How to apply to college

Max Kapur

March 7, 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 three algorithms for this more general setup. The first is a branch-and-bound routine. The second is an dynamic program that produces an exact solution in pseudopolynomial time. The third is a fully polynomial-time approximation scheme.

요약

본 논문은 다수의 확률 변수로 구성된 포트폴리오의 기대 최적값을 예산 조건 하에서 최대화하는 문제를 고려한다. 이를 최적 대학 지원 문제라고 부른다. 각 확률 변수의 비용, 즉 각 대학의 지원비가 동일한 경우, 최적 포트폴리오는 예산 제약식으로 결정된 포함 사슬 관계를 가짐을 보이고 이에 따라 다항 시간 해법을 제시한다. 대학의 지원비가 서로 다른 경우, 문제가 NP-complete함을 증명한다. 일반적인 상황을 위해 세 가지 해법을 도출한다. 첫째는 분지한계 기반 해법이다. 둘째는 의사 다항 시간 안에 정확한 해를 출력하는 동적 계획이다. 셋째는 전체 다항 시간 근사 해법이다.

Contents

1 서론

본 논문은 다음과 같은 최적화 문제를 고려한다.

$$\begin{aligned} \text{maximize} \quad & v(\mathcal{X}) = \mathrm{E}\Big[\max\big\{t_0, \max\{t_j Z_j : j \in \mathcal{X}\}\big\}\Big] \\ \text{subject to} \quad & \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j\in\mathcal{X}} g_j \leq H \end{aligned} \tag{1.1}$$

단, $\mathcal{C} = \{1 \ldots m\}$은 지표 집합이며 H는 예산 모수이다. 각 $j = 1 \ldots m$에 대해 $g_j > 0$는 비용 모수이며 Z_j는 확률 f_j를 가지는 서로 독립적인 Bernoulli 변수이다. 각 $j = 0 \ldots m$에 대해 t_j는 효용 모수이다.

다음 해석에 따라 이를 '최적 대학 지원 포트폴리오 문제'라고 부른다. m개의 대학교를 가지는 입학 시장을 고려하자. j번째 학교의 이름은 c_j이다. 이 시장에 참가하는 한 학생을 고려해서 그가 c_j에 진학하면 t_j 단위의 효용이 발생한다고 하자. 단, 어떤 대학에도 진학하지 않는 경우 그의 효용은 t_0이다. c_j의 지원비가 g_j이며 학생이 지원비에 쓸 수 있는 예산이 H라고 하자. 마지막, c_j에 지원하면 학생이 합격할 확률이 f_j이라고 하자. 따라서 합격하면 $Z_j = 1$, 합격 안 하면 $Z_j = 0$이 된다. f_j가 (학교의 전체적인 합격률이 아니라) 바로 이 학생의 합격 확률일 때 Z_j의 확률적 독립성을 적절한 가정이다. 그러면 학생의 목표는 주어진 예산 안에서 학생이 합격하는 가장 좋은 학교의 기대 효용을 최대화하는 것이다. 위 문제의 최적해가 $\mathcal{X}$일 때, 학생의 최적 대학 지원 전략은 $\mathcal{X}$에 지원하는 것이다.

Chao (2014)가 주장한 바처럼 대학 지원 전략은 약간 미묘한 포트폴리오 최적화 문제이다. 재정학 분야에서 고전적 포트폴리오 최적화 모형은 전체 자산에 대한 기대 총이익에서 위험회피 항을 뺌으로 선형식으로 제약된 오목 최대화 문제를 이룬다 (Markowitz 1952; Meucci 2005). 그러나 대학 지원자는 가치가 제일 높을 단 한 자산의 가치를 최대화하고자 한다. 어떤 학생이 자신의 j위 학교에 합격하면, $(j+1)$위 학교에 합격하든 불합격하든 무관심하므로 다른 상황이다. 이에 따라 학생이 최대화하려는 함수는 각 지원의 기대 효용에 대해 오목이 아닌 볼록 함수다. 또한 전형적인 입학 시장에서 대학의 효용과 합격 확률이 서로 반비례하므로 대학 지원 문제에 위험 관리가 내포되어 있다. 특히 선호도가 높으며 붙기 어려운 "상향학교"(reach school)와 선호도가 낮으며 붙기 쉬운 "안정학교"(safety school) 사이의 균형을 고려해야 한다 (김민희 2015). 마지막, 대학 지원의 조합적인 본성으로 인해 연속적인 포트폴리오 최적화 문제에서 흔히 사용하는 기울기 방법으로는 풀기가 어렵다. Chao는 지원을 제약 조건 대신 비용으로 모형화했으며 모형을 추정하기 위해 $m = 8$이 되도록 학교를 먼저 클러스터로 모았다. 이는 열거법을 통해 문제를 쉽게 풀 수 있는 규모이지만 본 연구는 더 일반적인 해법을 목표로 추구한다.

대학 지원 문제의 정수 모형은 m차 다항식을 목적함수로 갖춘 일종의 이진 배낭 문제로 볼 수도 있다. 본 논문에서 제시하는 분지한계법과 동적 계획 해법은 배낭 문제를 위한 기존 알고리즘하고 비슷하다 (Martello와 Toth 1990, §2.5–6). 입학 확률을 적당히 조정하면 원래 목적함수를 특성벡터의 선형 함수에 대한 임의로 정확한 근사로 변환할 수 있으며 NP-completeness 증명에서 이 사실을 활용한다. 배낭 문제에 확률성을 도입한 선행 연구 중 각 상품의 효용이 정해진 확률 분포로 결정되는 모형(Steinberg와 Parks 1979; Carraway 외 1993), 그리고 배낭에 삽입한 다음에 상품의 무게가 관측되는 온라인 모형(Dean et al. 2008)과 같은 바가 있다. 본 연구가 고려하는 문제는 Steinberg와 Parks 그리고 Carraway 외가 살펴본 "우선순위 배낭 문제"와는 피상적으로 비슷하지만, 우선순위 배낭 문제는 대학 지원 문제의 특색인 "maximax" 형태를 가지지는 않는다. 또한 우선순위 모형과 달리, 본 연구에서

스칼라-값 목적함수를 ‘결과 분포’의 선호 순위로 대체하고자 노력하지 않는다. 확률적 지배와 관련된 방법론적인 문제가 생기기 때문이다 (Sniedovich 1980). 대신 t_j-값으로 유도된 학생의 ‘결과’에 대한 선호 순위를 그대로 받아들이고 위에서 정의한, 모호하지 않은 문제를 위한 효율적인 계산법에 집중한다.

본 연구에서 특히 탐욕 해법의 타당성에 관심을 기울인다. 예를 들어 예산이 다 소비될 때까지 목적함수를 가장 많이 증가시키는 학교를 차례대로 선택하는 알고리즘은 일종의 탐욕 해법이다. 탐욕 해법은 예산 H으로 모수화된 해의 순서를 유도하며 해는 ‘포함 사슬 관계’(nestedness)로 서로에 연결된다. 즉 $H \leq H'$일 때, 예산 H에 해당하는 탐욕 해는 예산 H'에 해당하는 탐욕 해의 부분집합이다. Rozanov와 Tamir (2020)가 주장하듯, 최적해가 포함 사슬 관계를 맺음을 알면 최적해를 구하는 데일뿐더러 정보가 불확실한 상황에서 최적해를 이행하는 데에도 유용하다. 가령, 많은 미국 대학의 지원 기한은 11월 초인데 학업 계획서를 학교의 취향에 맞춰서 작성해야 하므로 여름부터 원서를 작성하는 학생이 많다. 그러나 시간이나 지원비에 대한 예산을 10월 말까지 모를 수도 있다. 포함 사슬 관계 성질, 또는 탐욕 해법의 타당성은 완전한 예산 정보가 없음에도 학교가 최적 포트폴리오에 진입하는 순서대로 원서를 작성하는 식으로 최적 전략을 이행해 낼 수 있다고 의미한다.

한편, 탐욕 알고리즘이 좋은 근사 해법이며 다른 가정을 추가하면 정확한 알고리즘인 되는 최적화 모형 몇몇이 있으며 대표적인 바는 카디널리티 제약 위에 자모(子模) 집합 함수 (submodular set function)를 최대화하는 문제이다 (Fisher et al. 1978). 반면에 이진 배낭 문제 같은 경우에서는 가장 직관적인 탐욕 알고리즘이 무한히 비최적인 해를 출력하는 인스턴스를 만들 수 있다 (Vazirani 2001). 대학 지원 문제에 대해 배낭 문제와 매우 유사한 결과를 도출할 수 있다. 모든 $g_j = 1$인 특수한 경우에서, 최적 포트폴리오가 탐욕 알고리즘의 타당성과 동등한 포함 사슬 관계를 만족하는 것을 보인다. 이 경우는 지원비가 없으며 정시 모집 기간에 학교 3개에만 지원할 수 있는 한국 입학 과정과 같다. 불행하게도 일반적인 경우에서 포함 사슬 관계가 성립하지 않을뿐더러 탐욕 알고리즘에 대해 최소한 근사 계수를 보장할 수 없다. 대신 분지한계법, 전형적인 입학 시장 인스턴스에 대해 효율적인 의사 다항 시간 동적 계획, 그리고 전체 다항 시간 안에 $(1-\varepsilon)$-근사해를 출력하는 알고리즘과 같은 3가지 해법은 제안한다.

1.1 본 논문의 구성

2절은 표기법은 명시하고 일방성을 잃지 않는 추가의 가정을 소개한다.

3절에서 각 $g_j = 1$이고 H가 $h \leq m$인 자연수인 특수한 경우를 고려한다. 직관적인 휴리스틱 해법이 실제로 $1/h$-근사 해법임을 보인다. 그다음에 최적 포트폴리오가 예산 제약에 대한 포함 사슬 관계를 보이고 이에 따라 $O(hm)$-시간 정확한 해법을 도출한다.

4절은 각 학교의 지원비가 다를 수 있는 상황으로 계속한다. 이진 배낭 문제에서의 다항 변환을 통해서 포트폴리오 최적화 문제가 NP-complete임을 증명한다. 일반적인 상황을 위해 3개의 해법은 제안한다. 첫째는 분지한계법이다. 둘째는 총 지출액으로 탐색해서 $O(Hm + m \log m)$과 같은 의사 다항 시간 동적 계획이다. 셋째는 분수를 내린 포트폴리오 가치로 탐색하는 다른 동적 계획이다. 이를 통해서 $O(m^3/\varepsilon)$-시간 안에 $(1-\varepsilon)$-근사해를 출력하는 전체 다항 시간 근사 해법(fully polynomial-time approximation scheme)을 얻는다.

5절에서 위에서 도출한 타당성과 계산 복잡성을 확인하는 계산적 실험 결과를 정리한다.

간단한 결론이 뒤를 이은다.

2 표기법과 예비 결과

Before discussing the solution algorithms, we will introduce some additional notation and a few preliminary results that will come in handy.

For the remainder of the paper, unless otherwise noted, we assume with trivial loss of generality that each $g_j \leq H$, each $f_j \in (0, 1]$, and $t_0 < t_1 \leq \cdots \leq t_m$. Below, we will show how to transform an arbitrary instance so that $t_0 = 0$, in which case each $t_j > 0$.

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*.

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

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 italic and script names such as $\mathcal{Y}_h$ and Y_h 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 $j = 0 \ldots m$,

$$p_j(\mathcal{X}) = \begin{cases} f_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i), & j \in \{0\} \cup \mathcal{X} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

where the empty product equals one. The following proposition follows immediately.

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

$$v(\mathcal{X}) = \sum_{j=0}^{m} t_j p_j(\mathcal{X}) = \sum_{j \in \{0\} \cup \mathcal{X}} \Big(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \Big) \tag{2.2}$$

Next, we show that without loss of generality, we may assume that $t_0 = 0$ (or any constant less than t_1).

정리 1. *Let $\bar{t}_j = t_j - \gamma$ for $j = 0 \ldots m$. Then $v(\mathcal{X}; \bar{t}_j) = v(\mathcal{X}; t_j) - \gamma$ regardless of $\mathcal{X}$.*

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

$$v(\mathcal{X}; \bar{t}_j) = \sum_{j \in \{0\} \cup \mathcal{X}} \bar{t}_j p_j(\mathcal{X}) = \sum_{j \in \{0\} \cup \mathcal{X}} (t_j - \gamma) p_j(\mathcal{X}) \tag{2.3}$$

$$= \sum_{j \in \{0\} \cup \mathcal{X}} t_j p_j(\mathcal{X}) - \gamma = v(\mathcal{X}; t_j) - \gamma \tag{2.4}$$

which completes the proof. □

3 동일한 지원 비용

In this section, we derive a polynomial-time algorithm for the special case in which $g_j = 1$ and H is a constant $h \leq m$. This case is similar to the centralized college admissions process in Korea, where there is no application fee, but by law, students are allowed to apply to no more than h schools. (In the Korean case, $m = 202$ and $h = 3$.) Applying Theorem 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.

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

$$\begin{aligned} \text{maximize} \quad & v(\mathcal{X}) = \sum_{j\in\mathcal{X}} \Big(f_j t_j \prod_{\substack{i\in\mathcal{X}:\\ i>j}} (1-f_i) \Big) \\ \text{subject to} \quad & \mathcal{X} \subseteq \mathcal{C}, \quad |\mathcal{X}| \leq h \end{aligned} \tag{3.1}$$

3.1 Naïve 해법의 근사 성질

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 inoptimal.

정의 2 (Naïve algorithm for Alma's problem)**.** Apply to the h schools having the highest expected utility $f_j t_j$.

The basic error of this algorithm is that it maximizes $\mathrm{E}\left[\sum t_j Z_j\right]$ instead of $\mathrm{E}\left[\max\{t_j Z_j\}\right]$. The latter is what Alma is truly concerned with, since in the end she can attend only one school. The following example shows that the naïve algorithm can produce a suboptimal solution.

예 1. Suppose $m = 3$, $h = 2$, $t = (70, 80, 90)$, and $f = (0.4, 0.4, 0.3)$. Then the naïve algorithm picks $\mathcal{T} = \{1, 2\}$ with $v(\mathcal{T}) = 70(0.4)(1-0.4) + 80(0.4) = 48.8$. But $\mathcal{X} = \{2, 3\}$ with $v(\mathcal{X}) = 80(0.4)(1-0.3) + 90(0.3) = 49.4$ is the optimal solution.

In fact, the naïve algorithm is a $(1/h)$-approximation algorithm for Alma's problem, as expressed in the following theorem.

정리 2. *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 $v(\mathcal{T}_h)/v(\mathcal{X}_h) \geq 1/h$.*

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

$$\begin{aligned} v(\mathcal{X}_h) = \mathrm{E}\big[\max_{j\in\mathcal{X}_h}\{t_j Z_j\}\big] &\leq \mathrm{E}\Big[\sum_{j\in\mathcal{X}_h}\{t_j Z_j\}\Big] \leq \mathrm{E}\Big[\sum_{j\in\mathcal{T}_h}\{t_j Z_j\}\Big] \\ &= h\,\mathrm{E}\Big[\tfrac{1}{h}\sum_{j\in\mathcal{T}_h}\{t_j Z_j\}\Big] \leq h\,\mathrm{E}\big[\max_{j\in\mathcal{T}_h}\{t_j Z_j\}\big] = h\,v(\mathcal{T}_h) \end{aligned} \tag{3.2}$$

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

The following example establishes the tightness of the approximation factor.

예 2. Pick any h and let $m = 2h$. For a small constant $\varepsilon \in (0, 1)$, let

$$t = \Big(\underbrace{1, \dots, 1}_{h}, \underbrace{\varepsilon^{-1}, \varepsilon^{-2}, \dots, \varepsilon^{-(h-1)}, \varepsilon^{-h}}_{h}\Big)$$
$$\text{and} \quad f = \Big(\underbrace{1, \dots, 1}_{h}, \underbrace{\varepsilon^{1}, \varepsilon^{2}, \dots, \varepsilon^{h-1}, \varepsilon^{h}}_{h}\Big)$$

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

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

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 3.2.)

따름정리 1. *The function $v(\mathcal{X})$ is not submodular.*

증명. If $v(\mathcal{X})$ is submodular, then theorem 4.2 of Fisher et al. (1978) implies that the naïve algorithm achieves an optimality ratio of $v(\mathcal{T}_h)/v(\mathcal{X}_h) \geq 1 - \big(\frac{h-1}{h}\big)^h$. Example 2 provides a counterexample. $\square$

Hope is not lost. We can still find the optimal solution in time polynomial in h and m, as we will now show.

3.2 포함 사슬 관계

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.

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

$$\mathcal{X}_1 \subset \mathcal{X}_2 \subset \cdots \subset \mathcal{X}_m. \tag{3.3}$$

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

증명. By induction on h. Applying Theorem 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 $t_k \leq t_l$. Optimality requires that

$$v(\mathcal{X}_1) = f_j t_j > v(\{k\}) = f_k t_k \tag{3.4}$$

and

$$\begin{aligned} v(\mathcal{X}_2) = f_k(1-f_l)t_k + f_l t_l > v(\{j,l\}) & & (3.5)\\ &= f_j(1-f_l)t_j + (1-f_j)f_l t_l + f_j f_l \max\{t_j,t_l\} & (3.6)\\ &\geq f_j(1-f_l)t_j + (1-f_j)f_l t_l + f_j f_l t_l & (3.7)\\ &= f_j(1-f_l)t_j + f_l t_l & (3.8)\\ &\geq f_k(1-f_l)t_k + f_l t_l = v(\mathcal{X}_2) & (3.9)\end{aligned}$$

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, assume that $v(\mathcal{Y}_h) < v(\mathcal{X}_h)$. Then

$$\begin{aligned} v(\mathcal{X}_{h+1}) &= v(\mathcal{Y}_h \cup \{k\})\\ &= (1-f_k)v(\mathcal{Y}_h) + f_k t_k\\ &< (1-f_k)v(\mathcal{X}_h) + f_k \operatorname{E}\big[\max\{t_k, X_h\}\big]\\ &= v(\mathcal{X}_h \cup \{k\}) \end{aligned} \tag{3.10}$$

contradicts the optimality of $\mathcal{X}_{h+1}$.

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 $\{k\}$.

We will use the function $w(\mathcal{Y})$ to grade portfolios $\mathcal{Y} \subseteq \mathcal{C} \setminus \{k\}$ according to how well they complement $\{k\}$. To construct $w(\mathcal{Y})$, let $\tilde{t}_j$ denote the expected utility Alma 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 $\tilde{t}_j = (1-f_k)t_j + f_k t_k$; for $j > k$, this is $\tilde{t}_j = t_j$. This means that

$$v(\mathcal{Y} \cup \{k\}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \tilde{t}_j p_j(\mathcal{Y}). \tag{3.11}$$

The transformation to $\tilde{t}$ does not change the order of the t_j-values. Therefore, the expression on the right side of (3.11) is itself a portfolio valuation function. In the corresponding market, t is replaced by $\tilde{t}$ and $\mathcal{C}$ is replaced by $\mathcal{C} \setminus \{k\}$. Now, we obtain $w(\mathcal{Y})$ through one more transformation: Define $\bar{t}_j = \tilde{t}_j - \tilde{t}_0$ so that $t_0 = 0$ and let

$$w(\mathcal{Y}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \bar{t}_j p_j(\mathcal{Y}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \tilde{t}_j p_j(\mathcal{Y}) - \tilde{t}_0 = v(\mathcal{Y} \cup \{k\}) - f_k t_k \tag{3.12}$$

where the second equality follows from Theorem 1. This identity says that the optimal complements to $\{k\}$, given by $\mathcal{Y}_{h-1}$ and $\mathcal{Y}_h$, are themselves optimal portfolios of size $h-1$ and h for the market whose objective function is $w(\mathcal{Y})$. Since $\bar{t}_0 = 0$ in the latter market, the inductive hypothesis implies that $\mathcal{Y}_{h-1} \subset \mathcal{Y}_h$, which completes the proof.[1] □

[1]We thank Yim Seho for discovering this critical transformation.

3.3 다항 시간 해법

Applying the result above yields an efficient 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 (2.2); 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 the inductive step in the proof of Theorem 3. Once school k is added to $\mathcal{X}$, we remove it from the set $\mathcal{C} \setminus \mathcal{X}$ of candidates, and update the t_j-values of the remaining schools according to the following transformation:

$$\bar{t}_j = \begin{cases} (1 - f_k)t_j, & t_j \leq t_k \\ t_j - f_k t_k, & t_j > t_k \end{cases} \tag{3.13}$$

It is easy to verify that this is the composition of the two transformations (from t to $\tilde{t}$, and from $\tilde{t}$ to $\bar{t}$) given in the proof. 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 (3.13), 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.

The algorithm below outputs a list `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 $\mathcal{X}_h = \{\mathtt{X}[1], \dots, \mathtt{X}[\mathtt{h}]\}$. The entries of the list `V` give the valuation thereof.

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

```
Data: Utility values $t \in [0, \infty)^m$, admissions probabilities $f \in [0, 1]^m$, application
      limit $h \leq m$.
1  $\mathcal{C} \leftarrow \{1 \dots m\}$;
2  X, V ← empty lists;
3  for $i = 1 \dots h$ do
4      $k \leftarrow \arg\max_{j \in \mathcal{C}} \{f_j t_j\}$;
5      $\mathcal{C} \leftarrow \mathcal{C} \setminus \{k\}$;
6      append!(X, $k$);
7      if $i = 1$ then append!(V, $f_k t_k$) else append!(V, V[i − 1] + $f_k t_k$);
8      for $j \in \mathcal{C}$ do
9          if $t_j \leq t_k$ then $t_j \leftarrow t_j(1 - f_k)$ else $t_j \leftarrow t_j - f_k t_k$;
10     end
11 end
12 return X, V
```

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

증명. Optimality follows from the proof of Theorem 3. Suppose $\mathcal{C}$ is stored as a list. Then 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 $O(hm)$. □

In our numerical experiments, we found it effective to store $\mathcal{C}$ as a binary max heap rather than a list. The heap is ordered according to the criterion $i \geq j \iff f_i t_i \geq f_j t_j$. Nominally, using a heap increases the cost of the main loop from $O(hm)$ to $O(hm \log m)$ because the heap is rebalanced when each t_j-value is updated. However, typical problem instances do not achieve this upper bound because the order of the $f_j t_j$-values changes only slightly between iterations. The cost of updating each t_j-value can be reduced to unit time using a Fibonacci heap (Fredman and Tarjan 1987), yielding the same overall computation time.

3.4 최적 포트폴리오의 성질

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

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

$$v(\mathcal{X}_h) - v(\mathcal{X}_{h-1}) \geq v(\mathcal{X}_{h+1}) - v(\mathcal{X}_h). \tag{3.14}$$

증명. 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 $\mathcal{X}_{h+1} = \mathcal{X}_{h-1} \cup \{j, k\}$. Define the random variables X_i as above. If $t_k \leq t_j$, then

$$\begin{aligned} 2v(\mathcal{X}_h) &= v(\mathcal{X}_{h-1} \cup \{j\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\ &\geq v(\mathcal{X}_{h-1} \cup \{k\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\ &= v(\mathcal{X}_{h-1} \cup \{k\}) + (1 - f_j)v(\mathcal{X}_{h-1}) + f_j \operatorname{E}[\max\{t_j, X_{h-1}\}] \\ &= v(\mathcal{X}_{h-1} \cup \{k\}) - f_j v(\mathcal{X}_{h-1}) + f_j \operatorname{E}[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\ &\geq v(\mathcal{X}_{h-1} \cup \{k\}) - f_j v(\mathcal{X}_{h-1} \cup \{k\}) + f_j \operatorname{E}[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\ &= (1 - f_j)v(\mathcal{X}_{h-1} \cup \{k\}) + f_j \operatorname{E}[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\ &= v(\mathcal{X}_{h-1} \cup \{j, k\}) + v(\mathcal{X}_{h-1}) \\ &= v(\mathcal{X}_{h+1}) + v(\mathcal{X}_{h-1}). \end{aligned} \tag{3.15}$$

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:

$$\begin{aligned}
2v(\mathcal{X}_h) &= v(\mathcal{X}_{h-1} \cup \{j\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&\geq v(\mathcal{X}_{h-1} \cup \{k\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&= (1 - f_k)v(\mathcal{X}_{h-1}) + f_k \operatorname{E}[\max\{t_k, X_{h-1}\}] + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&= v(\mathcal{X}_{h-1}) - f_k v(\mathcal{X}_{h-1}) + f_k \operatorname{E}[\max\{t_k, X_{h-1}\}] + v(\mathcal{X}_{h-1} \cup \{j\}) \qquad (3.16) \\
&\geq v(\mathcal{X}_{h-1}) - f_k v(\mathcal{X}_{h-1} \cup \{j\}) + f_k \operatorname{E}[\max\{t_k, X_{h-1}\}] + v(\mathcal{X}_{h-1} \cup \{j\}) \\
&= v(\mathcal{X}_{h-1}) + (1 - f_k)v(\mathcal{X}_{h-1} \cup \{j\}) + f_k \operatorname{E}[\max\{t_k, X_{h-1}\}] \\
&= v(\mathcal{X}_{h-1}) + v(\mathcal{X}_{h-1} \cup \{j, k\}) \\
&= v(\mathcal{X}_{h-1}) + v(\mathcal{X}_{h+1}) \qquad \square
\end{aligned}$$

It follows that when $\mathcal{X}_h$ is the optimal h-portfolio, for a given market, $v(\mathcal{X}_h)$ is $O(h)$. Example 2, in which $v(\mathcal{X}_h)$ can be made arbitrarily close to h, establishes the tightness of this bound.

3.5 작은 예

Let us consider a fictional market consisting of $m = 8$ schools. The market 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.

Iteration 1:
$$\begin{aligned}
C &= \{1, 2, 3, 4, 5, 6, 7, 8\} \\
f &= \{0.39, 0.33, 0.24, 0.24, 0.05, 0.03, 0.1, 0.12\} \\
t &= \{200, 250, 300, 350, 400, 450, 500, 550\} \\
f * t &= \{78.0, 82.5, 72.0, \underline{84.0}, 20.0, 13.5, 50.0, 66.0\} \qquad \Longrightarrow \mathcal{X}_3 = \{4\}
\end{aligned}$$

Iteration 2:
$$\begin{aligned}
C &= \{1, 2, 3, 5, 6, 7, 8\} \\
f &= \{0.39, 0.33, 0.24, 0.05, 0.03, 0.1, 0.12\} \\
t &= \{152, 190, 228, 316, 366, 416, 466\} \\
f * t &= \{59.28, \underline{62.7}, 54.72, 15.8, 10.98, 41.6, 55.92\} \qquad \Longrightarrow \mathcal{X}_3 = \{4, 2\}
\end{aligned}$$

Iteration 3:
$$\begin{aligned}
C &= \{1, 3, 5, 6, 7, 8\} \\
f &= \{0.39, 0.24, 0.05, 0.03, 0.1, 0.12\} \\
t &= \{101.84, 165.3, 253.3, 303.3, 353.3, 403.3\} \\
f * t &= \{39.718, 39.672, 12.665, 9.099, 35.33, \underline{48.396}\} \qquad \Longrightarrow \mathcal{X}_3 = \{4, 2, 8\}
\end{aligned}$$

$\dots$

Iteration 8:
$$C = \{6\}, f = \{0.03\}, t = \{177.622\}, f * t = \{\underline{5.329}\} \qquad \Longrightarrow \mathcal{X}_3 = \{4, 2, 8, 1, 7, 3, 5, 6\}$$

The output of the algorithm is $\mathtt{X} = [4, 2, 8, 1, 7, 3, 5, 6]$, and the optimal h-portfolio consists of

지표 j	학교 c_j	효용 t_j	합격 확률 f_j	지원 순위	$v(\mathcal{X}_h)$
1	수성대	200	0.39	4	230.0
2	금성대	250	0.33	2	146.7
3	화성대	300	0.24	6	281.5
4	목성대	350	0.24	1	84.0
5	토성대	400	0.05	7	288.8
6	천왕성대	450	0.03	8	294.1
7	해왕성대	500	0.10	5	257.7
8	명왕성대	550	0.12	3	195.1

Table 1: Market 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 number h or less.

its first h entries. The “priority” column of Table 1 shows the inverse permutation of $\mathtt{X}$, 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.

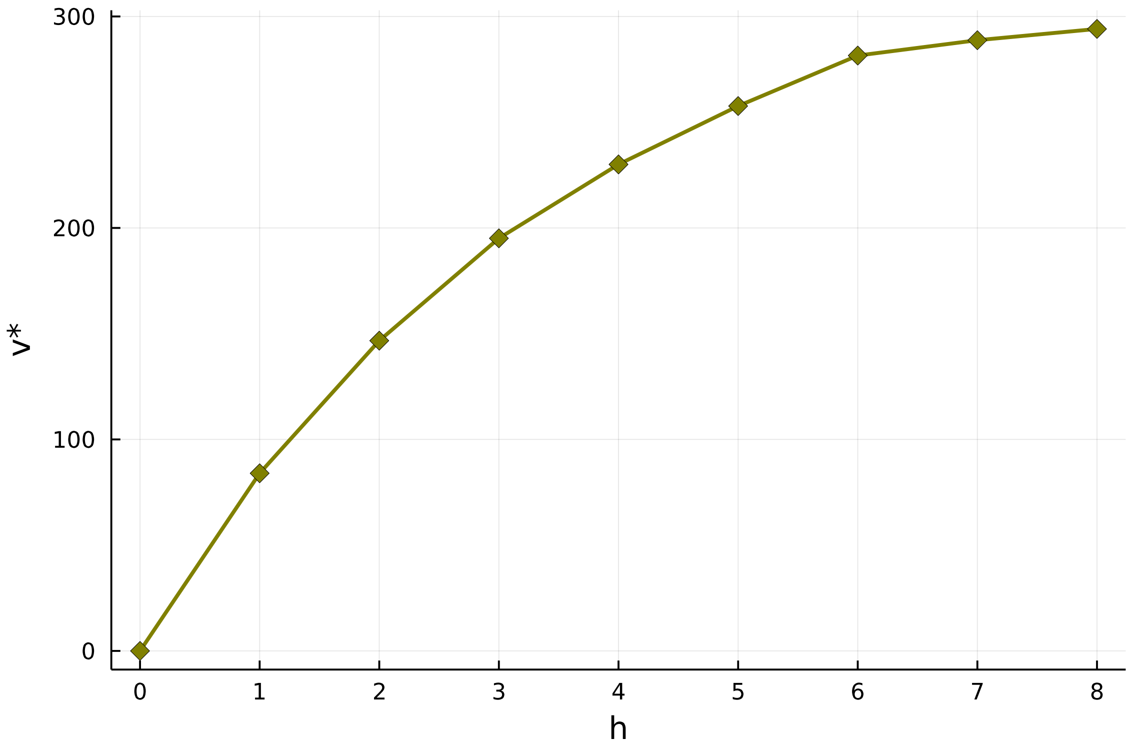

그림 1: $m = 8$개의 학교로 구성된 가상 입학 시상에서, 각 지원 제한 h에 해당하는 최적 포트폴리오의 가치 $v^* = v(\mathcal{X}_h)$.

4 다양한 지원 비용

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 Theorem 1, we assume $t_0 = 0$ and disregard the outside option throughout.

문제 2 (Ellis's problem)**.** Ellis's optimal college application portfolio is given by the solution to the following combinatorial optimization problem.

$$\begin{aligned} \text{maximize} \quad & v(\mathcal{X}) = \sum_{j \in \mathcal{X}} \Big(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \Big) \\ \text{subject to} \quad & \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j \in \mathcal{X}} g_j \leq H \end{aligned} \tag{4.1}$$

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

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).

문제 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 $\sum_{j \in \mathcal{B}'} w_j \leq W$.

문제 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 $\sum_{j \in \mathcal{X}} g_j \leq H$.

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

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

Consider an instance of the knapsack problem, and we will construct an instance of Ellis's problem 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 the objects in $\mathcal{B}$ are indexed in increasing order of u_j, that each $u_j > 0$, and that the knapsack instance admits a feasible solution other than the empty set.

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 $t_j = u_j/\delta$ for all j. 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 $\mathcal{X}$,

$$\begin{aligned}\sum_{j\in\mathcal{X}} u_j = \sum_{j\in\mathcal{X}} f_j t_j &> \sum_{j\in\mathcal{X}}\Big(f_j t_j \prod_{\substack{j'\in\mathcal{X}:\\ j'>j}} (1-f_{j'})\Big) = v(\mathcal{X}) \\ &= \sum_{j\in\mathcal{X}}\Big(u_j \prod_{\substack{j'\in\mathcal{X}:\\ j'>j}} (1-\delta)\Big) \geq (1-\delta)^m \sum_{j\in\mathcal{X}} u_j \\ &\geq (1-m\delta)\sum_{j\in\mathcal{X}} u_j \geq \sum_{j\in\mathcal{X}} u_j - m\delta U_{\max} = \sum_{j\in\mathcal{X}} u_j - 1.\end{aligned} \tag{4.2}$$

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. $\square$

An intuitive extension of the greedy algorithm for Alma's problem is to iteratively add to $\mathcal{X}$ the school k for which $[v(\mathcal{X}\cup\{k\}) - v(\mathcal{X})]/g_k$ is largest. However, 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 poor approximation ratio.

예 3. Let $t = (10, 2021)$, $f = (1, 1)$, $g = (1, 500)$, and $H = 500$. Then the greedy approximation algorithm produces the clearly inoptimal solution $\mathcal{X} = \{1\}$.

4.2 분지한계법

A traditional approach to knapsack 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.

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

$$\begin{aligned} \text{maximize} \quad & v(x) = \sum_{j=1}^{m}\Big(f_j t_j x_j \prod_{i>j}(1-f_i x_i)\Big) \\ \text{subject to} \quad & \sum_{j=1}^{m} g_j x_j \leq H, \;\; x_i \in \{0,1\}^m \end{aligned} \tag{4.3}$$

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.

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

$$\begin{aligned} \text{maximize} \quad & v_{\mathrm{LP}}(x) = \sum_{j=1}^{m} f_j t_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{m} g_j x_j \le H, \;\; x \in [0,1]^m \end{aligned} \tag{4.4}$$

Problem 6 is a continuous knapsack problem, which is easily solved in $O(m \log m)$ time by the following greedy algorithm: Start with $x = \mathbf{0}$. While $H > 0$, select the school k for which $f_j t_j / g_j$ is highest, set $x_k \leftarrow \min\{1, H/g_k\}$, and set $H \leftarrow H - g_k$ (Dantzig 1957).

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, $\mathcal{O}$ consists of those that are "out," 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

$$\begin{aligned} \text{maximize} \quad & v(x) = \gamma + \sum_{j \in \mathcal{N}} \Big(f_j \bar{t}_j x_j \prod_{\substack{i \in \mathcal{N}: \\ i > j}} (1 - f_i x_i) \Big) \\ \text{subject to} \quad & \sum_{j \in \mathcal{N}} g_j x_j \le \bar{H}; \;\; x_j \in \{0,1\}, \;\; j \in \mathcal{N}. \end{aligned} \tag{4.5}$$

The second is the corresponding instance of Problem 6:

$$\begin{aligned} \text{maximize} \quad & w_{\mathrm{LP}}(x) = \gamma + \sum_{j \in \mathcal{N}} f_j \bar{t}_j x_j \\ \text{subject to} \quad & \sum_{j \in \mathcal{N}} g_j x_j \le \bar{H}; \;\; x_j \in [0,1], \;\; j \in \mathcal{N} \end{aligned} \tag{4.6}$$

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 (3.13) 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 (3.13).

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 $g_j \le \bar{H}$ 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 in $\mathcal{N}$ 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 (3.13).

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.*

예 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 generate its children. To compute the subproblems, we first simple disregard school 1. Next, to eliminate school 2, we apply (3.13) 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 school 5 by applying (3.13) 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, $\bar{H} = H - g_2 - g_5 = 3$. Problems 4.5 and 4.6 are now easily obtained by substitution.

Since all the schools in $\mathcal{N}$ have $g_j \leq \bar{H}$, n has two children. Applying the node-generation rule, school 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 $v^*_{\mathrm{LP}}[n]$, the optimal objective value of the LP relaxation (4.6). 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 (4.5) is identical to the function $v(\mathcal{I} \cup \mathcal{Y})$, where $\mathcal{Y} = \{j : y_j = 1\}$. This means that the optimal objective value v^*_{LP} 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 $v^*_{\mathrm{LP}}[n_1]$ for some other node n_1, then n_1 and all its descendants can be eliminated from consideration.

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^*_{LP}-value, generates its children, and removes n from the candidate set. Next, the children n' of n are inspected; if one of them yields a new optimal solution, then we mark it as the best candidate and fathom any nodes n'' for which $v_{\mathcal{I}}[n'] > v^*_{\mathrm{LP}}[n'']$. When no nodes remain in the candidate set, the algorithm has explored every branch, so it terminates and returns the best candidate.

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

```
Data: 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 $n_0 \leftarrow (\emptyset, \emptyset, \mathcal{C})$;
Current lower bound $L \leftarrow 0$ and best solution $\mathcal{X} \leftarrow \emptyset$;
Initialize candidate set $\mathfrak{T} \leftarrow \{n_0\}$;
while not finished do
    if $\mathfrak{T} = \emptyset$ then return $\mathcal{X}, L$;
    else
        $n \leftarrow \arg\max\{v^*_{\mathrm{LP}}[n] : n \in \mathfrak{T}\}$;
        Remove $n$ from $\mathfrak{T}$;
        for each child $n'$ of $n$ do
            if $L < v_{\mathcal{I}}[n']$ then
                $L \leftarrow v_{\mathcal{I}}[n']$;
                Update $\mathcal{X}$ to the $\mathcal{I}$-set associated with $n'$;
            end
            if $n'$ is not a leaf then add $n'$ to $\mathfrak{T}$;
        end
        for $n''$ in $\mathfrak{T}$ do
            if $L > v^*_{\mathrm{LP}}[n'']$ then remove $n''$ from $\mathfrak{T}$;
        end
    end
end
```

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

증명. The discussion above implies that the as if the algorithm terminates, then an optimal solution exists among the leaves of the tree, and it is therefore returned.

Since the set of possible nodes is finite, 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 $m = 30$ schools.

4.3 의사 다항 시간 동적 계획

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 and $O(Hm)$ space. 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).

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 $h = 1 \dots H$:

$$V[j, h] = \max\bigl\{V[j-1, h], (1 - f_j)V[j-1, h-g_j] + f_j t_j\bigr\} \tag{4.7}$$

with the convention that $-\infty \cdot 0 = -\infty$. The corresponding optimal portfolios can be 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 $\mathcal{X}$.

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

```
Data: Utility values $t \in [0, \infty)^m$, admissions probabilities $f \in (0, 1]^m$, application
      costs $g \in \mathbb{N}^m$, budget $H \in \mathbb{N}$.
1  Index schools in ascending order by $t$;
2  Fill a lookup table with the entries of $V[j, h]$;
3  $h \leftarrow H$;
4  $\mathcal{X} \leftarrow \emptyset$;
5  for $j = m, m-1, \dots, 1$ do
6      if $V[j-1, h] < V[j, h]$ then
7          $\mathcal{X} \leftarrow \mathcal{X} \cup \{j\}$;
8          $h \leftarrow h - g_j$;
9      end
10 end
11 return $\mathcal{X}$
```

정리 8 (Validity of Algorithm 3)**.** *Algorithm 3 produces an optimal application portfolio for Ellis's problem in $O(Hm + m \log m)$ time and $O(Hm)$ space.*

증명. 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 $O(Hm)$. $\square$

4.4 전체 다항 시간 근사 해법

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 a fully polynomial-time approximation scheme for Ellis's problem that uses $O(m^3/\varepsilon)$ time and space. Here we assume, with a small loss of generality, that each t_j is a natural number.

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 $r[x] = \lfloor 10^P x \rfloor 10^{-P}$ 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 possible valuations possible in the fixed-point framework is finite:

$$\mathcal{V} = \Big\{0, 1 \times 10^{-P}, 2 \times 10^{-P}, \ldots, r\big[\bar{U} - 1 \times 10^{-P}\big], r\big[\bar{U}\big]\Big\} \tag{4.8}$$

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 $0 \leq j \leq m$ 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 $G[j, v]$, where $G[j, v] = \infty$ if $\mathcal{W}[j, v]$ does not exist. It is clear that if $v \leq 0$, 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

$$G[j, v] = \begin{cases} \infty, & t_j < v \\ \min\big\{G[j-1, v], g_j + G[j-1, v - \Delta_j(v)]\big\}, & t_j \geq v \end{cases} \tag{4.9}$$

$$\text{where} \qquad \Delta_j(v) = \begin{cases} r\left[\frac{f_j}{1-f_j}(t_j - v)\right], & f_j < 1 \\ \infty, & f_j = 1 \end{cases} \tag{4.10}$$

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

$$v(\mathcal{W}[j, v]) = (1 - f_j) v(\mathcal{W}[j, v] \setminus \{j\}) + f_j t_j. \tag{4.11}$$

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

$$G[j, v] = \min\bigl\{G[j-1, v], g_j\bigr\}. \tag{4.12}$$

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.

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

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

증명. (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 (4.9) and (4.10), if $j \in \mathcal{W}[j, v]$, then the true value of $(1 - f_j)v\bigl(\mathcal{W}[j-1, v - \Delta_j(v)]\bigr) + f_j t_j$ may exceed the fixed-point valuation v of $\mathcal{W}[j, v]$, but not by more than 10^{-P}. This error accumulates 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 (4.11). 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

$$v(\mathcal{W}) \geq v'(\mathcal{W}) \geq v'(\mathcal{X}) \geq v(\mathcal{X}) - m10^{-P} \geq \left(1 - \frac{m^2 10^{-P}}{\bar{U}}\right) v(\mathcal{X}) \geq (1-\varepsilon)\, v(\mathcal{X}) \tag{4.13}$$

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

$$|\mathcal{V}| = \bar{U} \times 10^P + 1 = \bar{U} \times 10^{\left\lceil \log_{10}\left(m^2/\varepsilon\bar{U}\right)\right\rceil} + 1 \leq \frac{m^2}{\varepsilon} \times \text{const.} \tag{4.14}$$

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

Since these bounds are polynomial in m and $1/\varepsilon$, Algorithm 4 is a fully polynomial-time approximation scheme for Ellis's problem (Vazirani 2001).

Algorithms 3 and 4 can be written using recursive functions instead; however, since each function references itself *twice,* the function values at each index must be recorded in a lookup table or otherwise memoized to prevent an exponential number of calls from forming on the stack.

5 계산적 실험

In this section, we present the results of numerical experiments designed to confirm the time complexity results established above. In both experiments, markets were generated by drawing t_j independently from an exponential distribution with a scale parameter of ten and rounding up to the nearest integer. To achieve partial negative correlation between t_j and f_j, we then set $f_j = 1/(t_j + 10Q)$, where Q is drawn uniformly from the interval $[0, 1)$. A typical instance is shown in Figure 2.

In Experiment 1, which concerns Algorithm 1, we take each $g_j = 1$ and set $H = h = \lfloor m/2 \rfloor$. In Experiment 2, which concerns Algorithms 2, 3, and 4, each g_j is drawn uniformly from the set $\{5, \dots, 10\}$ and we take $H = \lfloor \frac{1}{2} \sum g_j \rfloor$.

At each combination of the experimental variables, we generated ten markets, and each computation was repeated three times, with the fastest of the three recorded as the computation time. Therefore, each cell of each table represents 30 computations. We report the mean and standard deviation across the 50 markets. Where applicable, we do not count the time required to sort the entries of t.

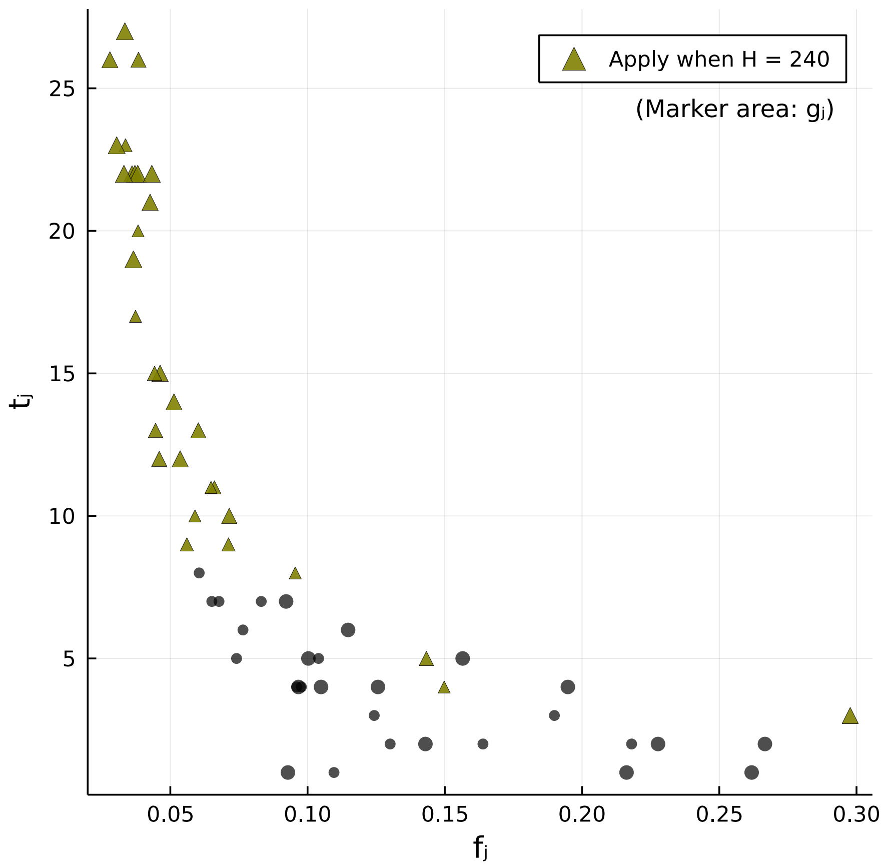

그림 2: $m = 60$개의 학교로 구성된 전형적인 무작위로 생성한 인스턴스와 해당 최적 포트폴리오. 지원비 g_j는 $\{5, \dots, 10\}$에서 균일한 확률로 선정했으며 최적 포트폴리오는 알고리즘 3(으)로 계산했다.

The dynamic programs, namely Algorithms 3 and 4, were implemented using recursive functions and memoization. Our implementation of Algorithm 4 differed slightly from that given above: 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 t_j-value by 2^P. These modifications yield a substantial performance improvement without changing the fundamental algorithm design or complexity analysis.

6 결론

7 참고문헌

김민희. 2015. "[대입 수시 전략] 총 6번의 기회 · · · '상향 · 소신 · 안정' 분산 지원하라." 중앙일보, 8월 26일. https://www.joongang.co.kr/article/18524069.

Budish, Eric. 2011. "The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes." *Journal of Political Economy* 119 (6): 1061–1103. https://doi.org/10.1086/664613.

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.

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.

Fisher, Marshall, George Nemhauser, and Laurence Wolsey. 1978. "An analysis of approximations for maximizing submodular set functions—I." *Mathematical Programming* 14: 265–94.

Fredman, Michael Lawrence and Robert Tarjan. 1987. "Fibonacci heaps and their uses in improved network optimization algorithms." *Journal of the Association for Computing Machinery* 34 (3): 596–615.

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.

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.

Othman, Abraham, Eric Budish, and Tuomas Sandholm. 2010. "Finding Approximate Competitive Equilibria: Efficient and Fair Course Allocation." In *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems.* New York: ACM. https://dl.acm.org/doi/abs/10.5555/1838206.1838323.

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.

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.

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