy algorithm is asymptotically $(1 - 1/e)$-optimal for the $g_j = 1$ case of the college application problem, whereas we show that the same algorithm is exact. As for the general problem, an equivalent approximation ratio is achievable when maximizing a submodular set function over a knapsack constraint using a variety of relaxation techniques (Badanidiyuru and Vondrák 2014; Kulik et al. 2013). Indeed, $1 - 1/e$ is the highest approximation coefficient achievable by a polynomial-time algorithm for this problem (Nemhauser and Wolsey 1978). But in the college application problem, the existence of the FPTAS supersedes these results.

1.2 Structure of this paper

Section 2 introduces some additional notation and assumptions that can be imposed without loss of generality. We also introduce a useful variable-elimination technique and prove that the objective function is submodular.

In Section 3, we consider the special case where each $g_j = 1$ and H is an integer $h \leq m$. We show that an intuitive heuristic is in fact a $1/h$-approximation algorithm. Then, we show that the optimal portfolios are nested in the budget constraint, which yields an exact algorithm that runs in $O(hm)$ time.

In Section 4, we turn to the scenario in which colleges differ in their application fees. We show that the decision form of the portfolio optimization problem is NP-complete through a polynomial reduction from the binary knapsack problem. We provide four algorithms for this more general setup. The first is a branch-and-bound routine. The second is a dynamic program that iterates on total expenditures and produces an exact solution in pseudopolynomial time, namely $O(Hm + m \log m)$. The third is a different dynamic program that iterates on truncated portfolio valuations. It yields a fully polynomial-time approximation scheme that produces a $(1-\varepsilon)$-optimal solution in $O(m^3/\varepsilon)$ time. The fourth is a simulated-annealing heuristic algorithm that demonstrates strong performance in our synthetic instances.

In Section 5, we present the results of computational experiments that confirm the validity and time complexity results established in the previous two sections. We also investigate the empirical accuracy of the simulated-annealing heuristic.

In the conclusion, we identify possible improvements to our solution algorithms and introduce a few extensions of the model that capture the features of real-world admissions processes.

2 Notation and preliminary results

Before discussing the solution algorithms, we will introduce additional notation and a few preliminary results. For the remainder of the paper, unless otherwise noted, we assume with trivial loss of generality that each $g_j \leq H$, $\sum g_j > H$, each $f_j \in (0, 1]$, and $t_0 < t_1 \leq \cdots \leq t_m$.

2.1 Refining the objective function

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.*

Definition 1 (Portfolio valuation function)**.** $v(\mathcal{X}) = \mathrm{E}\left[\max\left\{t_0, \max\{t_j Z_j : j \in \mathcal{X}\}\right\}\right]$.

It is helpful to define the random variable $X = \max\{t_j Z_j : j \in \mathcal{X}\}$ as the utility achieved by the schools in the portfolio, so that when $t_0 = 0$, $v(\mathcal{X}) = \mathrm{E}[X]$. Similar pairs of variables with script and italic names such as $\mathcal{Y}_h$ and Y_h will carry an analogous meaning.

Given an application portfolio, let $p_j(\mathcal{X})$ denote the probability that the student attends c_j. This occurs if and only if she *applies* to c_j, is *admitted* to c_j, and is *rejected* from any school she prefers to c_j; that is, any school with higher index. Hence, for $j = 0 \dots m$,

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

where the empty product equals one and $f_0 = 1$. The following proposition follows immediately.

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

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

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 $j = 0 \dots m$. Then $v(\mathcal{X}; \bar{t}_j) = v(\mathcal{X}; t_j) - \gamma$ regardless of $\mathcal{X}$.*

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 An elimination technique

Now, we present a variable-elimination technique that will prove useful throughout the paper.[1] Suppose that the student has already resolved to apply to c_k, and the remainder of her decision consists of determining which *other* schools to apply to. Writing her total application portfolio as $\mathcal{X} = \mathcal{Y} \cup \{k\}$, we devise a function $w(\mathcal{Y})$ that orders portfolios according to how well they "complement" the singleton portfolio $\{k\}$. Specifically, the difference between $v(\mathcal{Y} \cup \{k\})$ and $w(\mathcal{Y})$ is the constant $f_k t_k$.

To construct $w(\mathcal{Y})$, let $\tilde{t}_j$ denote the expected utility the student receives from school c_j *given* that she has been admitted to c_j and applied to c_k. For $j < k$ (including $j = 0$), this is

[1] We thank Yim Seho for pointing out this useful transformation.

$\tilde{t}_j = (1 - f_k)t_j + f_k t_k$; for $j > k$, this is $\tilde{t}_j = t_j$. This means that

$$v(\mathcal{Y} \cup \{k\}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \tilde{t}_j p_j(\mathcal{Y}). \tag{5}$$

The transformation to $\tilde{t}$ does not change the order of the t_j-values. Therefore, the expression on the right side of (5) is itself a portfolio valuation function. In the corresponding market, t is replaced by $\tilde{t}$ and $\mathcal{C}$ is replaced by $\mathcal{C} \setminus \{k\}$. To restore our convention that $t_0 = 0$, we obtain $w(\mathcal{Y})$ by taking $\bar{t}_j = \tilde{t}_j - \tilde{t}_0$ for all $j \neq k$ and letting

$$w(\mathcal{Y}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \bar{t}_j p_j(\mathcal{Y}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \tilde{t}_j p_j(\mathcal{Y}) - \tilde{t}_0 = v(\mathcal{Y} \cup \{k\}) - f_k t_k \tag{6}$$

where the second equality follows from Lemma 1. The validity of this transformation is summarized in the following theorem, where we write $v(\mathcal{X}; \bar{t})$ instead of $w(\mathcal{Y})$ to emphasize that $w(\mathcal{Y})$ is, in form, a portfolio valuation function.

Lemma 2 (Eliminate c_k)**.** *For $\mathcal{X} \subseteq \mathcal{C} \setminus \{k\}$, $v(\mathcal{X} \cup \{k\}; t) = v(\mathcal{X}; \bar{t}) + f_k t_k$, where*

$$\bar{t}_j = \begin{cases} (1 - f_k)t_j, & t_j \leq t_k \\ t_j - f_k t_k, & t_j > t_k. \end{cases} \tag{7}$$

Proof. It is easy to verify that (7) is the composition of the two transformations (from t to $\tilde{t}$, and from $\tilde{t}$ to $\bar{t}$) discussed above. □

The transformation (7) can be applied iteratively to accommodate the case where the student has already resolved to apply to multiple schools.

2.3 Submodularity of the objective

Now, we show that the portfolio valuation function is submodular. This result is primarily of taxonomical interest and may be safely skipped. Our solution algorithms for the college application problem are better than generic algorithms for submodular maximization, and we prove their validity using elementary analysis.

Definition 2 (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. It is self-evident that $v(\mathcal{X})$ is nondecreasing. To establish its submodularity, we apply proposition 2.1.iii of Nemhauser and Wolsey (1978) and show that

$$v(\mathcal{X} \cup \{j\}) - v(\mathcal{X}) \geq v(\mathcal{X} \cup \{j, k\}) - v(\mathcal{X} \cup \{k\}) \tag{8}$$

for $\mathcal{X} \subset \mathcal{C}$ and $j \neq k \in \mathcal{C} \setminus \mathcal{X}$. By Lemma 2, we can repeatedly eliminate the schools in $\mathcal{X}$ according to (7) to obtain a portfolio valuation function $w(\mathcal{Y})$ with parameter $\bar{t}$ such that

$w(\mathcal{Y}) = v(\mathcal{X} \cup \mathcal{Y}) + \text{const.}$ for any $\mathcal{Y} \subseteq \mathcal{C} \setminus \mathcal{X}$. Therefore, (8) is equivalent to

$$w(\{j\}) - w(\varnothing) \geq w(\{j,k\}) - w(\{k\}) \tag{9}$$

$$\iff \quad w(\{j\}) + w(\{k\}) \geq w(\{j,k\}) \tag{10}$$

$$\iff \quad \mathrm{E}[\,\bar{t}_j Z_j\,] + \mathrm{E}[\,\bar{t}_k Z_k\,] \geq \mathrm{E}\big[\max\{\bar{t}_j Z_j, \bar{t}_k Z_k\}\big] \tag{11}$$

which is plainly true. □

3 Homogeneous application costs

In this section, we focus on the special case in which each $g_j = 1$ and H is a natural number $h \leq m$. We show that an intuitive heuristic is in fact a $1/h$-approximation algorithm, then derive an exact polynomial-time solution algorithm. Applying Lemma 1, we assume that $t_0 = 0$ unless otherwise noted. Throughout this section, we will call the applicant Alma, and refer to the corresponding optimization problem as Alma's problem.

Problem 1 (Alma's problem)**.** Alma's optimal college application portfolio is given by the solution to the following combinatorial optimization problem:

$$\begin{aligned} \text{maximize} \quad & v(\mathcal{X}) = \sum_{j \in \mathcal{X}} \Big(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \Big) \\ \text{subject to} \quad & \mathcal{X} \subseteq \mathcal{C}, \quad |\mathcal{X}| \leq h \end{aligned} \tag{12}$$

3.1 Approximation properties of a naïve solution

The expected utility associated with a single school c_j is simply $\mathrm{E}[t_j Z_j] = f_j t_j$. It is therefore tempting to adopt the following strategy, which turns out to be suboptimal.

Definition 3 (Naïve algorithm for Alma's problem)**.** Apply to the h schools having the highest expected utility $f_j t_j$.

This algorithm's computation time is $O(m)$ using the PICK algorithm of Blum et al. (1973).

The basic error of the naïve algorithm is that it maximizes $\mathrm{E}\,[\sum t_j Z_j\,]$ instead of $\mathrm{E}\,[\max\{t_j Z_j\}]$. The latter is what Alma is truly concerned with, since in the end she can attend only one school. In fact, the naïve algorithm is a $(1/h)$-approximation algorithm for Alma's problem, as expressed in the following theorem.

Theorem 2 (Accuracy of the naïve algorithm)**.** *When the application limit is h, let $\mathcal{X}_h$ denote the optimal portfolio, and $\mathcal{T}_h$ the set of the h schools having the largest values of $f_j t_j$. Then $v(\mathcal{T}_h)/v(\mathcal{X}_h) \geq 1/h$.*

Proof. Because $\mathcal{T}_h$ maximizes the quantity $\mathrm{E}\big[\sum_{j \in \mathcal{T}_h} \{t_j Z_j\}\big]$, we have

$$\begin{aligned} v(\mathcal{X}_h) &= \mathrm{E}\Big[\max_{j \in \mathcal{X}_h}\{t_j Z_j\}\Big] \leq \mathrm{E}\Big[\sum_{j \in \mathcal{X}_h} \{t_j Z_j\}\Big] \leq \mathrm{E}\Big[\sum_{j \in \mathcal{T}_h} \{t_j Z_j\}\Big] \\ &= h\,\mathrm{E}\Big[\tfrac{1}{h} \sum_{j \in \mathcal{T}_h} \{t_j Z_j\}\Big] \leq h\,\mathrm{E}\Big[\max_{j \in \mathcal{T}_h}\{t_j Z_j\}\Big] = h v(\mathcal{T}_h). \end{aligned} \tag{13}$$

where the final inequality follows from the concavity of the max{} operator. □

The following example establishes the tightness of the approximation factor.

Example 1. Pick any h and let $m = 2h$. For a small constant $\varepsilon \in (0, 1)$, define the market as follows.

j	1	$\cdots$	h	$h+1$	$h+2$	$\cdots$	$m-1$	m
f_j	1	$\cdots$	1	ε^1	ε^2	$\cdots$	ε^{h-1}	ε^h
t_j	1	$\cdots$	1	ε^{-1}	ε^{-2}	$\cdots$	$\varepsilon^{-(h-1)}$	ε^{-h}

Since all $f_j t_j = 1$, the naïve algorithm can choose $\mathcal{T}_h = \{1, \ldots, h\}$, with $v(\mathcal{T}_h) = 1$. But the optimal solution is $\mathcal{X}_h = \{h+1, \ldots, m\}$, with

$$v(\mathcal{X}_h) = \sum_{j=h+1}^{m} \Big(f_j t_j \prod_{j'=j+1}^{m} (1 - f_{j'}) \Big) = \sum_{j=1}^{h} (1-\varepsilon)^j \approx h. \tag{14}$$

Thus, as ε approaches zero, we have $v(\mathcal{T}_h)/v(\mathcal{X}_h) \to 1/h$. (The optimality of $\mathcal{X}_h$ follows from the fact that it achieves the upper bound of Theorem 13.)

Although the naïve algorithm is suboptimal, we can still find the optimal solution in $O(hm)$ time, as we will now show.

3.2 The nestedness property

It turns out that the solution to Alma's problem possesses a special structure: An optimal portfolio of size $h+1$ includes an optimal portfolio of size h as a subset.

Theorem 3 (Nestedness of optimal application portfolios)**.** *There exists a sequence of portfolios* $\{\mathcal{X}_h\}_{h=1}^m$ *satisfying the nestedness relation*

$$\mathcal{X}_1 \subset \mathcal{X}_2 \subset \cdots \subset \mathcal{X}_m \tag{15}$$

such that each $\mathcal{X}_h$ *is an optimal application portfolio when the application limit is* h*.*

Proof. By induction on h. Applying Lemma 1, we assume that $t_0 = 0$.

(Base case.) First, we will show that $\mathcal{X}_1 \subset \mathcal{X}_2$. To get a contradiction, suppose that the optima are $\mathcal{X}_1 = \{j\}$ and $\mathcal{X}_2 = \{k, l\}$, where we may assume that $t_k \le t_l$. Optimality requires that

$$v(\mathcal{X}_1) = f_j t_j > v(\{k\}) = f_k t_k \tag{16}$$

and

$$\begin{aligned} v(\mathcal{X}_2) = f_k(1-f_l)t_k + f_l t_l &> v(\{j,l\}) \\ &= f_j(1-f_l)t_j + (1-f_j)f_l t_l + f_j f_l \max\{t_j, t_l\} \\ &\ge f_j(1-f_l)t_j + (1-f_j)f_l t_l + f_j f_l t_l \\ &= f_j(1-f_l)t_j + f_l t_l \\ &\ge f_k(1-f_l)t_k + f_l t_l = v(\mathcal{X}_2) \end{aligned} \tag{17}$$

which is a contradiction.

(Inductive step.) Assume that $\mathcal{X}_1 \subset \cdots \subset \mathcal{X}_h$, and we will show $\mathcal{X}_h \subset \mathcal{X}_{h+1}$. Let $k = \arg\max\{t_k : k \in \mathcal{X}_{h+1}\}$ and write $\mathcal{X}_{h+1} = \mathcal{Y}_h \cup \{k\}$.

Suppose $k \notin \mathcal{X}_h$. To get a contradiction, suppose that $v(\mathcal{Y}_h) < v(\mathcal{X}_h)$ and $v(\mathcal{X}_{h+1}) > v(\mathcal{X}_h \cup \{k\})$. Then

$$\begin{aligned} v(\mathcal{X}_{h+1}) &= v(\mathcal{Y}_h \cup \{k\}) \\ &= (1 - f_k)v(\mathcal{Y}_h) + f_k t_k \\ &\leq (1 - f_k)v(\mathcal{X}_h) + f_k \,\mathrm{E}\big[\max\{t_k, X_h\}\big] \\ &= v(\mathcal{X}_h \cup \{k\}) \end{aligned} \tag{18}$$

is a contradiction.

Now suppose that $k \in \mathcal{X}_h$. We can write $\mathcal{X}_h = \mathcal{Y}_{h-1} \cup \{k\}$, where $\mathcal{Y}_{h-1}$ is some portfolio of size $h-1$. It suffices to show that $\mathcal{Y}_{h-1} \subset \mathcal{Y}_h$. By definition, $\mathcal{Y}_{h-1}$ (respectively, $\mathcal{Y}_h$) maximizes the function $v(\mathcal{Y} \cup \{k\})$ over portfolios of size $h-1$ (respectively, h) that do not include k. That is, $\mathcal{Y}_{h-1}$ and $\mathcal{Y}_h$ are the optimal complements to the singleton portfolio $\{k\}$.

Applying Lemma 2, we eliminate c_k, transform the remaining t_j-values to $\bar{t}_j$ according to (7), and obtain a function $w(\mathcal{Y}) = v(\mathcal{Y} \cup \{k\}) - f_k t_k$ that grades portfolios $\mathcal{Y} \subseteq \mathcal{C} \setminus \{k\}$ according to how well they complement $\{k\}$. Since $w(\mathcal{Y})$ is itself a portfolio valuation funtion and $\bar{t}_0 = 0$, the inductive hypothesis implies that $\mathcal{Y}_{h-1} \subset \mathcal{Y}_h$, which completes the proof. □

3.3 Polynomial-time solution

Applying the result above yields an efficient greedy algorithm for the optimal portfolio: Start with the empty set and add schools one at a time, maximizing $v(\mathcal{X} \cup \{k\})$ at each addition. Sorting t is $O(m \log m)$. At each of the h iterations, there are $O(m)$ candidates for k, and computing $v(\mathcal{X} \cup \{k\})$ is $O(h)$ using (3); therefore, the time complexity of this algorithm is $O(h^2 m + m \log m)$.

We reduce the computation time to $O(hm)$ by taking advantage of the transformation from Lemma 2. Once school k is added to $\mathcal{X}$, we eliminate it from the set $\mathcal{C} \setminus \mathcal{X}$ of candidates, and update the t_j-values of the remaining schools according to (7). Now, the *next* school added must be the optimal singleton portfolio in the modified market. But the optimal singleton portfolio consists simply of the school with the highest value of $f_j \bar{t}_j$. Therefore, by updating the t_j-values at each iteration according to (7), we eliminate the need to compute $v(\mathcal{X})$ entirely. Moreover, this algorithm does not require the schools to be indexed in ascending order by t_j, which removes the $O(m \log m)$ sorting cost. Algorithm 1 outputs an array `X` of the h schools to which Alma should apply. The schools appear in the order of entry such that when the algorithm is run with $h = m$, the optimal portfolio of size h is given by $\mathcal{X}_h = \{\texttt{X[1]}, \ldots, \texttt{X[h]}\}$. The entries of the array `V` give the valuation thereof.

Theorem 4 (Validity of Algorithm 1)**.** *Algorithm 1 produces an optimal application portfolio for Alma's problem in $O(hm)$ time.*

Proof. Optimality follows from the proof of Theorem 3. At each of the h iterations of the main loop, finding the top school costs $O(m)$, and the t_j-values of the remaining $O(m)$ schools are each updated in unit time. Therefore, the overall time complexity is $O(hm)$. □

It is possible to store $\mathcal{C}$ as a binary max heap rather than an array. The heap is ordered according to the criterion $i \geq j \iff f_i t_i \geq f_j t_j$, and by draining the heap and reheapifying at the end of each iteration, the computation time remains $O(hm)$. However, in our numerical experiments, whose results are reported in Section 5, we found the array implementation to be

Algorithm 1: Optimal portfolio algorithm for Alma's problem.

Input: Utility values $t \in (0, \infty)^m$, admissions probabilities $f \in (0, 1]^m$, application limit $h \leq m$.

$\mathcal{C} \leftarrow \{1 \dots m\}$;
$\mathtt{X}, \mathtt{V} \leftarrow$ empty h-arrays;
for $i = 1 \dots h$ **do**
  $k \leftarrow \arg\max_{j \in \mathcal{C}} \{f_j t_j\}$;
  $\mathcal{C} \leftarrow \mathcal{C} \setminus \{k\}$;
  $\mathtt{X}[i] \leftarrow k$;
  if $i = 1$ **then** $\mathtt{V}[i] \leftarrow f_k t_k$ **else** $\mathtt{V}[i] \leftarrow \mathtt{V}[i-1] + f_k t_k$;
  for $j \in \mathcal{C}$ **do**
    if $t_j \leq t_k$ **then** $t_j \leftarrow (1 - f_k) t_j$ **else** $t_j \leftarrow t_j - f_k t_k$;
  end
end
return $\mathtt{X}, \mathtt{V}$

much faster, because it is possible to identify the entering school k as the utility parameters are updated, which all but eliminates the cost of the $\arg\max\{\}$ operation.

3.4 Diminishing returns to application

The nestedness property implies that Alma's expected utility is a discretely concave function of h.

Theorem 5 (Optimal portfolio valuation concave in h)**.** *For $h = 2 \dots (m-1)$,*

$$v(\mathcal{X}_h) - v(\mathcal{X}_{h-1}) \geq v(\mathcal{X}_{h+1}) - v(\mathcal{X}_h). \tag{19}$$

Proof. Applying Theorem 3, we write $\mathcal{X}_h = \mathcal{X}_{h-1} \cup \{j\}$ and $\mathcal{X}_{h+1} = \mathcal{X}_{h-1} \cup \{j, k\}$. By optimality, $v(\mathcal{X}_h) - v(\mathcal{X}_{h-1}) \geq v(\mathcal{X}_{h-1} \cup \{k\}) - v(\mathcal{X}_{h-1})$. By submodularity and nestedness, $v(\mathcal{X}_{h-1} \cup \{k\}) - v(\mathcal{X}_{h-1}) \geq v(\mathcal{X}_h \cup \{k\}) - v(\mathcal{X}_h) = v(\mathcal{X}_{h+1}) - v(\mathcal{X}_h)$. □

(An elementary proof is given in § A.1.) It follows that when $\mathcal{X}_h$ is the optimal h-portfolio for a given market, $v(\mathcal{X}_h)$ is $O(h)$. In other words, Alma's problem exhibits diminishing returns to the application budget. Example 1, in which $v(\mathcal{X}_h)$ can be made arbitrarily close to h, establishes the tightness of this bound.

3.5 A small example

Let us examine a fictional admissions market consisting of $m = 8$ schools. The school data, along with the optimal solutions for each $h \leq m$, appear in Table 1.

Below are shown the first several iterations of Algorithm 1. The values of f_j, t_j, and their product are recorded only for the schools remaining in $\mathcal{C}$. $f * t$, where $(f * t)_j = f_j t_j$, denotes the entrywise product of f and t. The school added at each iteration, underlined, is the one whose

$f_j t_j$-value is greatest.

$$
\begin{aligned}
\text{Iteration 1:} \quad C &= \{1,2,3,4,5,6,7,8\} \\
f &= \{0.39, 0.33, 0.24, 0.24, 0.05, 0.03, 0.1, 0.12\} \\
t &= \{200, 250, 300, 350, 400, 450, 500, 550\} \\
f * t &= \{78.0, 82.5, 72.0, \underline{84.0}, 20.0, 13.5, 50.0, 66.0\} \implies \mathcal{X}_3 = \{4\} \\
\text{Iteration 2:} \quad C &= \{1,2,3,5,6,7,8\} \\
f &= \{0.39, 0.33, 0.24, 0.05, 0.03, 0.1, 0.12\} \\
t &= \{152, 190, 228, 316, 366, 416, 466\} \\
f * t &= \{59.28, \underline{62.7}, 54.72, 15.8, 10.98, 41.6, 55.92\} \implies \mathcal{X}_3 = \{4,2\} \\
\text{Iteration 3:} \quad C &= \{1,3,5,6,7,8\} \\
f &= \{0.39, 0.24, 0.05, 0.03, 0.1, 0.12\} \\
t &= \{101.84, 165.3, 253.3, 303.3, 353.3, 403.3\} \\
f * t &= \{39.718, 39.672, 12.665, 9.099, 35.33, \underline{48.396}\} \implies \mathcal{X}_3 = \{4,2,8\} \\
&\dots \\
\text{Iteration 8:} \quad C &= \{6\}, f = \{0.03\}, t = \{177.622\}, f * t = \{\underline{5.329}\} \\
&\implies \mathcal{X}_3 = \{4,2,8,1,7,3,5,6\}
\end{aligned}
$$

The output of the algorithm is $\texttt{X} = [4,2,8,1,7,3,5,6]$, and the optimal h-portfolio consists of its first h entries. The "priority" column of Table 1 shows the inverse permutation of $\texttt{X}$, which is the minimum value of h for which the school is included in the optimal portfolio. Figure 1 shows the value of the optimal portfolio as a function of h. The concave shape of the plot suggests the result of Theorem 5.

Index j	**School** c_j	**Admit prob.** f_j	**Utility** t_j	**Priority**	$v(\mathcal{X}_h)$
1	Mercury University	0.39	200	4	230.0
2	Venus University	0.33	250	2	146.7
3	Mars University	0.24	300	6	281.5
4	Jupiter University	0.24	350	1	84.0
5	Saturn University	0.05	400	7	288.8
6	Uranus University	0.03	450	8	294.1
7	Neptune University	0.10	500	5	257.7
8	Pluto College	0.12	550	3	195.1

Table 1: College data and optimal application portfolios for a fictional market with $m = 8$ schools. By the nestedness property (Theorem 3), the optimal portfolio when the application limit is h consists of the h schools having priority h or less.

4 Heterogeneous application costs

Now we turn to the more general problem in which the constant g_j represents the *cost* of applying to c_j and the student, whom we now call Ellis, has a *budget* of H to spend on college applications. Applying Lemma 1, we assume $t_0 = 0$ throughout.

Problem 2 (Ellis's problem)**.** Ellis's optimal college application portfolio is given by the solution

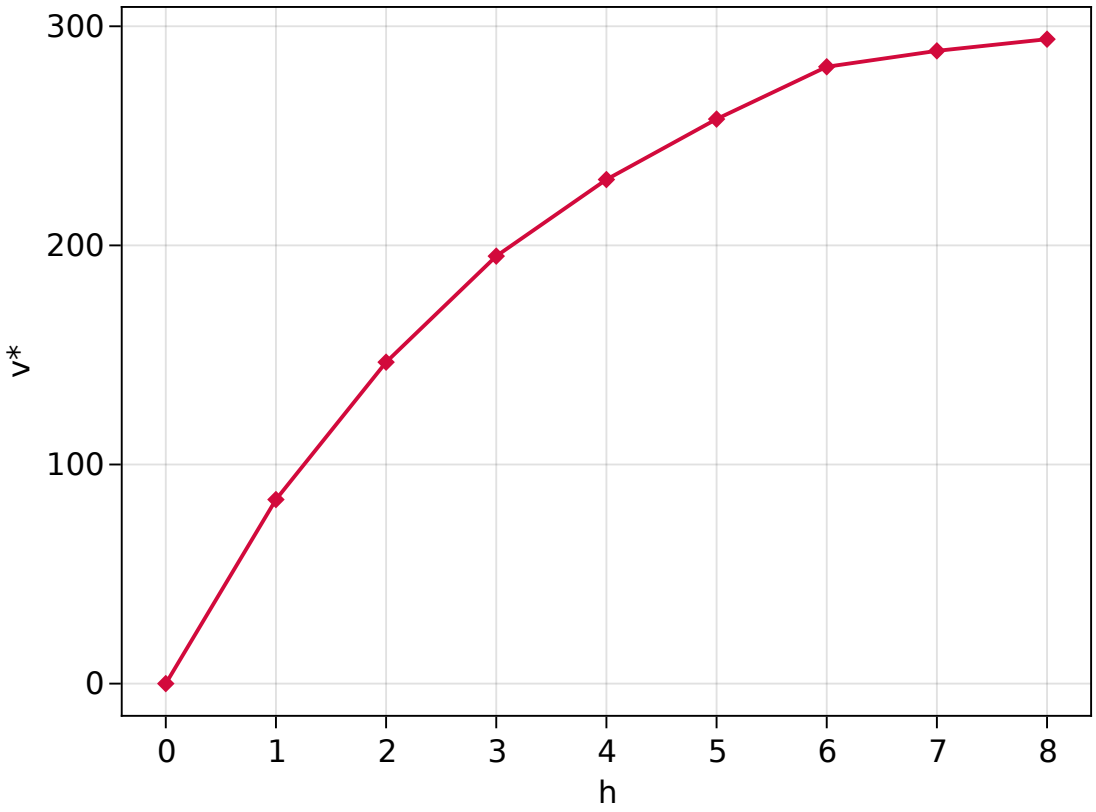

Figure 1: Application limit h versus the optimal portfolio valuation $v^* = v(\mathcal{X}_h)$ in a fictional market with $m = 8$ schools. The concave shape suggests the result of Theorem 5.

to the following combinatorial optimization problem.

$$\begin{aligned} \text{maximize} \quad & v(\mathcal{X}) = \sum_{j \in \mathcal{X}} \left(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right) \\ \text{subject to} \quad & \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j \in \mathcal{X}} g_j \leq H \end{aligned} \tag{20}$$

In this section, we show that this problem is NP-complete, then provide four algorithmic solutions: an exact branch-and-bound routine, an exact dynamic program, a fully polynomial-time approximation scheme (FPTAS), and a simulated-annealing heuristic.

4.1 NP-completeness

The optima for Ellis's problem are not necessarily nested, nor is the number of schools in the optimal portfolio necessarily increasing in H. For example, if $f = (0.5, 0.5, 0.5)$, $t = (1, 1, 219)$, and $g = (1, 1, 3)$, then it is evident that the optimal portfolio for $H = 2$ is $\{1, 2\}$ while that for $H = 3$ is $\{3\}$. In fact, Ellis's problem is NP-complete, as we will show by a transformation from the binary knapsack problem, which is known to be NP-complete (Garey and Johnson 1979, § 3.2.1).

Problem 3 (Decision form of knapsack problem)**.** An *instance* consists of a set $\mathcal{B}$ of m objects, utility values $u_j \in \mathbb{N}$ and weight $w_j \in \mathbb{N}$ for each $j \in \mathcal{B}$, and target utility $U \in \mathbb{N}$ and knapsack

capacity $W \in \mathbb{N}$. The instance is called a *yes-instance* if and only if there exists a set $\mathcal{B}' \subseteq \mathcal{B}$ having $\sum_{j\in\mathcal{B}'} u_j \geq U$ and $\sum_{j\in\mathcal{B}'} w_j \leq W$.

Problem 4 (Decision form of Ellis's problem)**.** An *instance* consists of an instance of Ellis's problem and a target valuation V. The instance is called a *yes-instance* if and only if there exists a portfolio $\mathcal{X} \subseteq \mathcal{C}$ having $v(\mathcal{X}) \geq V$ and $\sum_{j\in\mathcal{X}} g_j \leq H$.

Theorem 6. *The decision form of Ellis's problem is NP-complete.*

Proof. It is obvious that the problem is in NP.

Consider an instance of the knapsack problem, and we will construct an instance of Problem 4 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 $\delta = 1/mU_{\max} > 0$, and construct an instance of Ellis's problem 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 Ellis's problem if and only if it is feasible for the knapsack instance. Now, we observe that for any nonempty $\mathcal{X}$,

$$\begin{aligned}\sum_{j\in\mathcal{X}} u_j &= \sum_{j\in\mathcal{X}} f_j t_j > \sum_{j\in\mathcal{X}} \Big(f_j t_j \prod_{\substack{j'\in\mathcal{X}:\\ j'>j}} (1-f_{j'}) \Big) = v(\mathcal{X}) \\ &= \sum_{j\in\mathcal{X}} \Big(u_j \prod_{\substack{j'\in\mathcal{X}:\\ j'>j}} (1-\delta) \Big) \geq (1-\delta)^m \sum_{j\in\mathcal{X}} u_j \\ &\geq (1-m\delta) \sum_{j\in\mathcal{X}} u_j \geq \sum_{j\in\mathcal{X}} u_j - m\delta U_{\max} = \sum_{j\in\mathcal{X}} u_j - 1.\end{aligned} \tag{21}$$

This means that the utility of an application portfolio $\mathcal{X}$ in the corresponding knapsack instance is the smallest integer 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. □

An intuitive extension of the greedy algorithm for Alma's problem is to iteratively add to $\mathcal{X}$ the school k for which $\big(v(\mathcal{X} \cup \{k\}) - v(\mathcal{X})\big)/g_k$ is largest. But the construction above shows that the objective function of Ellis's problem can approximate that of a knapsack problem with arbitrary precision. Therefore, in pathological examples such as the following, the greedy algorithm can achieve an arbitrarily small approximation ratio.

Example 2. Let $t = (10, 2021)$, $f = (0.1, 0.1)$, $g = (1, 500)$, and $H = 500$. Then the greedy approximation algorithm produces the clearly suboptimal solution $\mathcal{X} = \{1\}$.

We might expect to obtain a better approximate solution to Ellis's problem by solving the following knapsack problem (nevermind that it is NP-complete) as a surrogate.

$$\text{maximize} \quad \sum_{j\in\mathcal{X}} f_j t_j \qquad \text{subject to} \quad \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j\in\mathcal{X}} g_j \leq H \tag{22}$$

However, upon inspection, this solution merely generalizes the naïve algorithm for Alma's problem (Definition 3). In Ellis's problem, because the number of schools in the optimal portfolio can be as large as $m - 1$, the knapsack solution's approximation coefficient is $1/(m-1)$ by analogy with (13). A tight example is as follows.

Example 3. Consider the following instance of Ellis's problem, where $H = m - 1$.

j	1	$\cdots$	$m-1$	m
f_j	1	$\cdots$	1	$\frac{1}{m-1}$
t_j	$\frac{1}{m-1}$	$\cdots$	$\frac{1}{m-1}$	$m-1$
g_j	1	$\cdots$	1	$m-1$

The feasible solutions $\mathcal{Y} = \{1, \ldots, m-1\}$ and $\mathcal{X} = \{m\}$ each have $\sum_{j\in\mathcal{Y}} f_j t_j = \sum_{j\in\mathcal{X}} f_j t_j = 1$, and thus the knapsack algorithm can choose $\mathcal{Y}$ with $v(\mathcal{Y}) = 1/(m-1)$. But the optimal solution is $\mathcal{X}$ with $v(\mathcal{X}) = 1$.

In summary, neither a greedy algorithm nor a knapsack-based algorithm solves Ellis's problem to optimality.

4.2 Branch-and-bound algorithm

A traditional approach to integer optimization problems is the branch-and-bound framework, which generates subproblems in which the values of one or more decision variables are fixed and uses an upper bound on the objective function to exclude, or *fathom,* branches of the decision tree that cannot yield a solution better than the best solution on hand (Martello and Toth 1990; Kellerer et al. 2004). In this subsection, we present an integer formulation of Ellis's problem and a linear program (LP) that bounds the objective value from above. We tighten the LP bound for specific subproblems by reusing the conditional transformation of the t_j-values from Algorithm 1. A branch-and-bound routine emerges naturally from these ingredients.

To begin, let us characterize the portfolio $\mathcal{X}$ as the binary vector $x \in \{0,1\}^m$, where $x_j = 1$ if and only if $j \in \{X\}$. Then it is not difficult to see that Ellis's problem is equivalent to the following integer nonlinear program.

Problem 5 (Integer NLP for Ellis's problem)**.**

$$\begin{aligned} \text{maximize} \quad & v(x) = \sum_{j=1}^{m} \Big(f_j t_j x_j \prod_{i>j} (1 - f_i x_i) \Big) \\ \text{subject to} \quad & \sum_{j=1}^{m} g_j x_j \le H, \quad x_i \in \{0,1\}^m \end{aligned} \tag{23}$$

Since the product in $v(x)$ does not exceed one, the following LP relaxation is an upper bound on the valuation of the optimal portfolio.

Problem 6 (LP relaxation for Ellis's problem)**.**

$$\begin{aligned} \text{maximize} \quad & v_{\text{LP}}(x) = \sum_{j=1}^{m} f_j t_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{m} g_j x_j \le H, \quad x \in [0,1]^m \end{aligned} \tag{24}$$

Problem 6 is a continuous knapsack problem, which is easily solved in $O(m \log m)$ time by a greedy algorithm (Dantzig 1957). Balas and Zemel (1980, § 2) provide an $O(m)$ algorithm.

In our branch-and-bound framework, a *node* is characterized by a three-way partition of schools $\mathcal{C} = \mathcal{I} \cup \mathcal{O} \cup \mathcal{N}$ satisfying $\sum_{j\in\mathcal{I}} g_j \le H$. $\mathcal{I}$ consists of schools that are "in" the application portfolio ($x_j = 1$), $\mathcal{O}$ consists of those that are "out" ($x_j = 0$), and $\mathcal{N}$ consists of those that are

"negotiable." The choice of partition induces a pair of subproblems. The first subproblem is an instance of Problem 5, namely

$$\begin{aligned} \text{maximize} \quad & v(x) = \gamma + \sum_{j \in \mathcal{N}} \Big(f_j \bar{t}_j x_j \prod_{\substack{i \in \mathcal{N}: \\ i > j}} (1 - f_i x_i) \Big) \\ \text{subject to} \quad & \sum_{j \in \mathcal{N}} g_j x_j \leq \bar{H}; \quad x_j \in \{0, 1\}, \quad j \in \mathcal{N}. \end{aligned} \tag{25}$$

The second is the corresponding instance of Problem 6:

$$\begin{aligned} \text{maximize} \quad & w_{\text{LP}}(x) = \gamma + \sum_{j \in \mathcal{N}} f_j \bar{t}_j x_j \\ \text{subject to} \quad & \sum_{j \in \mathcal{N}} g_j x_j \leq \bar{H}; \quad x_j \in [0, 1], \quad j \in \mathcal{N} \end{aligned} \tag{26}$$

In both subproblems, $\bar{H} = H - \sum_{j \in \mathcal{I}} g_j$ denotes the residual budget. The parameters γ and $\bar{t}$ are obtained by iteratively applying the transformation (7) to the schools in $\mathcal{I}$. For each $j \in \mathcal{I}$, we increment γ by the current value of $f_j \bar{t}_j$, eliminate j from the market, and update the remaining $\bar{t}_j$-values using (7). (An example is given below.)

Given a node $n = (\mathcal{I}, \mathcal{O}, \mathcal{N})$, its children are generated as follows. Every node has two, one, or zero children. In the typical case, we select a school $j \in \mathcal{N}$ for which $g_j \leq \bar{H}$ and generate one child by moving j to $\mathcal{I}$, and another child by moving j it to $\mathcal{O}$. Equivalently, we set $x_j = 1$ in one child and $x_j = 0$ in the other. In principle, any school can be chosen for j, but as a greedy heuristic, we choose the school for which the ratio $f_j \bar{t}_j / g_j$ is highest. Notice that this method of generating children ensures that each node's $\mathcal{I}$-set differs from its parent's by at most a single school, so the constant γ and transformed $\bar{t}_j$-values for the new node can be obtained by a single application of (7).

There are two atypical cases. First, if every school in $\mathcal{N}$ has $g_j > \bar{H}$, then there is no school that can be added to $\mathcal{I}$ in a feasible portfolio, and the optimal portfolio on this branch is $\mathcal{I}$ itself. In this case, we generate only one child by moving all the schools from $\mathcal{N}$ to $\mathcal{O}$. Second, if $\mathcal{N} = \emptyset$, then the node has zero children, and as no further branching is possible, the node is called a *leaf.*

Example 4. Consider a market in which $t = (20, 40, 60, 80, 100)$, $f = (0.5, 0.5, 0.5, 0.5, 0.5)$, $g = (3, 2, 3, 2, 3)$, and $H = 8$, and the node $n = (\mathcal{I}, \mathcal{O}, \mathcal{N}) = (\{2, 5\}, \{1\}, \{3, 4\})$. Let us compute the two subproblems associated with n and identify its children. To compute the subproblems, we first simply disregard c_1. Next, to eliminate c_2, we apply (7) to t to obtain $\{\bar{\bar{t}}_3, \bar{\bar{t}}_4, \bar{\bar{t}}_5\} = \{(1 - f_2)t_3, (1 - f_2)t_4, (1 - f_2)t_5\} = \{30, 40, 50\}$ and $\bar{\bar{\gamma}} = f_2 t_2 = 20$. We eliminate c_5 by again applying (7) to $\bar{\bar{t}}$ to obtain $\{\bar{t}_3, \bar{t}_4\} = \{\bar{\bar{t}}_3 - f_5 \bar{\bar{t}}_5, \bar{\bar{t}}_4 - f_5 \bar{\bar{t}}_5\} = \{5, 15\}$ and $\gamma = \bar{\bar{\gamma}} + f_5 \bar{\bar{t}}_5 = 35$. Finally, $\bar{H} = H - g_2 - g_5 = 3$. Now problems (25) and (26) are easily obtained by substitution.

Since at least one of the schools in $\mathcal{N}$ has $g_j \leq \bar{H}$, n has two children. Applying the node-generation rule, c_4 has the highest $f_j \bar{t}_j / g_j$-ratio, so the children are $n_1 = (\{2, 4, 5\}, \{1\}, \{3\})$ and $n_2 = (\{2, 5\}, \{1, 4\}, \{3\})$.

To implement the branch-and-bound algorithm, we represent the set of candidate nodes—namely, nonleaf nodes whose children have not yet been generated—by $\mathfrak{T}$. Each time a node $n = (\mathcal{I}, \mathcal{O}, \mathcal{N})$ is generated, we record the values $v_{\mathcal{I}}[n] = v(\mathcal{I})$ and $v^*_{\text{LP}}[n]$, the optimal objective value of the LP relaxation (26). Because $\mathcal{I}$ is a feasible portfolio, $v_{\mathcal{I}}[n]$ is a lower bound on the optimal objective value. Moreover, by the argument given in the proof of Theorem 3, the

objective function of (25) is identical to the function $v(\mathcal{I} \cup \mathcal{X})$. This means that $v^*_{\text{LP}}[n]$ is an upper bound on the valuation of any portfolio that contains $\mathcal{I}$ as a subset and does not include any school in $\mathcal{O}$, and hence on the valuation of any portfolio on this branch. Therefore, if upon generating a new node n_2, we discover that its objective value $v_{\mathcal{I}}[n_2]$ is greater than $v^*_{\text{LP}}[n_1]$ for some other node n_1, then we can disregard n_1 and all its descendants.

The algorithm is initialized by populating $\mathfrak{T}$ with the root node $n_0 = (\emptyset, \emptyset, \mathcal{C})$. At each iteration, it selects the node $n \in \mathfrak{T}$ having the highest $v^*_{\text{LP}}[n]$-value, generates its children, and removes n from the candidate set. Next, the children n' of n are inspected and added to the tree. If one of the children yields a new optimal solution, then we mark it as the best candidate and fathom any nodes n'' for which $v_{\mathcal{I}}[n'] > v^*_{\text{LP}}[n'']$. When no nodes remain in the candidate set, the algorithm has explored every branch, so it returns the best candidate and terminates.

Algorithm 2: Branch and bound for Ellis's problem.

Input: Utility values $t \in (0, \infty)^m$, admissions probabilities $f \in (0, 1]^m$, application costs $g \in (0, \infty)^m$, budget $H \in (0, \infty)$.

```
Root node n_0 ← (∅, ∅, C);
Current lower bound L ← 0 and best solution X ← ∅;
Candidate set T ← {n_0};
while not finished do
    if T = ∅ then return X, L;
    else
        n ← arg max{v*_LP[n] : n ∈ T};
        Remove n from T;
        for each child n' of n do
            if L < v_I[n'] then
                L ← v_I[n'];
                Update X to the I-set associated with n';
            end
            if v*_LP[n] > L and n' is not a leaf then add n' to T;
        end
        for n'' ∈ T do
            if L > v*_LP[n''] then remove n'' from T;
        end
    end
end
```

Theorem 7 (Validity of Algorithm 2)**.** *Algorithm 2 produces an optimal application portfolio for Ellis's problem.*

Proof. Because an optimal solution exists among the leaves of the tree, the discussion above implies that as long as the algorithm terminates, it returns an optimal solution.

To show that the algorithm does not cycle, it suffices to show that no node is generated twice. Suppose not: that two distinct nodes n_1 and n_2 share the same partition $(\mathcal{I}_{12}, \mathcal{O}_{12}, \mathcal{N}_{12})$. Trace each node's lineage up the tree and let n denote the *first* node at which the lineages meet. n must have two children, or else its sole child is a common ancestor of n_1 and n_2, and one of these children, say n_3, must be an ancestor of n_1 while the other, say n_4, is an ancestor of n_2. Write $n_3 = (\mathcal{I}_3, \mathcal{O}_3, \mathcal{N}_3)$ and $n_4 = (\mathcal{I}_4, \mathcal{O}_4, \mathcal{N}_4)$. By the node-generation rule, there is a school j in $\mathcal{I}_3 \cap \mathcal{O}_4$, and the $\mathcal{I}$-set (respectively, $\mathcal{O}$-set) for any descendant of $\mathcal{I}_3$ (respectively, $\mathcal{O}_4$) is a

superset of $\mathcal{I}_3$ (respectively, $\mathcal{O}_4$). Therefore, $j \in \mathcal{I}_{12} \cap \mathcal{O}_{12}$, meaning that $(\mathcal{I}_{12}, \mathcal{O}_{12}, \mathcal{N}_{12})$ is not a partition of $\mathcal{C}$, a contradiction. □

The branch-and-bound algorithm is an interesting benchmark, but as a kind of enumeration algorithm, its computation time grows rapidly in the problem size, and unlike the approximation scheme we propose later on, there is no guaranteed bound on the approximation error after a fixed number of iterations. Moreover, when there are many schools in $\mathcal{N}$, the LP upper bound may be much higher than $v_{\mathcal{I}}[n']$. This means that significant fathoming operations do not occur until the algorithm has explored deep into the tree, at which point the bulk of the computational effort has already been exhausted. In our numerical experiments, Algorithm 2 was ineffective on instances larger than about $m = 35$ schools, and therefore it does not represent a significant improvement over naïve enumeration. We speculate that the algorithm can be improved by incorporating cover inequalities (Wolsey 1998, § 9.3).

4.3 Pseudopolynomial-time dynamic program

In this subsection, we assume, with a small loss of generality, that $g_j \in \mathbb{N}$ for $j = 1 \dots m$ and $H \in \mathbb{N}$, and provide an algorithmic solution to Ellis's problem that runs in $O(Hm + m \log m)$ time. The algorithm resembles a familiar dynamic programming algorithm for the binary knapsack problem (Dantzig 1957; *Wikipedia*, s.v. "Knapsack problem"). Because we cannot assume that $H \leq m$ (as was the case in Alma's problem), this represents a pseudopolynomial-time solution (Garey and Johnson 1979, § 4.2). However, it is quite effective for typical college-application instances in which the application costs are small integers.

For $j = 0 \dots m$ and $h = 0 \dots H$, let $\mathcal{X}[j, h]$ denote the optimal portfolio using only the schools $\{1, \dots, j\}$ and costing no more than h, and let $V[j, h] = v(\mathcal{X}[j, h])$. It is clear that if $j = 0$ or $h = 0$, then $\mathcal{X}[j, h] = \emptyset$ and $V[j, h] = 0$. For convenience, we also define $V[j, h] = -\infty$ 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 $(1 - f_j)v(\mathcal{X}[j, h] \setminus \{j\}) + f_j t_j$. This requires that $\mathcal{X}[j, h] \setminus \{j\}$ 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 Bellman equation for $j = 1 \dots m$ and $h = 1 \dots H$:

$$V[j, h] = \max\big\{V[j - 1, h], (1 - f_j)V[j - 1, h - g_j] + f_j t_j\big\} \tag{27}$$

with the convention that $-\infty \cdot 0 = -\infty$. 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 8 (Validity of Algorithm 3)**.** *Algorithm 3 produces an optimal application portfolio for Ellis's 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 time, and the size of the table is $O(Hm)$. □

Algorithm 3: Dynamic program for Ellis's problem with integral application costs.

Input: Utility values $t \in (0, \infty)^m$, admissions probabilities $f \in (0, 1]^m$, application costs $g \in \mathbb{N}^m$, budget $H \in \mathbb{N}$.

Index schools in ascending order by t;
Fill a lookup table with the values of $V[j, h]$;
$h \leftarrow H$;
$\mathcal{X} \leftarrow \emptyset$;
for $j = m, m-1, \ldots, 1$ **do**
  if $V[j-1, h] < V[j, h]$ **then**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{j\}$;
    $h \leftarrow h - g_j$;
  end
end
return $\mathcal{X}$

4.4 Fully polynomial-time approximation scheme

As with the knapsack problem, Ellis's problem admits a complementary dynamic program that iterates on the value of the cheapest portfolio instead of on the cost of the most valuable portfolio. We will use this algorithm as the basis for an FPTAS for Ellis's problem that uses $O(m^3/\varepsilon)$ time and space.

We will represent approximate portfolio valuations using a fixed-point decimal with a precision of P, where P is the number of digits to retain after the decimal point. Let $r[x] = 10^{-P}\lfloor 10^P x \rfloor$ denote the value of x rounded down to its nearest fixed-point representation. Since $\bar{U} = \sum_{j \in \mathcal{C}} f_j t_j$ is an upper bound on the valuation of any portfolio, and 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 is finite:

$$\mathcal{V} = \left\{0, 1 \times 10^{-P}, 2 \times 10^{-P}, \ldots, r\left[\bar{U} - 1 \times 10^{-P}\right], r\left[\bar{U}\right]\right\} \tag{28}$$

Then $|\mathcal{V}| = \bar{U} \times 10^P + 1$.

For the remainder of this subsection, unless otherwise specified, the word *valuation* refers to a portfolio's valuation within the fixed-point framework, with the understanding that this is an approximation. We will account for the approximation error below when we prove the dynamic program's validity.

For integers $0 \le j \le m$ and $v \in [-\infty, 0) \cup \mathcal{V}$, 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. It is clear that if $v \le 0$, then $\mathcal{W}[j, v] = \emptyset$ and $G[j, h] = 0$, and that if $j = 0$ and $v > 0$, then $G[j, h] = \infty$. For the remaining indices (where $j, v > 0$), we claim that

$$G[j, v] = \begin{cases} \infty, & t_j < v \\ \min\left\{G[j-1, v], g_j + G[j-1, v - \Delta_j(v)]\right\}, & t_j \ge v \end{cases} \tag{29}$$

where

$$\Delta_j(v) = \begin{cases} r\left[\frac{f_j}{1-f_j}(t_j - v)\right], & f_j < 1 \\ \infty, & f_j = 1. \end{cases} \tag{30}$$

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

$$v(\mathcal{W}[j, v]) = (1 - f_j)v(\mathcal{W}[j, v] \setminus \{j\}) + f_j t_j. \tag{31}$$

Therefore, the subportfolio $\mathcal{W}[j, v] \setminus \{j\}$ must have a valuation of at least $v - \Delta$, where Δ satisfies $v = (1 - f_j)(v - \Delta) + f_j t_j$. When $f_j < 1$, the solution to this equation is $\Delta = \frac{f_j}{1-f_j}(t_j - v)$. By rounding this value down, we ensure that the true valuation of $\mathcal{W}[j, v]$ is *at least* $v - \Delta$. When $t_j \geq v$ and $f_j = 1$, the singleton $\{j\}$ has $v(\{j\}) \geq v$, so

$$G[j, v] = \min\big\{G[j-1, v], g_j\big\}. \tag{32}$$

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

Once $G[j, v]$ has been calculated at each index, the associated portfolio can be found by applying the observation that $\mathcal{W}[j, v]$ contains j if and only if $G[j, v] < G[j-1, v]$. Then an approximate solution to Ellis's problem is obtained by computing the largest achievable objective value $\max\{w : G[m, w] \leq H\}$ and corresponding portfolio.

Algorithm 4: Fully polynomial-time approximation scheme for Ellis's problem.

Input: Utility values $t \in (0, \infty)^m$, admissions probabilities $f \in (0, 1]^m$, application costs $g \in (0, \infty)^m$, budget $H \in (0, \infty)^m$.
Parameters: Tolerance $\varepsilon \in (0, 1)$.

1 Index schools in ascending order by t;
2 Set precision $P \leftarrow \lceil \log_{10}(m^2/\varepsilon\bar{U}) \rceil$;
3 Fill a lookup table with the entries of $G[j, h]$;
4 $v \leftarrow \max\{w \in \mathcal{V} : G[m, w] \leq H\}$;
5 $\mathcal{X} \leftarrow \emptyset$;
6 **for** $j = m, m-1, \ldots, 1$ **do**
7 **if** $G[j, v] < \infty$ *and* $G[j, v] < G[j-1, v]$ **then**
8 $\mathcal{X} \leftarrow \mathcal{X} \cup \{j\}$;
9 $v \leftarrow v - \Delta_j(v)$;
10 **end**
11 **end**
12 **return** $\mathcal{X}$

Theorem 9 (Validity of Algorithm 4)**.** *Algorithm 4 produces a* $(1 - \varepsilon)$*-optimal application portfolio for Ellis's problem in* $O(m^3/\varepsilon)$ *time.*

Proof. (Optimality.) Let $\mathcal{W}$ denote the output of Algorithm 4 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, $v(\mathcal{X}) \geq \bar{U}/m$.

Because $\Delta_j(v)$ is rounded down in the recursion relation defined by (29) and (30), if $j \in \mathcal{W}[j, v]$, then 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 10^{-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, $v(\mathcal{W}) - v'(\mathcal{W}) \leq m10^{-P}$.

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 10^{-P} at each addition in accordance with (31). By the same logic, $v(\mathcal{X}) - v'(\mathcal{X}) \leq m10^{-P}$. The optimality of $\mathcal{W}$ in the fixed-point environment implies that $v'(\mathcal{W}) \geq v'(\mathcal{X})$.

Applying these observations, we have

$$v(\mathcal{W}) \geq v'(\mathcal{W}) \geq v'(\mathcal{X}) \geq v(\mathcal{X}) - m10^{-P} \geq \left(1 - \frac{m^2 10^{-P}}{\bar{U}}\right) v(\mathcal{X}) \geq (1-\varepsilon)\, v(\mathcal{X}) \tag{33}$$

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

$$|\mathcal{V}| = \bar{U} \times 10^P + 1 = \bar{U} \times 10^{\left\lceil \log_{10}\left(m^2/\varepsilon\bar{U}\right)\right\rceil} + 1 \leq \frac{m^2}{\varepsilon} \times \text{const.} \tag{34}$$

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

Since its time complexity is polynomial in m and $1/\varepsilon$, Algorithm 4 is an FPTAS for Ellis's problem (Vazirani 2001).

Algorithms 3 and 4 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 a lookup table or memoized to prevent an exponential number of calls from forming on the stack.

4.5 Simulated annealing heuristic

Simulated annealing is a popular local-search heuristic for nonconvex optimization problems. Because of their inherent randomness, simulated-annealing algorithms offer no accuracy guarantee; however, they are easy to implement, computationally inexpensive, and often produce solutions with a small optimality gap. In this section, we present a simulated-annealing algorithm for Ellis's problem. It is patterned on the procedure for integer programs outlined by Wolsey (1998, § 12.3.2).

To generate an initial solution, our algorithm adopts the greedy heuristic of adding schools in descending order by $f_j t_j / g_j$ until the budget is exhausted. While Example 2 shows that the approximation coefficient of this solution can be arbitrarily small, in typical instances, it is much higher. Next, given a candidate solution $\mathcal{X}$, we generate a neighbor $\mathcal{X}_n$ by copying $\mathcal{X}$, adding schools to $\mathcal{X}_n$ until it becomes infeasible, then removing elements of $\mathcal{X}$ from $\mathcal{X}_n$ until feasibility is restored. If $v(\mathcal{X}_n) \geq v(\mathcal{X})$, then the candidate solution is updated to equal the neighbor; if not, then the candidate solution is updated with probability $\exp\big(v(\mathcal{X}_n) - v(\mathcal{X})/T\big)$, where T is a temperature parameter. This is repeated N times, and the algorithm returns the best $\mathcal{X}$ observed.

Proposition 2. *The computation time of Algorithm 5 is $O(Nm)$.*

In our numerical experiments, the output of Algorithm 5 was typically quite close to the optimum.

Algorithm 5: Simulated annealing for Ellis's problem.

Input: Utility values $t(0,\infty)^m$, admissions probabilities $f \in (0,1]^m$, application costs $g \in (0,\infty)^m$, budget $H \in (0,\infty)^m$.

Parameters: Initial temperature $T \geq 0$, temperature reduction parameter $r \in (0,1]$, number of iterations N.

Add schools to $\mathcal{X}$ in descending order by $f_j t_j / g_j$ until the budget is exhausted;
for $i = 1 \dots N$ **do**
  $\mathcal{X}_n \leftarrow \text{copy}(\mathcal{X})$;
  Add random schools from $\mathcal{C} \setminus \mathcal{X}$ to $\mathcal{X}_n$ until $\mathcal{X}_n$ infeasible;
  Remove random schools in $\mathcal{X}$ from $\mathcal{X}_n$ until feasibility restored;
  $\Delta = v(\mathcal{X}_n) - v(\mathcal{X})$;
  if $\Delta \geq 0$ **then** $\mathcal{X} \leftarrow \mathcal{X}_n$ **else** $\mathcal{X} \leftarrow \mathcal{X}_n$ with probability $\exp(\Delta/T)$;
  $T \leftarrow rT$;
end
return *the best* $\mathcal{X}$ *found*

5 Numerical experiments

In this section, we present the results of numerical experiments designed to test the practical efficacy of the algorithms derived above.

5.1 Implementation notes

We chose to implement our algorithms in the Julia language (v1.8.0b1) because its system of parametric data types allows the compiler to optimize for differential cases such as when the t_j-values are integers or floating-point numbers. Julia also offers convenient macros for parallel computing, which enabled us to solve more and larger problems in the benchmark (Bezanson et al. 2017). We made extensive use of the DataStructures.jl package (v0.18.11, github.com/JuliaCollections). The code is available at github.com/maxkapur/OptimalApplication.jl.

The dynamic programs, namely Algorithms 3 and 4, were implemented using recursive functions and dictionary memoization, as described in Cormen et al. (1990, § 16.6). Our implementation of Algorithm 4 also differed from that described in Subsection 4.4 in that we represented portfolio valuations in *binary* rather than decimal, with the definitions of P and $\mathcal{V}$ modified accordingly, and instead of fixed-point numbers, we worked in integers by multiplying each element of $\mathcal{V}$ by 2^P. These modifications yield a substantial performance improvement without changing the fundamental algorithm design or complexity analysis.

5.2 Experimental procedure

We conducted three numerical experiments. Experiments 1 and 2 evaluate the computation times of our algorithms for Alma's problem and Ellis's problem, respectively. Experiment 3 investigates the empirical accuracy of the simulated-annealing heuristic as compared to an exact algorithm.

To generate synthetic markets, we drew the t_j-values independently from an exponential distribution with a scale parameter of ten and rounded up to the nearest integer. To achieve partial negative correlation between t_j and f_j, we then set $f_j = 1/(t_j + 10\,Q)$, where Q is drawn uniformly from the interval $[0,1)$. In Experiment 1, which concerns Alma's problem, we set $h = \lfloor m/2 \rfloor$. In Experiments 2 and 3, which concern Ellis's problem, each g_j is drawn

uniformly from the set $\{5, \ldots, 10\}$ and we set $H = \lfloor \frac{1}{2} \sum g_j \rfloor$. This cost distribution resembles real-world college application fees, which are often multiples of \$10 between \$50 and \$100. A typical instance is shown in Figure 2.

The experimental variables in Experiments 1 and 2 were the market size m, the choice of algorithm, and (for Algorithm 4) the tolerance ε. For each combination of the experimental variables, we generated 50 markets, and the optimal portfolio was computed three times, with the fastest of the three repetitions recorded as the computation time. We then report the mean and standard deviation across the 50 markets. Therefore, each cell of each table below represents a statistic over 150 computations. Where applicable, we do not count the time required to sort the entries of t. As the simulated-annealing heuristic's computation time is negligible, it was excluded from Experiment 2.

In Experiment 3, we generated 500 markets with sizes varying uniformly in the logarithm between $m = 2^3$ and 2^{11}. For each market, we computed a heuristic solution using Algorithm 5 and an exact solution using Algorithm 3 and calculated the optimality ratio. We chose to use $N = 500$ iterations in the simulated-annealing algorithm, and the temperature parameters $T = 1/4$ and $r = 1/16$ were selected by a preliminary grid search.

5.3 Summary of results

Experiment 1 compared the performance of Algorithm 1 for homogeneous-cost markets of various sizes when the set of candidate schools $\mathcal{C}$ is stored as an array and as a binary max heap ordered by the $f_j \bar{t}_j$-values. The results appear in Table 2. Our results indicate that the array implementation is faster.

In Experiment 2, we turned to the general problem, and compared the performance of the exact algorithms (Algorithms 2 and 3) and the approximation scheme (Algorithm 4) at tolerances 0.5 and 0.05. The results, which appear in Table 3, broadly agree with the time complexity analyses presented above. The branch-and-bound algorithm proved impractical for even medium-sized instances. Overall, we found the exact dynamic program to be the fastest algorithm, while the FPTAS was rather slow, a result that echoes the results of computational studies on knapsack problems (Martello and Toth 1990, § 2.10). The strong performance of Algorithm 3 is partly attributable to the structure of our synthetic instances, in which application costs are small integers and H is proportional in expectation to m, meaning the expected computation time is $O(m^2)$. However, the relative advantage of Algorithm 3 may be even more pronounced in real college-application instances because the typical student's application budget accommodates at most a dozen or so schools and is constant in m.

The results of Experiment 3, which evaluated the performance of the simulated-annealing heuristic in our synthetic instances, are plotted in Figure 3. In the every instance, the heuristic found a solution within 10 percent of optimality, and in the vast majority of instances, it was within 2 percent of optimality. The heuristic performs most poorly in small markets of size $m \leq 16$, but as exact algorithms are tractable at this scale, this result presents no cause for concern.

6 Conclusion and ideas for future research

This study has introduced a novel combinatorial optimization problem that we call the college application problem. It can be viewed as a kind of asset allocation or knapsack problem with a submodular objective. We showed that the special case in which colleges have identical applica-

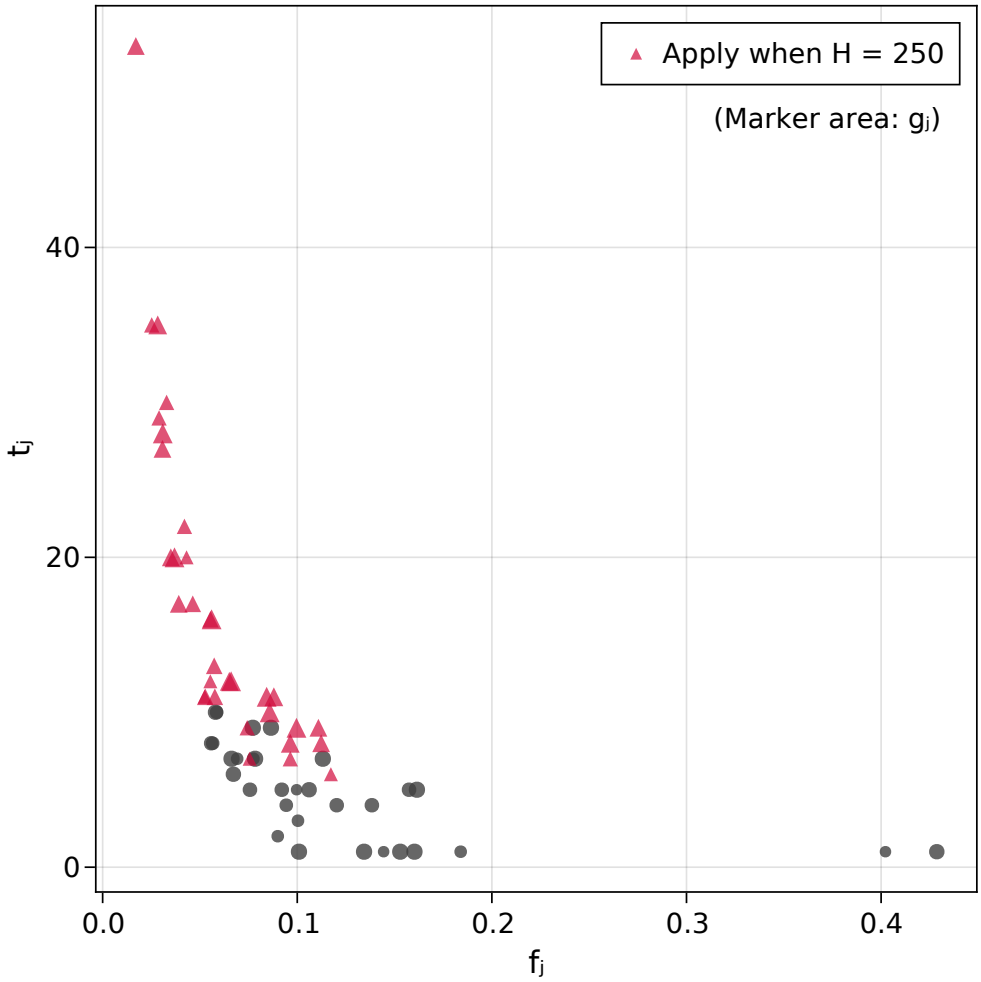

Figure 2: A typical randomly-generated instance with $m = 64$ schools and its optimal application portfolio. The application costs g_j were drawn uniformly from $\{5, \ldots, 10\}$. The optimal portfolio was computed using Algorithm 3.

Number of schools m	**Algorithm 1 with array**		**Algorithm 1 with heap**	
16	0.00	(0.00)	0.01	(0.00)
64	0.01	(0.00)	0.08	(0.02)
256	0.06	(0.00)	0.96	(0.26)
1024	0.82	(0.03)	14.27	(2.28)
4096	12.87	(0.71)	230.77	(18.75)
16384	199.48	(1.77)	3999.19	(283.48)

Table 2: (Experiment 1.) Time in ms to compute an optimal portfolio for an admissions market with homogeneous application costs using Algorithm 1 when $\mathcal{C}$ is stored as an array and as a heap. For each value of m, 50 markets were generated, and the computation time was recorded as fastest of three repetitions of the algorithm. The table shows the average time (standard deviation) over the 50 instances. In every case, $h = m/2$.

tion costs can be solved in polynomial time by a greedy algorithm because the optimal solutions are nested in the budget constraint. The general problem is NP-complete. We provided four solution algorithms. The strongest from a theoretical standpoint is an FPTAS that produces a $(1-\varepsilon)$-approximate solution in $O(m^3/\varepsilon)$ time. On the other hand, for typical college-application

Number of schools m	Algorithm 2: Branch & bound		Algorithm 3: Costs DP		Algorithm 4: FPTAS, $\varepsilon = 0.5$		Algorithm 4: FPTAS, $\varepsilon = 0.05$	
8	0.02	(0.01)	0.01	(0.00)	0.05	(0.01)	0.16	(0.04)
16	0.10	(0.04)	0.07	(0.02)	0.39	(0.10)	2.59	(0.62)
32	12.43	(9.88)	0.29	(0.05)	2.15	(0.29)	33.07	(10.72)
64	—	(—)	1.24	(0.16)	13.24	(2.22)	339.64	(100.38)
128	—	(—)	6.20	(0.64)	79.92	(20.18)	2042.63	(749.71)
256	—	(—)	30.63	(2.28)	818.50	(632.98)	18949.50	(3533.88)

Table 3: (Experiment 2.) Time in ms to compute an optimal or $(1-\varepsilon)$-optimal portfolio for an admissions market with heterogeneous application costs using the three algorithms developed in Section 4. The branch-and-bound algorithm is impractical for large markets. For each value of m, 50 markets were generated, and the computation time was recorded as fastest of three repetitions of the algorithm. The table shows the average time (standard deviation) over the 50 instances.

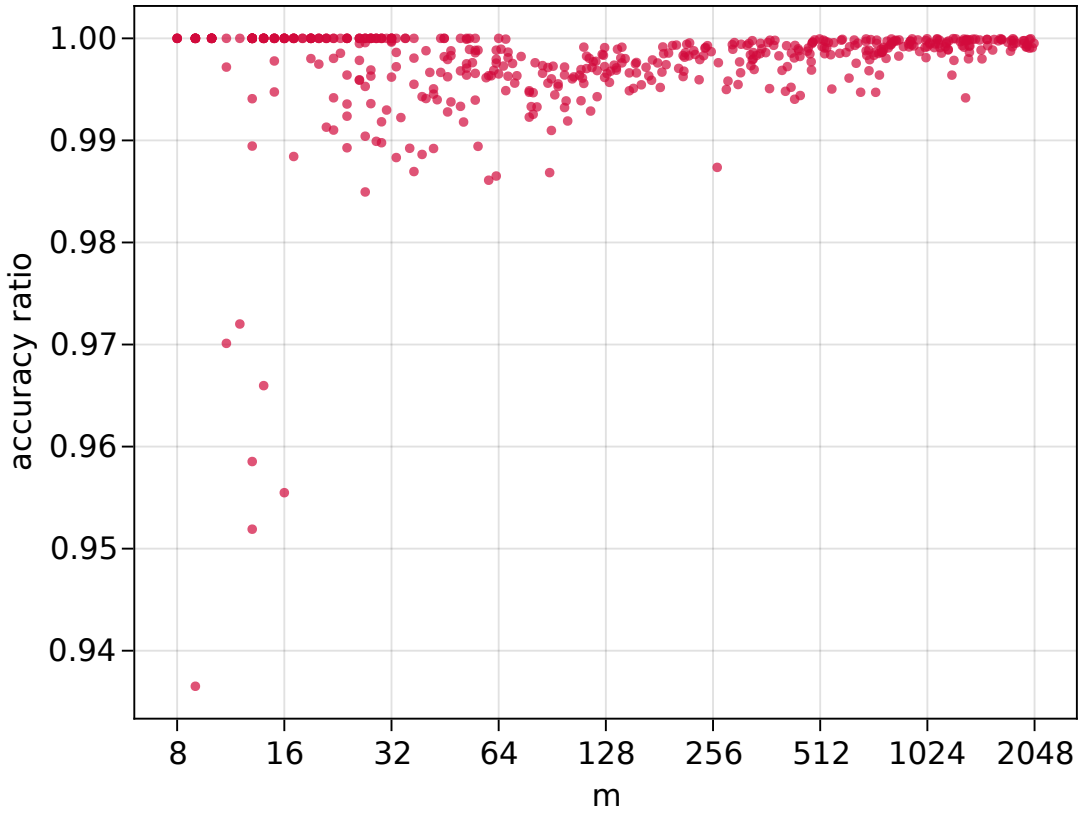

Figure 3: (Experiment 3.) Optimality ratio achieved by simulated annealing (Algorithm 5) in markets of varying size, with parameters $N = 500$, $T = 1/4$, and $r = 1/16$. Optimal portfolios were computed using Algorithm 3.

instances, the dynamic program based on application expenditures is both easier to implement and substantially more time efficient. A heuristic simulated-annealing algorithm also exhibited strong empirical performance in our computational study. The algorithms discussed in this paper are summarized in Table 4.

Three extensions of this problem appear amenable to future study.

Algorithm	Reference	Problem	Restrictions	Exactness	Computation time
Naïve	Definition 3	Homogeneous costs	None	$(1/h)$-opt.	$O(m)$
Greedy	Algorithm 1	Homogeneous costs	None	Exact	$O(hm)$
Branch and bound	Algorithm 2	General	None	Exact	$O(2^m)$
Costs DP	Algorithm 3	General	g_j integer	Exact	$O(Hm + m \log m)$
FPTAS	Algorithm 4	General	None	$(1-\varepsilon)$-opt.	$O(m^3/\varepsilon)$
Simulated annealing	Algorithm 5	General	None	0-opt.	$O(Nm)$

Table 4: Summary of algorithms discussed in this paper.

6.1 Explicit treatment of risk aversion

The familiar Markowitz portfolio optimization model includes an explicit risk-aversion term, whereas our model has settled for an implicit treatment of risk, namely the tradeoff between selective schools and safety schools inherent in the maximax objective function. However, it is possible to augment the objective function $v(\mathcal{X}) = \mathrm{E}[X]$ to incorporate a variance penalty $\beta \geq 0$ as follows:

$$\begin{aligned} v_\beta(\mathcal{X}) &= \mathrm{E}[X] - \beta \operatorname{Var}(X) \\ &= \mathrm{E}[X] - \beta \left(\mathrm{E}[X^2] - \mathrm{E}[X]^2 \right) \\ &= \sum_{j \in \{0\} \cup \mathcal{X}} \left(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right) - \beta \sum_{j \in \{0\} \cup \mathcal{X}} \left(f_j t_j^2 \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right) + \beta v(\mathcal{X})^2 \\ &= v(\mathcal{X}; \tau) + \beta v(\mathcal{X}; t)^2 \end{aligned} \tag{35}$$

where $\tau_j = t_j - \beta t_j^2$. Since the first term is itself a portfolio valuation function, and the second is a monotonic transformation of one, we speculate that one of the algorithms given above could be used as a subroutine to trace out the efficient frontier by maximizing $v(\mathcal{X}; t)$ subject to the budget constraint and $v(\mathcal{X}; \tau) \geq \alpha$ for various values of $\alpha \geq 0$.

A parametric treatment of risk aversion may align our model more closely to real-world applicant behavior. Conventional wisdom asserts that the best college application strategy combines reach, target, and safety schools in roughly equal proportion (Jeon 2015). When the application budget is small relative to the size of the admissions market, our algorithms recommend a similar approach (see the example of Subsection 3.5). However, inspecting equation (7), which discounts school utility parameters to reflect their marginal value relative to the schools already in the portfolio, reveals that low-utility schools are penalized more harshly than high utility schools in the marginal analysis. Consequentially, as the application budget grows, the optimal portfolio in our model tends to favor reach schools over safety schools. (See Figure 2 for a clear illustration of this phenomenon.)

A potential application of our model is an econometric study that estimates students' perceptions of college quality (t_j) from their observed application behavior ($\mathcal{X}$) under the assumption of utility maximization. Empirical research indicates that heterogeneous risk aversion is a significant source of variation in human decision-making, especially in the domains of educational choice and employment search (Kahneman 2011; Hartlaub and Schneider 2012; van Huizen

and Alessie 2019). Therefore, an endogenous risk-aversion parameter would greatly enhance our model's explanatory power in the econometric setting.

6.2 Signaling strategies

Another direction of further research with immediate application in college admissions is to incorporate additional signaling strategies into the problem. In the Korean admissions process, the online application form contains three multiple-choice fields, labeled *ga, na,* and *da* for the first three letters of the Korean alphabet, which students use to indicate the colleges to which they wish to apply. Most schools appear only in one or two of the three fields. Therefore, students are restricted not only in the number of applications they can submit, but in the combinations of schools that may coincide in a feasible application portfolio. From a portfolio optimization perspective, this is a *diversification constraint,* because its primary purpose is to prevent students from applying only to top-tier schools. However, in addition to overall selectivity, the three fields also indicate different evaluation schemes: The *ga* and *na* customarily indicate the applicant's first and second choice, and applications filed under these fields are evaluated with greater emphasis on standardized-test scores than those filed under the *da* field. Some colleges appear in multiple fields and run two parallel admissions processes, each with different evaluation criteria. Therefore, if a student recognizes that she has a comparative advantage in (for example) her interview skills, she may increase her chances of admission to a competitive school by applying under the *da* field. The optimal application strategy in this context can be quite subtle, and it is a subject of perennial debate on Korean social networks.

An analogous feature of the American college admissions process is known as early decision, in which at the moment of application, a student commits to enrolling in a college if admitted. In principle, students who apply with early decision may obtain a better chance of admission by signaling their eagerness to attend, but doing so weakens the college's incentive to entice the student with discretionary financial aid, altering the utility calculus.

In the integer formulation of the college application problem (Problem 5), a natural way to model these signaling strategies without introducing too much complexity is to split each c_j into c_{j+} and c_{j-}. The binary decision variables are $x_{j+} = 1$ if the student applies to c_j with high priority (that is, in the *ga* or *na* field or with early decision) and $x_{j-} = 1$ if applying with low priority (that is, in the *da* field or without early decision). Now, assuming that the differential admission probabilities f_{j+} and f_{j-} and utilities t_{j+} and t_{j-} are known, adding the logical constraints

$$x_{j+} + x_{j-} \leq 1, \quad j = 1 \dots m \qquad \text{and} \qquad \sum_{j=1}^{m} x_{j+} \leq 1 \tag{36}$$

completes the formulation. These are knapsack constraints; therefore, the submodular maximization algorithm of Kulik et al. (2013) can be used to obtain a $(1 - 1/e - \varepsilon)$-approximate solution for this problem. When the budget constraint is a cardinality constraint (as in Alma's problem), we note that adding these constraints yields a matroidal solution set, and Calinescu et al. (2011)'s algorithm yields an alternative solution, which has the same approximation coefficient of $1 - 1/e - \varepsilon$.

6.3 Memory-efficient dynamic programs

Our numerical experiments suggest that the performance of the dynamic programming algorithms is bottlenecked by not computation time, but memory usage. Reducing these algorithms' storage requirements would enable us to solve considerably larger problems.

Abstractly speaking, Algorithms 3 and 4 are two-dimensional dynamic programs that represent the optimal solution by $Z[N, C]$. Here N is the number of decision variables, C is a constraint parameter, and $Z[n, c]$ is the optimal objective value achievable using the first $n \leq N$ decision variables when the constraint parameter is $c \leq C$. The algorithm iterates on a recursion relation that expresses $Z[n, c]$ as a function of $Z[n-1, c]$ and $Z[n-1, c']$ for some $c' \leq c$. (In Algorithm 3, Z is the maximal portfolio valuation, $N = m$, and $C = H$; in Algorithm 4, Z is the minimal application expenditures, $N = m$, and $C = |\mathcal{V}| \propto m^2/\varepsilon$.)

When such a dynamic program is implemented using a lookup table or dictionary memoization, producing the optimal solution requires $O(NC)$ time and space. Kellerer et al. (2004, § 3.3) provide a technique for transforming the dynamic program Z into a divide-and-conquer algorithm that produces the optimal solution in $O(NC)$ time and $O(N + C)$ space, a significant improvement. However, their technique requires the objective function to be additively separable in a certain sense that appears difficult to conform to the college application problem.

A Appendix

A.1 Elementary proof of Theorem 5

Proof. We will prove the equivalent expression $2v(\mathcal{X}_h) \geq v(\mathcal{X}_{h+1}) + v(\mathcal{X}_{h-1})$. Applying Theorem 3, we write $\mathcal{X}_h = \mathcal{X}_{h-1} \cup \{j\}$ and $\mathcal{X}_{h+1} = \mathcal{X}_{h-1} \cup \{j, k\}$. If $t_k \leq t_j$, then

$$\begin{aligned}
2v(\mathcal{X}_h) &= v(\mathcal{X}_{h-1} \cup \{j\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&\geq v(\mathcal{X}_{h-1} \cup \{k\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&= v(\mathcal{X}_{h-1} \cup \{k\}) + (1 - f_j)v(\mathcal{X}_{h-1}) + f_j \,\mathrm{E}[\max\{t_j, X_{h-1}\}] \\
&= v(\mathcal{X}_{h-1} \cup \{k\}) - f_j v(\mathcal{X}_{h-1}) + f_j \,\mathrm{E}[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\
&\geq v(\mathcal{X}_{h-1} \cup \{k\}) - f_j v(\mathcal{X}_{h-1} \cup \{k\}) + f_j \,\mathrm{E}[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\
&= (1 - f_j)v(\mathcal{X}_{h-1} \cup \{k\}) + f_j \,\mathrm{E}[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\
&= v(\mathcal{X}_{h-1} \cup \{j, k\}) + v(\mathcal{X}_{h-1}) \\
&= v(\mathcal{X}_{h+1}) + v(\mathcal{X}_{h-1}).
\end{aligned} \tag{37}$$

The first inequality follows from the optimality of $\mathcal{X}_h$, while the second follows from the fact that adding k to $\mathcal{X}_{h-1}$ can only increase its valuation.

If $t_k \geq t_j$, then the steps are analogous:

$$\begin{aligned}
2v(\mathcal{X}_h) &= v(\mathcal{X}_{h-1} \cup \{j\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&\geq v(\mathcal{X}_{h-1} \cup \{k\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&= (1 - f_k)v(\mathcal{X}_{h-1}) + f_k \,\mathrm{E}[\max\{t_k, X_{h-1}\}] + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&= v(\mathcal{X}_{h-1}) - f_k v(\mathcal{X}_{h-1}) + f_k \,\mathrm{E}[\max\{t_k, X_{h-1}\}] + v(\mathcal{X}_{h-1} \cup \{j\}) \qquad (38) \\
&\geq v(\mathcal{X}_{h-1}) - f_k v(\mathcal{X}_{h-1} \cup \{j\}) + f_k \,\mathrm{E}[\max\{t_k, X_{h-1}\}] + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&= v(\mathcal{X}_{h-1}) + (1 - f_k)v(\mathcal{X}_{h-1} \cup \{j\}) + f_k \,\mathrm{E}[\max\{t_k, X_{h-1}\}] \\
&= v(\mathcal{X}_{h-1}) + v(\mathcal{X}_{h-1} \cup \{j, k\}) \\
&= v(\mathcal{X}_{h-1}) + v(\mathcal{X}_{h+1}) \qquad \square
\end{aligned}$$

References

Acharya, Mohan S., Asfia Armaan, and Aneeta S. Antony. 2019. "A Comparison of Regression Models for Prediction of Graduate Admissions." In *Second International Conference on Computational Intelligence in Data Science.* https://doi.org/10.1109/ICCIDS.2019.8862140.

Assad, Arjang. 1985. "Nested Optimal Policies for Set Functions with Applications to Scheduling." *Mathematics of Operations Research* 10 (1): 82–99.

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

Balas, Egon and Eitan Zemel. 1980. "An Algorithm for Large Zero-One Knapsack Problems." *Operations Research* 28 (5): 1130–54. https://doi.org/10.1287/opre.28.5.1130.

Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2017. "Julia: A Fresh Approach to Numerical Computing." *SIAM Review* 59: 65–98. https://doi.org/10.1137/141000671.

Blum, Manuel, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. 1973. "Time Bounds for Selection." *Journal of Computer and System Sciences* 7 (4): 448–61. https://doi.org/10.1016/S0022-0000(73)80033-9.

Calinescu, Gruia, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2011. "Maximizing a Monotone Submodular Function Subject to a Matroid Constraint." *SIAM Journal on Computing* 40 (6): 1740–66. https://doi.org/10.1137/080733991.

Carraway, Robert, Robert Schmidt, and Lawrence Weatherford. 1993. "An Algorithm for Maximizing Target Achievement in the Stochastic Knapsack Problem with Normal Returns." *Naval Research Logistics* 40 (2): 161–73. https://doi.org/10.1002/nav.3220400203.

Cormen, Thomas, Charles Leiserson, and Ronald Rivest. 1990. *Introduction to Algorithms.* Cambridge, MA: The MIT Press.

Dantzig, George B. 1957. "Discrete-Variable Extremum Problems." *Operations Research* 5 (2): 266–88.

Dean, Brian, Michel Goemans, and Jan Vondrák. 2008. "Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity." *Mathematics of Operations Research* 33 (4): 945–64. https://doi.org/10.1287/moor.1080.0330.

Kellerer, Hans, Ulrich Pferschy, and David Pisinger. 2004. *Knapsack Problems.* Berlin: Springer.

Kulik, Ariel, Hadas Shachnai, and Tami Tamir. 2013. "Approximations for Monotone and Nonmonotone Submodular Maximization with Knapsack Constraints." *Mathematics of Operations Research* 38 (4): 729–39. https://doi.org/10.1287/moor.2013.0592.

Jeon, Minhee. 2015. "[College application strategy] Six chances total. . . divide applications across reach, target, and safety schools" (in Korean). Jungang Ilbo, Aug. 26. https://www.joongang.co.kr/article/18524069.

Fisher, Marshall, George Nemhauser, and Laurence Wolsey. 1978. "An Analysis of Approximations for Maximizing Submodular Set Functions—I." *Mathematical Programming* 14: 265–94.

Fu, Chao. 2014. "Equilibrium Tuition, Applications, Admissions, and Enrollment in the College Market." *Journal of Political Economy* 122 (2): 225–81. https://doi.org/10.1086/675503.

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

Hartlaub, Vanessa and Thorsten Schneider. 2012. "Educational Choice and Risk Aversion: How

Important Is Structural vs. Individual Risk Aversion?" *SOEPpapers on Multidisciplinary Panel Data Research,* no. 433. https://www.diw.de/documents/publikationen/73/diw_01.c.394455.de/diw_sp0433.pdf.

Kahneman, Daniel. 2011. *Thinking, Fast and Slow.* New York: Macmillan.

Markowitz, Harry. 1952. "Portfolio Selection." *The Journal of Finance* 7 (1): 77–91. https://www.jstor.org/stable/2975974.

Martello, Silvano and Paolo Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations.* New York: John Wiley & Sons.

Meucci, Attilio. 2005. *Risk and Asset Allocation.* Berlin: Springer-Verlag, 2005.

Nemhauser, George and Laurence Wolsey. 1978. "Best Algorithms for Approximating the Maximum of a Submodular Set Function." *Mathematics of Operations Research* 3 (3): 177–88. https://doi.org/10.1287/moor.3.3.177.

Parker, R. Gary and Ronald L. Rardin. 1988. *Discrete Optimization.* San Diego: Academic Press.

Rozanov, Mark and Arie Tamir. 2020. "The Nestedness Property of the Convex Ordered Median Location Problem on a Tree." *Discrete Optimization* 36: 100581. https://doi.org/10.1016/j.disopt.2020.100581.

Sklarow, Mark. 2018. *State of the Profession 2018: The 10 Trends Reshaping Independent Educational Consulting.* Technical report, Independent Educational Consultants Association. https://www.iecaonline.com/wp-content/uploads/2020/02/IECA-Current-Trends-2018.pdf.

Sniedovich, Moshe. 1980. "Preference Order Stochastic Knapsack Problems: Methodological Issues." *The Journal of the Operational Research Society* 31 (11): 1025–32. https://www.jstor.org/stable/2581283.

Steinberg, E. and M. S. Parks. 1979. "A Preference Order Dynamic Program for a Knapsack Problem with Stochastic Rewards." *The Journal of the Operational Research Society* 30 (2): 141–47. https://www.jstor.org/stable/3009295.

Van Huizen, Thomas and Rob Alessie. 2019. "Risk Aversion and Job Mobility.'' *Journal of Economic Behavior & Organization* 164: 91–106. https://doi.org/10.1016/j.jebo.2019.01.021.

Vazirani, Vijay. 2001. *Approximation Algorithms.* Berlin: Springer.

Wolsey, Laurence. 1998. *Integer Programming.* New York: John Wiley & Sons.