

How to apply to college

Max Kapur

March 11, 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 최대값을 예산 조건 하에서 최대화하는 문제를 고려한다. 이를 대학 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 다행 시간 해법을 제시한다. 대학의 지원 비용이 서로 다른 경우, 문제가 NP-complete함을 증명한다. 세 가지 일반적인 해법을 도출한다. 첫째는 분지한계 기반 해법이다. 둘째는 의사 다행 시간 안에 정확한 해를 출력하는 동적 계획 해법이다. 마지막으로 다른 동적 계획 기반으로 완전 다행 시간 근사 해법(fully polynomial-time approximation scheme)이 존재함을 보인다.

Contents

1	서론	3
1.1	본 논문의 구성	4
2	표기법과 예비 결과	5
3	동일한 지원 비용	5
3.1	나이브 해법의 근사 성질	6
3.2	포함 사슬 관계	7
3.3	다항 시간 해법	8
3.4	최적 포트폴리오의 성질	9
3.5	작은 예	10
4	동일하지 않은 지원 비용	12
4.1	NP-completeness	12
4.2	분지한계법	13
4.3	의사 다항 시간 동적 계획	16
4.4	완전 다항 시간 근사 해법	17
5	계산적 실험	20
5.1	실험 방법	20
5.2	결과 정리	20
5.3	알고리즘의 이행	20
6	결론	22
7	참고문헌	23

1 서론

본 논문은 다음과 같은 최적화 문제를 고려한다.

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

단, $\mathcal{C} = \{1 \dots m\}$ 은 지표 집합이며 H 는 예산을 나타내는 모수이다. 각 $j = 1 \dots m$ 에 대해 $g_j > 0$ 는 비용 모수이며 Z_j 는 확률 f_j 를 가지는 서로 독립적인 Bernoulli 변수이다. 각 $j = 0 \dots m$ 에 대해 t_j 는 효용 모수이다.

다음 해석에 따라 이를 ‘대학 지원 포트폴리오 최적화 문제’라고 부른다. m 개의 대학교를 가지는 입학 시장을 고려하자. j 번째 학교의 이름은 c_j 이다. 어떤 학생이 c_j 에 입학하게 되면 효용 t_j 를 얻게 된다고 하자. 단, 어떤 대학에도 진학하지 않는 경우 그 학생의 효용은 t_0 이다. c_j 의 지원 비용이 g_j 이며 학생이 지원에 쓸 수 있는 예산이 H 라고 하자. (예산 제약 조건은 또한 원서를 작성하는 시간이나 지원 개수에 대한 법적인 제한을 나타낼 수도 있다.) 마지막으로, c_j 에 지원하면 학생이 합격할 확률이 f_j 이라고 하자. 따라서 합격하면 $Z_j = 1$, 합격 안 하면 $Z_j = 0$ 이 된다. f_j 가 (학교의 전체적인 합격률이 아니라) 바로 이 학생의 합격 확률일 때 Z_j 의 확률적 독립성은 적절한 가정이다. 그러면 학생의 목표는 주어진 예산 안에서 학생이 합격하는 학교들에서 얻는 효용 중 기대 최대값을 최대화하는 것이다. 위 문제의 최적해가 \mathcal{X} 일 때, 학생의 최적 대학 지원 전략은 \mathcal{X} 에 속한 학교로 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 뺨으로 선형식으로 제약된 오목 최대화 문제를 이룬다 (Markowitz 1952; Meucci 2005). 그러나 대학 지원자는 가치가 제일 높은 단일 자산의 기대 가치를 최대화하고자 한다. 어떤 학생이 자신이 j 번째로 선호하는 학교에 합격하면 ($j+1$)번째 학교의 합격 여부는 무관한 상황이 된다. 이는 학생이 최대화하려는 함수를 각 지원 발송의 기대 효용에 대해 오목이 아닌 볼록 함수로 만든다. 또한 입학 시장에서는 전형적으로 대학의 효용과 합격 확률이 서로 반비례하므로 대학 지원 문제는 위험 관리를 포함하게 된다. 특히 선호도가 높으면 불기 어려운 “상향” 지원 학교(reach school)와 선호도가 낮으며 불기 쉬운 “안정” 지원 학교(safety school) 사이의 균형을 고려해야 한다 (김민희 2015). 마지막, 대학 지원의 조합적인 본성으로 인해 연속적인 포트폴리오 최적화 문제에서 흔히 사용하는 기울기 해법으로는 풀기가 어렵다. Chao는 지원 비용을 제약 조건 대신 목적함수의 한 항으로 모형화했으며, 모형의 모수를 추정하기 위해 $m = 8$ 이 되도록 학교들의 클러스터를 먼저 구성했다. 이는 열거법을 통해 문 쉽게 풀 수 있는 규모이지만 본 연구는 더 일반적인 m 에 대한 해법을 추구한다.

대학 지원 문제의 정수 모형은 m 차 다항식을 목적함수로 갖춘 일종의 이진 배낭 문제로 볼 수도 있다. 본 논문에서 제시하는 분지한계법과 동적 계획 해법은 배낭 문제를 위한 기존 알고리즘과 매우 비슷하다 (Martello와 Toth 1990, § 2.5–6). 입학 확률을 적당히 조정하면 특성벡터의 선형 함수를 원래 목적함수로 원하는 만큼의 정확성을 가지고 근사할 수 있으며 NP-completeness 증명에서 이 성질을 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 무게를 관측할 수 있는 온라인 모형 (Dean et al. 2008) 등이 있다. 본 연구가 고려하는 문제는 Steinberg와 Parks 그리고 Carraway 외가 고려한 “우선순위 배낭 문제”와는 유사성을 가지지만, 우선순위 배낭 문제는 대학 지원 문제의 특색인 ‘maximax’ 형태를 가지지는 않는다. 또한 우선순위 모형과 달리, 본 연구에서 실수값 목적함수를 선호 순위를 정의해야 하는 ‘결과 분포’로 대체할 필요가 없다. 확률적 우위에 대한 상반된 개념들의 문제를 야기시키기 때문이다 (Sniedovich 1980). 대신 t_j -값으로 유도된 학생의 ‘결과’에 대한 선호 순위를 받아들여 위에서 정의한 것처럼 명확히

정의된 문제를 위한 효율적인 계산법을 지향한다.

본 연구에서 특히 탐욕 해법의 가능성에 관심을 기울인다. 예를 들어 예산이 다 소비될 때까지 목적함수를 가장 많이 증가시키는 학교를 차례대로 추가하는 알고리즘은 일종의 탐욕 해법이다. 탐욕 해법은 예산 H 으로 모수화된 해의 순서를 유도하며 그의 원소들은 ‘포함 사슬 관계’(nestedness)로 연결한다. 즉 $H \leq H'$ 일 때, 예산 H 에 해당하는 탐욕 해는 예산 H' 에 해당하는 탐욕 해의 부분집합이 된다. Rozanov와 Tamir (2020)가 주장하듯, 최적해가 포함 사슬 관계를 가지면 최적해를 구하는 것뿐 아니라 정보가 불확실한 상황에서 최적해를 구현하는 데에도 유용하다. 가령, 많은 미국 대학의 지원 기한은 11월 초인데 학업 계획서를 학교의 취향에 맞춰서 작성해야 하므로 여름부터 원서를 작성하는 학생이 많다. 그러나 지원 예산은 10월 말까지 모를 수도 있다. 포함 사슬 관계, 또는 탐욕 해법의 타당성은 완전한 예산 정보가 없어도 학교가 최적 포트폴리오에 진입하는 순서대로 원서를 작성하면 최적 전략을 구현해 낼 수 있음을 의미한다.

탐욕 알고리즘이 좋은 근사해거나 정확한 알고리즘이 되는 최적화 모형들이 알려져 있다. 집합 크기 제약 하에서 하위 모듈 집합 함수(submodular set function)를 최대화하는 문제가 대표적인 예다 (Fisher et al. 1978). 반면에 이진 배낭 문제 같은 경우에는 가장 직관적인 탐욕 알고리즘이 최적과 거리가 먼 해를 출력하는 예를 만들 수 있다 (Vazirani 2001). 대학 지원 문제와 배낭 문제가 서로 매우 유사함을 보인다. 모든 $g_j = 1$ 인 특수한 경우에서, 최적 포트폴리오가 탐욕 알고리즘의 타당성과 동등한 포함 사슬 관계를 만족하는 것을 증명한다. 이 경우는 지원 비용이 없으며 정시 모집 기간에 학교 3개에만 지원할 수 있는 한국 입학 과정과 같다. 그러나 일반적인 경우에서 포함 사슬 관계가 성립하지 않을뿐더러 탐욕 알고리즘이 어떤 근사 계수를 보장할 수 없을 보일 수 있다. 대신 분지한계법, 전형적인 입학 시장 인스턴스에 대해 효율적인 의사 다행 시간 동적 계획, 그리고 완전 다행 시간 안에 $(1 - \varepsilon)$ -근사해를 출력하는 알고리즘 등 3가지 해법을 제안한다.

1.1 본 논문의 구성

2절은 표기법과 일반성을 제한하지 않는 편의적 가정을 제시한다.

3절에서 각 $g_j = 1$ 이고 H 가 $h \leq m$ 이고 자연수인 특수한 경우를 고려한다. 직관적인 휴리스틱 해법이 실제로 $1/h$ -근사 해법임을 증명한다. 그다음에 최적 포트폴리오의 예산 제약에 대한 포함 사슬 관계를 보이고 이를 이용해서 $O(hm)$ -시간 정확한 해법을 도출한다.

4절에서 각 학교의 지원 비용이 동일하지 않은 일반적인 상황을 다룬다. 이진 배낭 문제에서의 다행 변환을 통해서 포트폴리오 최적화 문제가 NP-complete임을 증명한다. 또한 3개의 일반적인 해법을 제안한다. 첫째는 분지한계법이다. 둘째는 총 지출액으로 탐색하는 $O(Hm + m \log m)$ 과 같은 의사 다행 시간 동적 계획 해법이다. 셋째는 실수값을 분수값으로 내림한 포트폴리오 가치를 탐색하는 또다른 동적 계획 해법이다. 이를 통해서 $O(m^3/\varepsilon)$ -시간 안에 $(1 - \varepsilon)$ -근사해를 출력하는 완전 다행 시간 근사 해법(fully polynomial-time approximation scheme)을 얻는다.

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

그리고 간단한 결론을 제시한다.

2 표기법과 예비 결과

해법은 의논하기 전, 본 절에서 추가적인 표기법은 명시하고 몇 가지 유용한 예비 결과를 정리하고자 한다.

남은 논문에서 다른 언급이 없으면, 일반성을 잃지 않고 각 $g_j \leq H$, 각 $f_j \in (0, 1]$, 그리고 $t_0 < t_1 \leq \dots \leq t_m$ 이라고 가정한다. 뒤에서 임의의 인스턴스를 $t_0 = 0$ 이 되도록 변환하는 방법을 제시하는데 이를 적용하면 $t_j > 0$ 인 가정을 세울 수 있다.

학생이 지원하는 학교 집합 $\mathcal{X} \subseteq \mathcal{C}$ 를 ‘지원 포트폴리오’라고 부르자. \mathcal{X} 에서 학생이 받는 기대 효용을 포트폴리오의 ‘가치’라고 한다.

정의 1 (포트폴리오 가치 함수). $v(\mathcal{X}) = E[\max\{t_0, \max\{t_j Z_j : j \in \mathcal{X}\}\}]$.

편의상 포트폴리오에 속한 학교가 실제로 이루는 효용을 확률 변수 $X = \max\{t_j Z_j : j \in \mathcal{X}\}$ 로 정의한다. 그러면 $t_0 = 0$ 이면 $v(\mathcal{X}) = E[X]$ 임을 알 수 있다. 비슷한 식으로 \mathcal{Y}_h 와 \mathcal{Y}_H 같은 스크립트체와 이탈릭체로 표기한 변수 쌍은 유사한 의미를 가진다.

주어진 지원 포트폴리오에 대해 학생이 c_j 로 진학할 확률을 $p_j(\mathcal{X})$ 라고 하자. 이 사건이 발생하는 필수충분 조건은 c_j 에 지원하고, c_j 에서 합격받고, c_j 보다 선호하는 (즉, 지표가 더 높은) 학교에서 불합격받는 것이다. 따라서 $j = 0 \dots m$ 에 대해

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

단, 공집합의 곱은 1이다. 다음 제의는 직설적인 결과다.

제의 1 (포트폴리오 가치 함수의 다른 식).

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

그다음에, 일반성을 제한하지 않고 $t_0 = 0$ (또는 t_1 보다 작은 임의의 상수)이라고 가정할 수 있음을 보이자.

정리 1. 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} .

증명. 정의에 따라 $\sum_{j=0}^m p_j(\mathcal{X}) = \sum_{j \in \{0\} \cup \mathcal{X}} p_j(\mathcal{X}) = 1$ 이다. 그러면 다음 수식으로 증명을 완성할 수 있다.

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

3 동일한 지원 비용

본 절에서 모든 $g_j = 1$ 이며 H 가 자연수 $h \leq m$ 인 특수한 경우에 집중한다. 직관적인 휴리스틱 해법이 실제로 $1/h$ -근사 해법임을 보인 다음에 정확한 다행 시간 해법을 도출한다. 이 특수한 경우는 지원 비용이 없으며 국가 법에 따라 최대 h 개의 학교에 지원할 수 있는 한국의 중심화된 대학 지원 과정과 비슷하다. (한국 경우, $m = 202$ 이고 $h = 3$ 이다.) 정리 1을(를) 적용해서 다른 언급이 없으면

$t_0 = 0$ 임을 가정하자. 본 절 내내 지원자를 ‘알마’라고 부르고 해당 최적화 문제를 ‘알마의 문제’라고 부른다.

문제 1 (알마의 문제). 알마의 최적 대학 지원 포트폴리오는 다음 조합 최적화 문제의 최적해이다.

$$\begin{aligned} \text{maximize } v(\mathcal{X}) &= \sum_{j \in \mathcal{X}} \left(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right) \\ \text{subject to } \mathcal{X} &\subseteq \mathcal{C}, \quad |\mathcal{X}| \leq h \end{aligned} \quad (4)$$

3.1 나이브 해법의 근사 성질

단일 학교 c_j 에 해당하는 기대 효용이 단순히 $E[t_j Z_j] = f_j t_j$ 이므로 다음과 같은 알고리즘이 매력적으로 보일 수 있지만 사실은 최적해가 아니다.

정의 2 (알마의 문제를 위한 나이브 해법). 기대 효용 $f_j t_j$ 가 가장 높은 h 개의 학교로 지원한다.

위 알고리즘의 기본 실수는 $E[\max\{t_j Z_j\}]$ 대신 $E[\sum t_j Z_j]$ 을 최대화하는 것이다. 결국에는 단 한 학교로만 진학할 수 있으므로 알마의 목적은 후자가 아닌 전자다. 다음 예는 나이브 해법이 최적해가 아닌 해를 출력할 수 있음을 보인다.

예 1. $m = 3$, $h = 2$, $t = (70, 80, 90)$, 그리고 $f = (0.4, 0.4, 0.3)$ 이라고 하자. 그러면 나이브 해법은 $\mathcal{T} = \{1, 2\}$ 를 선택하며 그의 기대 효용은 $v(\mathcal{T}) = 70(0.4)(1 - 0.4) + 80(0.4) = 48.8$ 이다. 그러나 $\mathcal{X} = \{2, 3\}$ 를 선택하면 기대 효용은 $v(\mathcal{X}) = 80(0.4)(1 - 0.3) + 90(0.3) = 49.4$ 이고 이는 최적해다.

다음 정리는 나이브 해법이 알마의 문제를 위한 $(1/h)$ -근사 해법임을 의미한다.

정리 2 (나이브 해법의 정확성). 지원 제한이 h 일 때, 최적 포트폴리오가 \mathcal{X}_h 고 기대 효용 $f_j t_j$ 가 가장 높은 h 개의 학교의 집합이 \mathcal{T}_h 라고 하자. 그러면 $v(\mathcal{T}_h)/v(\mathcal{X}_h) \geq 1/h$ 이다.

증명. \mathcal{T}_h 는 $E[\sum_{j \in \mathcal{T}_h} \{t_j Z_j\}]$ 를 최대화하므로,

$$\begin{aligned} v(\mathcal{X}_h) &= E[\max_{j \in \mathcal{X}_h} \{t_j Z_j\}] \leq E[\sum_{j \in \mathcal{X}_h} \{t_j Z_j\}] \leq E[\sum_{j \in \mathcal{T}_h} \{t_j Z_j\}] \\ &= h E[\frac{1}{h} \sum_{j \in \mathcal{T}_h} \{t_j Z_j\}] \leq h E[\max_{j \in \mathcal{T}_h} \{t_j Z_j\}] = h v(\mathcal{T}_h). \end{aligned} \quad (5)$$

단, 마지막 부등호는 $\max\{\cdot\}$ 연산의 볼록성에 따라 성립한다. \square

다음 예는 근사 계수가 타이트(tight)함을 보인다.

예 2. 임의의 h 를 택하고 $m = 2h$ 라고 하자. 작은 상수 $\varepsilon \in (0, 1)$ 에 대해 시장을 다음처럼 구성한다.

$$\begin{aligned} t &= \left(\underbrace{1, \dots, 1}_h, \underbrace{\varepsilon^{-1}, \varepsilon^{-2}, \dots, \varepsilon^{-(h-1)}, \varepsilon^{-h}}_h \right) \\ \text{and } f &= \left(\underbrace{1, \dots, 1}_h, \underbrace{\varepsilon^1, \varepsilon^2, \dots, \varepsilon^{h-1}, \varepsilon^h}_h \right) \end{aligned}$$

모든 $f_j t_j = 1$ 이므로 나이브 해법은 $\mathcal{T}_h = \{1, \dots, h\}$ 를 출력할 수 있으며 $v(\mathcal{T}_h) = 1$ 이다. 그러나 최적해는 $\mathcal{X}_h = \{h+1, \dots, m\}$ 이며 그의 기대 효용은 다음과 같다.

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

따라서 $\varepsilon \mid 0$ 에 가까워지면서 $v(\mathcal{T}_h)/v(\mathcal{X}_h) \rightarrow 1/h \mid$ 된다. (\mathcal{X}_h 의 최적성은 정리 5의(가) 제시하는 상한을 실천하기 때문에 성립한다.)

따름정리 1. $v(\mathcal{X})$ 는 하위 모듈 함수가 아니다.

증명. $v(\mathcal{X})$ 가 하위 모듈 함수라면 Fisher 외(1978)의 정리 4.2는 나이브 해법이 $v(\mathcal{T}_h)/v(\mathcal{X}_h) \geq 1 - \left(\frac{h-1}{h}\right)^h$ 같은 근사 비율을 이룬다고 의미한다. 예 2는 반례다. \square

나이브 해법은 최적해가 아니지만 h 와 m 의 다향 시간 안에 최적해를 구할 수 있음을 보이자.

3.2 포함 사슬 관계

알마의 문제의 최적해에는 특별한 구조가 있다. 크기 $h+1$ 에 대한 최적 포트폴리오는 항상 크기 h 에 대한 최적 포트폴리오를 부분집합으로 포함한다.

정리 3 (최적 포트폴리오의 포함 사슬 관계).

$$\mathcal{X}_1 \subset \mathcal{X}_2 \subset \cdots \subset \mathcal{X}_m. \quad (6)$$

각 \mathcal{X}_h 가 지원 제한 h 에 대한 최적 포트폴리오며 위의 포함 사슬 관계를 만족하는 포트폴리오 수열 $\{\mathcal{X}_h\}_{h=1}^m$ 가 존재한다.

증명. h 에 대해 귀납법을 적용한다. 정리 1에 따라 $t_0 = 0 \mid$ 라고 가정한다.

(기본 경우.) 우선 $\mathcal{X}_1 \subset \mathcal{X}_2$ 임을 증명하자. 모순을 보이기 위해 최적해가 $\mathcal{X}_1 = \{j\}$ 와 $\mathcal{X}_2 = \{k, l\}$ 이라고 하자. 여기서 $t_k \leq t_l \mid$ 라고 가정할 수 있다. 최적성에 따라 다음이 성립한다:

$$v(\mathcal{X}_1) = f_j t_j > v(\{k\}) = f_k t_k \quad (7)$$

그리고

$$\begin{aligned} v(\mathcal{X}_2) &= f_k(1-f_l)t_k + f_l t_l > v(\{j, l\}) \\ &= f_j(1-f_l)t_j + (1-f_j)f_l t_l + f_j f_l \max\{t_j, t_l\} \\ &\geq f_j(1-f_l)t_j + (1-f_j)f_l t_l + f_j f_l t_l \\ &= f_j(1-f_l)t_j + f_l t_l \\ &\geq f_k(1-f_l)t_k + f_l t_l = v(\mathcal{X}_2) \end{aligned} \quad (8)$$

이는 모순이다.

(추론.) $\mathcal{X}_1 \subset \cdots \subset \mathcal{X}_h$ 라고 가정하고 $\mathcal{X}_h \subset \mathcal{X}_{h+1}$ 임을 보이자. $k = \arg \max\{t_k : k \in \mathcal{X}_{h+1}\}$ 으로 정의해서 $\mathcal{X}_{h+1} = \mathcal{Y}_h \cup \{k\}$ 으로 표현하자.

$k \notin \mathcal{X}_h$ 인 경우를 고려하자. 모순을 보이기 위해 $v(\mathcal{Y}_h) < v(\mathcal{X}_h)$ 라고 하면

$$\begin{aligned} v(\mathcal{X}_{h+1}) &= v(\mathcal{Y}_h \cup \{k\}) \\ &= (1-f_k)v(\mathcal{Y}_h) + f_k t_k \\ &< (1-f_k)v(\mathcal{X}_h) + f_k E[\max\{t_k, X_h\}] \\ &= v(\mathcal{X}_h \cup \{k\}) \end{aligned} \quad (9)$$

이는 \mathcal{X}_{h+1} 의 최적성을 위반한다.

이제 $k \in \mathcal{X}_h$ 인 경우를 고려하자. 그러면 크기 $h-1$ 인 어떤 포트폴리오 \mathcal{Y}_{h-1} 에 대해 $\mathcal{X}_h = \mathcal{Y}_{h-1} \cup \{k\}$ 으로 표현할 수 있다. $\mathcal{Y}_{h-1} \subset \mathcal{Y}_h$ 임을 보이면 충분하다. 정의에 따라 \mathcal{Y}_{h-1} (\mathcal{Y}_h)는 크기가 $h-1$ (h)인 포트폴리오 중에 함수 $v(\mathcal{Y} \cup \{k\})$ 를 최대화한다. 다시 말해 \mathcal{Y}_{h-1} 과 \mathcal{Y}_h 는 각각 단일 원소 포트폴리오 $\{k\}$ 와 최적으로 ‘보완적인’ 학교 집합이다.

함수 $w(\mathcal{Y})$ 를 이용해서 $\mathcal{Y} \subseteq \mathcal{C} \setminus \{k\}$ 인 포트폴리오와 $\{k\}$ 사이의 ‘보완성’을 평가하자. $w(\mathcal{Y})$ 를 구성하기 위해, 알마가 c_j 에 합격한 상황에서 c_k 에 지원했을 때 c_j 에서 받는 조건부 기대 효용을 \tilde{t}_j 라고 하자. $j = 0$ 을 포함한 $j < k$ 에 대해 이값은 $\tilde{t}_j = t_j(1 - f_k) + t_k f_k$ 이며, $j > k$ 에 대해 $\tilde{t}_j = t_j$ 임을 알 수 있다. 따라서

$$v(\mathcal{Y} \cup \{k\}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \tilde{t}_j p_j(\mathcal{Y}). \quad (10)$$

\tilde{t} 로 변환하면 t_j 의 순서가 변하지 않기 때문에 (10)의 우변 그 자체가 포트폴리오 가치 함수이다. t 를 \tilde{t} 로, \mathcal{C} 를 $\mathcal{C} \setminus \{k\}$ 로 대체하면 대응하는 시장이 된다. 이제 또 하나의 변환을 적용한다. $t_0 = 0$ 이 되도록 $\bar{t}_j = \tilde{t}_j - \tilde{t}_0$ 으로 정의하고 $w(\mathcal{Y})$ 를 다음과 같이 얻을 수 있다.

$$w(\mathcal{Y}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \bar{t}_j p_j(\mathcal{Y}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \tilde{t}_j p_j(\mathcal{Y}) - \tilde{t}_0 = v(\mathcal{Y} \cup \{k\}) - f_k t_k \quad (11)$$

두번째 등호는 정리 1에 따라 성립한다. 이 등호는, $\{k\}$ 와 최적으로 보완적인 \mathcal{Y}_{h-1} 과 \mathcal{Y}_h 는 목적함수가 $w(\mathcal{Y})$ 인 시장에서 각각 크기가 $h-1$ 과 h 인 최적 포트폴리오임을 의미한다. 새로운 시장에서 $\tilde{t}_0 = 0$ 이므로 귀납법 가설을 적용하면 $\mathcal{Y}_{h-1} \subset \mathcal{Y}_h$ 임을 알 수 있다. 따라서 증명이 완성된다.¹ □

3.3 다항 시간 해법

위의 결과를 적용하면 효율적인 최적 포트폴리오 탐욕 해법을 얻는다: 공집합으로 시작하고 $v(\mathcal{X} \cup \{k\})$ 를 최대화하는 학교를 차례대로 추가한다. t 를 배열하는 시간을 $O(m \log m)$ 이다. 모든 h 개의 반복단계에서 k 의 후보자의 개수는 $O(m)$ 이며 (3)을(를) 사용하면 $v(\mathcal{X} \cup \{k\})$ 를 계산하는 시간은 $O(h)$ 이다. 따라서 이 해법의 계산 시간이 $O(h^2 m + m \log m)$ 임을 알 수 있다. 정리 3 증명의 추론 단계에서 제시한 변환을 활용하면 계산 시간을 $O(hm)$ 으로 감소시킬 수 있다. \mathcal{X} 에 k 를 추가하면, 후보자 집합인 $\mathcal{C} \setminus \mathcal{X}$ 에서 k 를 제거하고 남은 학교의 t_j -값은 다음처럼 수정한다.

$$\bar{t}_j = \begin{cases} (1 - f_k)t_j, & t_j \leq t_k \\ t_j - f_k t_k, & t_j > t_k \end{cases} \quad (12)$$

이것이 증명에서 정의한 두개의 변환(t 에서 \tilde{t} 로, 그리고 \tilde{t} 에서 \bar{t} 로)의 합성임을 쉽게 확인할 수 있다. 그러면 다음으로 추가하는 학교는 수정된 시장의 최적 단일 원소 포트폴리오가 되어야 한다. 그런데 최적 단일 원소 포트폴리오는 단순히 $f_j \bar{t}_j$ 가 가장 높은 학교로 이루어진다. 따라서 각 반복 단계에서 (12)을(를) 이용해서 t_j -값을 수정하면 $v(\mathcal{X})$ 를 계산할 필요가 완전히 없어진다. 게다가 이 알고리즘에서 t_j 를 배열할 필요가 없으므로 $O(m \log m)$ 인 배열 비용이 사라진다.

아래 알고리즘은 알마가 지원하는 h 개의 학교를 목록 \mathbf{X} 로 출력한다. 그의 원소들은 진입하는 순서대로 등장한다. 즉, $h = m$ 으로 알고리즘을 돌리면 크기가 h 인 최적 포트폴리오는 $\mathcal{X}_h =$

¹이 유용한 변환을 발견한 임세호에게 감사를 표한다.

$\{X[1], \dots, X[h]\}$ 와 같다. 또한 목록 V 의 원소들은 해당 포트폴리오의 가치다.

Algorithm 1: 알마의 문제를 위한 최적 포트폴리오 알고리즘.

Data: 효용 모수 $t \in (0, \infty)^m$, 합격 확률 $f \in (0, 1]^m$, 지원 제한 $h \leq m$.

```

1  $C \leftarrow \{1 \dots m\};$ 
2  $X, V \leftarrow$  빈 목록;
3 for  $i = 1 \dots h$  do
4    $k \leftarrow \arg \max_{j \in C} \{f_j t_j\};$ 
5    $C \leftarrow C \setminus \{k\};$ 
6    $\text{append!}(X, k);$ 
7   if  $i = 1$  then  $\text{append!}(V, f_k t_k)$  else  $\text{append!}(V, V[i - 1] + f_k t_k);$ 
8   for  $j \in C$  do
9     if  $t_j \leq t_k$  then  $t_j \leftarrow (1 - f_k)t_j$  else  $t_j \leftarrow t_j - f_k t_k;$ 
10  end
11 end
12 return  $X, V$ 
```

정리 4 (알고리즘 1의 타당성). 알고리즘 1은(는) $O(hm)$ -시간 안에 알마의 문제를 위한 최적 지원 포트폴리오를 출력한다.

증명. 최적성은 정리 3의 증명에 따라 성립한다. C 를 목록 구현으로 저장한다고 하자. 그러면 각 h 개의 반복단계에서 최적 추가 학교를 구하는 시간은 $O(m)$ 이며 나머지 $O(m)$ 개 학교의 t_j -값을 각 단위 시간으로 수정할 수 있다. 따라서 전체 시간 복잡도가 $O(hm)$ 이다. \square

(5절에서 결과를 제시할) 수리 실험에서 C 를 목록 대신 이진 최대 힙 구현으로 저장하는 방법도 고려한다. 단, 힙의 순서는 $i \geq j \iff f_i t_i \geq f_j t_j$ 의 조건으로 정의한다. 기술적으로, t_j -값을 수정할 때마다 힙의 구조를 다시 복구해야 하므로 이는 주 반복단계의 계산 비용을 $O(hm)$ 에서 $O(hm \log m)$ 으로 증가시킨다. 그러나 전형적인 시장에서는, 반복단계 사이의 $f_j t_j$ -값의 순서가 크게 변하지 않으므로 이 상황은 자주 발생하지 않는다. 또한 피보나치 힙을 사용하면 t_j -값을 수정하는 시간을 단위 시간으로 줄여서 동등한 계산 시간을 얻어 낼 수 있다 (Fredman and Tarjan 1987).

3.4 최적 포트폴리오의 성질

포함 사슬 관계 성질은 알마의 기대 효용이 h 의 이산 오목 함수임을 의미한다.

정리 5 (최적 포트폴리오 가치의 h -오목성). $h = 2 \dots (m - 1)$ 에 대해,

$$v(\mathcal{X}_h) - v(\mathcal{X}_{h-1}) \geq v(\mathcal{X}_{h+1}) - v(\mathcal{X}_h). \quad (13)$$

증명. $2v(\mathcal{X}_h) \geq v(\mathcal{X}_{h+1}) + v(\mathcal{X}_{h-1})$ 같은 동등한 부등식을 증명하자. 정리 3을(를) 적용하면 $\mathcal{X}_h =$

$\mathcal{X}_{h-1} \cup \{j\}$ 그리고 $\mathcal{X}_{h+1} = \mathcal{X}_{h-1} \cup \{j, k\}$ 으로 표현할 수 있다. $t_k \leq t_j$ 인 경우,

$$\begin{aligned}
 2v(\mathcal{X}_h) &= v(\mathcal{X}_{h-1} \cup \{j\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
 &\geq v(\mathcal{X}_{h-1} \cup \{k\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
 &= v(\mathcal{X}_{h-1} \cup \{k\}) + (1 - f_j)v(\mathcal{X}_{h-1}) + f_j E[\max\{t_j, X_{h-1}\}] \\
 &= v(\mathcal{X}_{h-1} \cup \{k\}) - f_j v(\mathcal{X}_{h-1}) + f_j E[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\
 &\geq v(\mathcal{X}_{h-1} \cup \{k\}) - f_j v(\mathcal{X}_{h-1} \cup \{k\}) + f_j E[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\
 &= (1 - f_j)v(\mathcal{X}_{h-1} \cup \{k\}) + f_j E[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\
 &= v(\mathcal{X}_{h-1} \cup \{j, k\}) + v(\mathcal{X}_{h-1}) \\
 &= v(\mathcal{X}_{h+1}) + v(\mathcal{X}_{h-1}).
 \end{aligned} \tag{14}$$

첫번째 부등식은 \mathcal{X}_h 의 최적성에 따르며, 두번째 부등식은 \mathcal{X}_{h-1} 에 j 를 더하면 가치가 증가할 수 밖에 없기 때문이다.

$t_k \geq t_j$ 인 경우는 유사하다:

$$\begin{aligned}
 2v(\mathcal{X}_h) &= v(\mathcal{X}_{h-1} \cup \{j\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
 &\geq v(\mathcal{X}_{h-1} \cup \{k\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
 &= (1 - f_k)v(\mathcal{X}_{h-1}) + f_k E[\max\{t_k, X_{h-1}\}] + v(\mathcal{X}_{h-1} \cup \{j\}) \\
 &= v(\mathcal{X}_{h-1}) - f_k v(\mathcal{X}_{h-1}) + f_k E[\max\{t_k, X_{h-1}\}] + v(\mathcal{X}_{h-1} \cup \{j\}) \\
 &\geq v(\mathcal{X}_{h-1}) - f_k v(\mathcal{X}_{h-1} \cup \{j\}) + f_k E[\max\{t_k, X_{h-1}\}] + v(\mathcal{X}_{h-1} \cup \{j\}) \\
 &= v(\mathcal{X}_{h-1}) + (1 - f_k)v(\mathcal{X}_{h-1} \cup \{j\}) + f_k E[\max\{t_k, X_{h-1}\}] \\
 &= v(\mathcal{X}_{h-1}) + v(\mathcal{X}_{h-1} \cup \{j, k\}) \\
 &= v(\mathcal{X}_{h-1}) + v(\mathcal{X}_{h+1})
 \end{aligned} \tag{15}$$

위 결과는 \mathcal{X}_h 가 어떤 시장의 최적 h -포트폴리오일 때, $v(\mathcal{X}_h)$ 가 $O(h)$ 함수라고 의미한다. 예2에서 $v(\mathcal{X}_h)$ 를 h 에 임의로 가깝게 조정할 수 있으므로 이 상한이 타이트함을 알 수 있다.

3.5 작은 예

$m = 8$ 개의 학교로 구성된 가상 입학 시장을 고려하자. 학교 자료와 각 $h \leq m$ 에 대응하는 최적해는 표 1에서 나타난다.

아래에서 알고리즘 1의 몇몇 반복단계가 나타난다. f_j , t_j , 그리고 그 곱의 값은 \mathcal{C} 에 남아 있는 학교에 해당하는 값만 기록한다. $(f * t)_j = f_j t_j$ 로 정의된 $f * t$ 는 f 와 t 의 원소 당 곱을 의미한다.

각 반복단계에서 최적해에 추가하는 학교는 $f_j t_j$ -값이 가장 높은 학교이며 이를 밑줄로 강조한다.

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

알고리즘의 출력은 $X = [4, 2, 8, 1, 7, 3, 5, 6]$ 이며 최적 h -포트폴리오는 그의 첫 h 개의 원소로 이루어진다. 표 1에서 등장하는 “지원 순위” 자료는 X 의 역순열(inverse permutation)이며 이는 해당 학교가 최적 포트폴리오에 포함되는 최소 h -값을 의미한다. 그림 1은 최적 포트폴리오의 가치를 h 의 함수로 나타낸다. 곡선의 오목성은 정리 5의 결과를 시사한다.

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

표 1: $m = 8$ 개의 학교로 이루어진 가상 입학 시장의 대학교 자료와 최적 지원 포트폴리오. 포함 사슬 관계 성질(정리 3)에 따라, 지원 제한이 h 일 때 최적 포트폴리오는 지원 순위가 h 이하인 h 개의 학교로 구성된다.

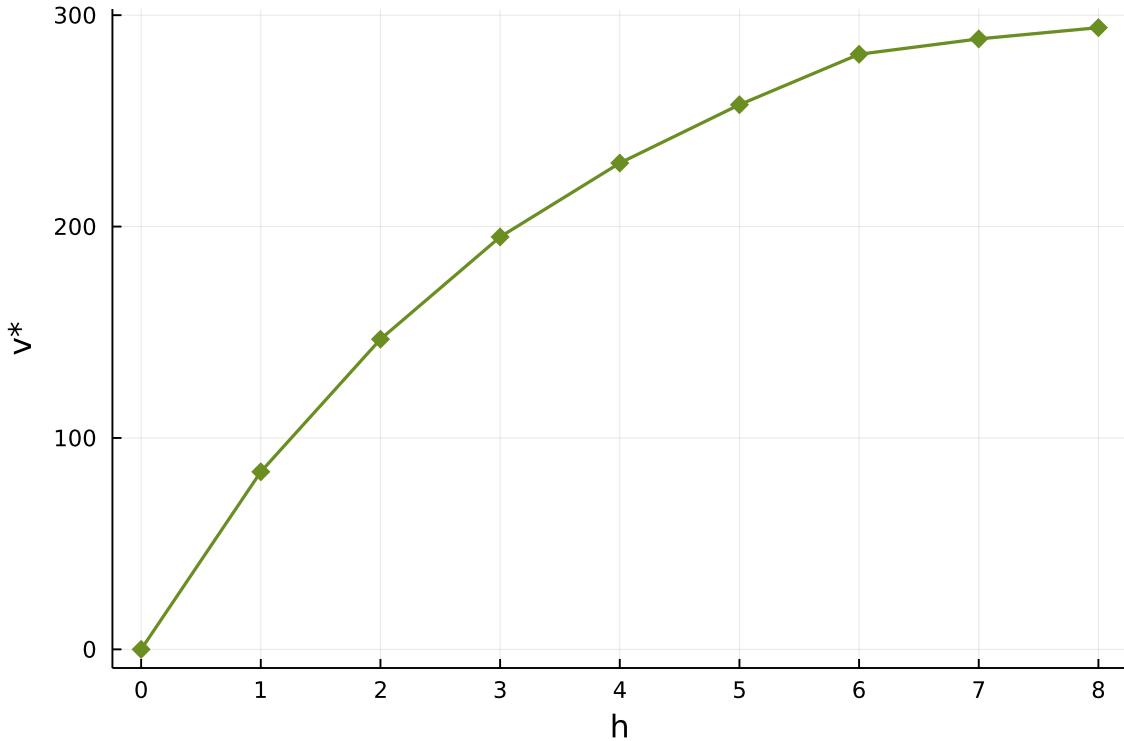


그림 1: $m = 8$ 개의 학교로 구성된 가상 입학 시장에서, 각 지원 제한 h 에 해당하는 최적 포트폴리오의 가치 $v^* = v(\mathcal{X}_h)$. 곡선의 오목성은 정리 5의 결과를 시사한다.

4 동일하지 않은 지원 비용

이제 g_j 가 c_j 에 지원하는 비용을 나타내며 학생이 지원에 쓸 수 있는 예산이 H 인 일반적인 문제를 고려한다. 이 문제를 풀고자 하는 학생을 엘리스라고 부르자. 본 절 내내 정리 1을(를) 적용해서 $t_0 = 0$ 임을 가정하자.

문제 2 (엘리스의 문제). 엘리스의 최적 대학 지원 포트폴리오는 다음 조합 최적화 문제의 최적해이다.

$$\begin{aligned} \text{maximize } & v(\mathcal{X}) = \sum_{j \in \mathcal{X}} \left(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right) \\ \text{subject to } & \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j \in \mathcal{X}} g_j \leq H \end{aligned} \tag{16}$$

본 절에서 이 문제가 NP-complete임을 보이고 3가지 해법을 제시한다: 정확한 분지한계법 해법, 정확한 동적 계획 해법, 그리고 완전 다향 시간 근사 해법 (fully polynomial-time approximation scheme, FPTAS).

4.1 NP-completeness

엘리스 문제의 최적해는 필수적으로 포함 사슬 관계를 가지지 않으며 최적 포트폴리오에 속한 학교의 개수는 H 에 대해 감소할 수도 있다. 예를 들어 $f = (0.5, 0.5, 0.5)$, $t = (1, 1, 219)$, 그리고 $g = (1, 1, 3)$ 이면 $H = 2$ 에 대응하는 최적해가 $\{1, 2\}$ 이며 $H = 3$ 에 대응하는 최적해는 $\{3\}$ 임을 쉽게 알 수 있다. 사실은 엘리스의 문제가 NP-complete이며 이를 이진 배낭 문제에서의 변환을 통해 증명할 수 있다. 배낭 문제의 NP-completeness가 이미 알려져 있다 (Garey and Johnson 1979).

문제 3 (배낭 문제의 결정 형태). ‘인스턴스’는 m 개 상품으로 이루어진 집합 \mathcal{B} , 각 $j \in \mathcal{B}$ 에 대한 효용값 $u_j \in \mathbb{N}$ 과 무게 $w_j \in \mathbb{N}$, 목적 효용 $U \in \mathbb{N}$, 그리고 배낭 용량 $W \in \mathbb{N}$ 로 구성된다. 인스턴스가 ‘예-인스턴스’가 될 필수충분 조건은 $\sum_{j \in \mathcal{B}'} u_j \geq U$ 와 $\sum_{j \in \mathcal{B}'} w_j \leq W$ 를 만족하는 집합 $\mathcal{B}' \subseteq \mathcal{B}$ 가 존재하는 것이다.

문제 4 (엘리스 문제의 결정 형태). ‘인스턴스’는 엘리스 문제의 인스턴스와 목적 포트폴리오 가치 V 로 구성된다. 인스턴스가 ‘예-인스턴스’가 될 필수충분 조건은 $v(\mathcal{X}) \geq V$ 와 $\sum_{j \in \mathcal{X}} g_j \leq H$ 를 만족하는 포트폴리오 $\mathcal{X} \subseteq \mathcal{C}$ 가 존재하는 것이다.

정리 6. 엘리스 문제의 결정 형태는 *NP-complete*다.

증명. 문제가 NP에 속하는 것은 명백하다.

임의의 배낭 문제 인스턴스를 택하고, 예-인스턴스 여부가 동치인 문제 4의 인스턴스를 만들자. 일반성을 잃지 않고 \mathcal{B} 에 속한 상품이 u_j 가 증가하는 순서대로 배열되며 모든 $u_j > 0$ 이고 $w_j \leq W$ 임을 가정할 수 있다.

$U_{\max} = \sum_{j \in \mathcal{B}} u_j$ 와 $\delta = 1/mU_{\max} > 0$ 일 때, $\mathcal{C} = \mathcal{B}$, $H = W$, 모든 $f_j = \delta$, 그리고 각 $t_j = u_j/\delta$ 인 엘리스 문제의 인스턴스를 고려하자. $\mathcal{X} \subseteq \mathcal{C}$ 가 엘리스 문제의 가능해가 되는 것은 배낭 문제 인스턴스의 가능해가 되는 것과 동치임을 쉽게 알 수 있다. 그러면 공집합이 아닌 임의의 \mathcal{X} 에 대해,

$$\begin{aligned} \sum_{j \in \mathcal{X}} u_j &= \sum_{j \in \mathcal{X}} f_j t_j > \sum_{j \in \mathcal{X}} \left(f_j t_j \prod_{\substack{j' \in \mathcal{X}: \\ j' > j}} (1 - f_{j'}) \right) = v(\mathcal{X}) \\ &= \sum_{j \in \mathcal{X}} \left(u_j \prod_{\substack{j' \in \mathcal{X}: \\ j' > j}} (1 - \delta) \right) \geq (1 - \delta)^m \sum_{j \in \mathcal{X}} u_j \\ &\geq (1 - m\delta) \sum_{j \in \mathcal{X}} u_j \geq \sum_{j \in \mathcal{X}} u_j - m\delta U_{\max} = \sum_{j \in \mathcal{X}} u_j - 1. \end{aligned} \tag{17}$$

이는 지원 포트폴리오 \mathcal{X} 가 해당 배낭 문제에서 일으키는 효용이 $v(\mathcal{X})$ 보다 크면서 가장 작은 정수 임을 의미한다. 즉, $\sum_{j \in \mathcal{X}} u_j \geq U$ 가 성립할 필수충분 조건은 $v(\mathcal{X}) \geq U - 1$ 이다. $V = U - 1$ 으로 정의해서 변환을 완성하면 증명이 완료된다. \square

알마 문제를 위한 탐욕 해법의 직관적인 확장은 $[v(\mathcal{X} \cup \{k\}) - v(\mathcal{X})]/g_k$ -값이 가장 높은 학교 k 를 \mathcal{X} 에 차례대로 추가하는 것이다. 그러나 위에서 도출한 변환은 엘리스 문제의 목적함수를 배낭 문제의 목적함수의 원하는 만큼 정확한 근사로 만들 수 있음을 의미한다. 따라서 다음 같은 병례에서 탐욕 해법은 임의로 낮은 근사 비율을 일으킬 수 있다.

예 3. $t = (10, 2021)$, $f = (1, 1)$, $g = (1, 500)$, 그리고 $H = 500$ 이라고 하자. 그러면 탐욕 해법은 $\mathcal{X} = \{1\}$ 을 출력하며 이는 분명히 최적해가 아니다.

4.2 분지한계법

정수 최적화의 고전적인 기법 중 분지한계법(branch and bound)이 있다. 이는 하나 이상의 결정 변수를 고정시키고 해당 부문제를 탐색하는 해법이며, 목적함수의 상한(bound)을 활용해서 기준 해 보다 좋은 해를 생성할 수 없는 결정 나무의 가지(branch)를 제외시켜 결정 공간을 줄인다 (Martello and Toth 1990; Kellerer et al. 2004). 본 항에서, 엘리스 문제의 정수 모형과 그의 상한이 되는 선형 계획을 제시한다. 알고리즘 1에서 개발한 t_j -값의 변환을 활용해서 부문제의 선형 완화 문제에 의한 상한을 더 타이트하게 만든다. 이 기반으로 분지한계법 알고리즘을 도출한다.

우선 포트폴리오 \mathcal{X} 를 이진 특성 벡터 $x \in \{0, 1\}^m$ 로 표현하자. 단, $x_j = 1$ 인 것은 $j \in \{X\}$ 과 동치다. 그러면 엘리스의 문제가 다음 비선형 정수 최적화 문제와 동등함을 볼 수 있다.

문제 5 (엘리스의 문제를 위한 비선형 정수 계획).

$$\begin{aligned} \text{maximize } & v(x) = \sum_{j=1}^m \left(f_j t_j x_j \prod_{i>j} (1 - f_i x_i) \right) \\ \text{subject to } & \sum_{j=1}^m g_j x_j \leq H, \quad x_i \in \{0, 1\}^m \end{aligned} \tag{18}$$

$v(x)$ 에서 등장하는 곱은 1을 넘을 수 없으므로 다음 선형 완화 문제는 최적 포트폴리오 가치의 상한이 돈다.

문제 6 (엘리스의 문제를 위한 선형 완화 문제).

$$\begin{aligned} \text{maximize } & v_{LP}(x) = \sum_{j=1}^m f_j t_j x_j \\ \text{subject to } & \sum_{j=1}^m g_j x_j \leq H, \quad x \in [0, 1]^m \end{aligned} \tag{19}$$

문제 6은(는) “연속 배낭 문제”라고 불리며 다음 탐욕 알고리즘을 적용하면 $O(m \log m)$ -시간 안에 쉽게 풀 수 있다: $x = \mathbf{0}$ 로 초기화한다. $H > 0$ 가 성립하는 동안, $f_j t_j / g_j$ -값이 가장 높은 학교 j 를 택한다. $x_j \leftarrow \min\{1, H/g_j\}$ 그리고 $H \leftarrow H - g_j$ 같이 수정해서 반복한다 (Dantzig 1957).

본 연구의 분지한계법 설계에서, ‘마디’(node)는 학교를 세 군으로 나눈 분할 $\mathcal{C} = \mathcal{I} \cup \mathcal{O} \cup \mathcal{N}$ 로 묘사한다. 단, $\sum_{j \in \mathcal{I}} g_j \leq H$ 을 만족한다. \mathcal{I} 는 지원 포트폴리오에 속한 학교(‘in’: $x_j = 1$), \mathcal{O} 는 속하지 않은 학교(‘out’: $x_j = 0$), 그리고 \mathcal{N} 은 미결의 학교(‘negotiable’)로 이루어진다. 분할이 결정되면 부문제의 한 쌍이 결정해진다. 첫번째 부문제는 다음 같은 문제 5의 인스턴스다.

$$\begin{aligned} \text{maximize } & v(x) = \gamma + \sum_{j \in \mathcal{N}} \left(f_j \bar{t}_j x_j \prod_{\substack{i \in \mathcal{N}: \\ i > j}} (1 - f_i x_i) \right) \\ \text{subject to } & \sum_{j \in \mathcal{N}} g_j x_j \leq \bar{H}; \quad x_j \in \{0, 1\}, \quad j \in \mathcal{N}. \end{aligned} \tag{20}$$

두번째 부문제는 대응하는 문제 6 인스턴스다.

$$\begin{aligned} \text{maximize } & w_{LP}(x) = \gamma + \sum_{j \in \mathcal{N}} f_j \bar{t}_j x_j \\ \text{subject to } & \sum_{j \in \mathcal{N}} g_j x_j \leq \bar{H}; \quad x_j \in [0, 1], \quad j \in \mathcal{N} \end{aligned} \tag{21}$$

각 부문제에서 $\bar{H} = H - \sum_{j \in \mathcal{I}} g_j$ 는 잔여 예산을 의미한다. \mathcal{I} 에 속한 학교에 (12)의 변환을 적용해서 모수 γ 와 \bar{t} 를 얻을 수 있다. 즉, 각 $j \in \mathcal{I}$ 에 대해 γ 를 $f_j \bar{t}_j$ 만큼 증가시키고, 시장에서 j 를 소거하고, (12)에 따라 남은 \bar{t}_j -값을 수정한다. (아래에서 예를 제시한다.)

마디 $n = (\mathcal{I}, \mathcal{O}, \mathcal{N})$ 이 주어질 때, 그의 새끼 마디를 다음처럼 생성한다. 모든 마디에는 2, 1, 혹은 0개의 새끼가 있다. 전형적인 경우에서, $g_j \leq \bar{H}$ 를 만족하는 학교 $j \in \mathcal{N}$ 선택한다. j 를 \mathcal{I} 로 이동시켜서 한 새끼를 얻고, j 를 \mathcal{O} 로 이동시켜서 다른 새끼를 얻는다. 즉, 한 새끼 마디에서 $x_j = 1$ 로 고정시키도 다른 새끼 마디에서 $x_j = 0$ 으로 고정시킨다. 일반적으로 j 는 임의의 학교가 될 수 있지만 탐욕적인 휴리스틱으로 $f_j \bar{t}_j / g_j$ 의 비율이 가장 높은 학교를 선택한다. 이런 식으로 새끼를 생성하면 각 마디의 \mathcal{I} -집합은 자신의 어미 마디의 \mathcal{I} -집합과 최대 한 학교로만 다르다. 따라서 새로운 마디를 생성할 때, (12)을(를) 한번만 적용하면 상수 γ 와 변화된 \bar{t}_j -값을 얻을 수 있다.

이례적인 경우 2가지 있다. 첫번째는, \mathcal{N} 에 속한 모든 학교에 대해 $g_j > \bar{H}$ 이면 가능성을 유지하며 \mathcal{I} 에 추가할 수 있는 학교가 없으므로 본 가지의 최적 포트폴리오는 단지 \mathcal{I} 자체다. 이때 \mathcal{N} 에 원소를 모두 \mathcal{O} 로 이동시켜서 단일 새끼를 생성한다. 두번째는, $\mathcal{N} = \emptyset$ 인 마디는 새끼가 없다. 더 이상 분지할 수 없으므로 이 마디를 ‘잎 마디’(leaf)라고 한다.

예 4. $t = (20, 40, 60, 80, 100)$, $f = (0.5, 0.5, 0.5, 0.5, 0.5)$, $g = (3, 2, 3, 2, 3)$, 그리고 $H = 8$ 인 시장과 마디 $n = (\mathcal{I}, \mathcal{O}, \mathcal{N}) = (\{2, 5\}, \{1\}, \{3, 4\})$ 이 주어질 때 n 에 대응하는 부문제와 새끼 마디를 구하자. 부문제를 계산하기 위해 먼저 c_1 을 단순히 무시한다. c_2 를 소거하기 위해 t 에 (12)에 적용해서 $\{\bar{t}_3, \bar{t}_4, \bar{t}_5\} = \{(1-f_2)t_3, (1-f_2)t_4, (1-f_2)t_5\} = \{30, 40, 50\}$ 과 $\bar{\gamma} = f_2 t_2 = 20$ 을 얻는다. c_5 를 소거하기 위해 \bar{t} 에 (12)에 다시 적용해서 $\{\bar{t}_3, \bar{t}_4\} = \{\bar{t}_3 - f_5 \bar{t}_5, \bar{t}_4 - f_5 \bar{t}_5\} = \{5, 15\}$ 와 $\gamma = \bar{\gamma} + f_5 \bar{t}_5 = 35$ 를 얻는다. 마지막으로, $\bar{H} = H - g_2 - g_5 = 3$ 이다. 이제 (20)과(와) (21)인 부문제를 쉽게 구할 수 있다.

\mathcal{N} 에 속한 학교 중에 $g_j \leq \bar{H}$ 인 학교가 있으므로 n 은 새끼 마디 2개 있다. $f_j \bar{t}_j / g_j$ -비율이 가장 높은 학교는 c_4 이므로 새끼 마디를 생성하는 법칙을 적용하면 $n_1 = (\{2, 4, 5\}, \{1\}, \{3\})$ 과 $n_2 = (\{2, 5\}, \{1, 4\}, \{3\})$ 같은 새끼를 얻는다.

분지한계법을 실행하기 위해 후보 마디, 즉 잎이 아니면서 새끼를 아직 생성하지 않은 마디의 집합을 \mathfrak{T} 라고 부르자. 마디 $n = (\mathcal{I}, \mathcal{O}, \mathcal{N})$ 를 생성할 때마다, $v_{\mathcal{I}}[n] = v(\mathcal{I})$ 과 선형 완화 문제 (21)의 최대값 $v_{LP}^*[n]$ 를 기록한다. \mathcal{I} 는 예산에 가능한 포트폴리오이므로 $v_{\mathcal{I}}[n]$ 는 원래 목적함수 최대값의 하한이 된다. 또한 정리 3의 증명 과정에서 보였듯, (20)의 목적함수는 함수 $v(\mathcal{I} \cup \mathcal{X})$ 과 동치다. 이는 $v_{LP}^*[n]$ 는 \mathcal{I} 를 부분집합으로 포함하며 \mathcal{O} 에 속한 학교를 포함하지 않은, 즉 이 가지에 있는 모든 포트폴리오의 가치에 대한 상한임을 의미한다. 따라서 새로운 마디 n_2 를 생성했을 때, 만약 그의 목적함수값 $v_{\mathcal{I}}[n_2]$ 가 어떤 다른 마디 n_1 에 대응하는 $v_{LP}^*[n_1]$ 보다 크다면, n_1 과 그의 모든 후손을 무시할 수 있다.

\mathfrak{T} 에 뿐만 아니라 마디 $n_0 = (\emptyset, \emptyset, \mathcal{C})$ 를 넣어서 알고리즘을 초기화한다. 각 반복단계에서 $v_{LP}^*[n]$ -값이 가장 높은 마디 $n \in \mathfrak{T}$ 을 선택한다. n 의 새끼를 생성하고 후보 집합에서 제거한다. 그다음에 n 의 각 새끼 마디 n' 을 검증한다. 그중에 새로운 최적해를 발견하면 이를 가장 좋은 후보로 표하고 $v_{\mathcal{I}}[n'] > v_{LP}^*[n'']$ 인 마디 n'' 을 후보 집합에서 제거한다. 후보 집합이 공집합이 되면 알고리즘이 모든 가지를 탐색했으므로 가장 좋은 후보를 출력하고 종료한다.

정리 7 (알고리즘 2의 타당성). 알고리즘 2은(는) 엘리스 문제의 최적 지원 포트폴리오를 출력한다.

증명. 나무의 잎 마디 중 최적해가 반드시 존재하므로 위의 논의에 따라 알고리즘이 종료한다면 최적해를 출력하는 것을 알 수 있다.

알고리즘에서 회로가 생기지 않음을 보이기 위해 어떤 한 마디가 두번 생성되지 않는 것을 증명하면 충분하다. 모순을 보이기 위해 같은 분할 $(\mathcal{I}_{12}, \mathcal{O}_{12}, \mathcal{N}_{12})$ 를 가지는 상이한 마디 n_1 과 n_2 가 생성된다고 하자. 나무에 올라가면서 두 마디의 가계가 서로 만나는 첫 마디를 n 이라고 하자. n 은 새끼 마디 하나만 가지면 그 자체가 n_1 과 n_2 의 공통 선조가 되며 이는 모순이므로 n 새끼의 개수가 2임을 알 수 있다. 또한 그 중 하나는 n_1 의 선조며 다른 하나는 n_2 의 선조다. 각각 $n_3 = (\mathcal{I}_3, \mathcal{O}_3, \mathcal{N}_3)$ 그리고 $n_4 = (\mathcal{I}_4, \mathcal{O}_4, \mathcal{N}_4)$ 라고 하자. 마디 생성 규칙에 따라 $\mathcal{I}_3 \cap \mathcal{O}_4$ 에 속한 학교 j 가 존재하며, \mathcal{I}_3 (\mathcal{O}_4)의 모든 후손 마디의 \mathcal{I} -집합 (\mathcal{O} -집합)은 \mathcal{I}_3 (\mathcal{O}_4)의 확대 집합이다. 따라서 $j \in \mathcal{I}_{12} \cap \mathcal{O}_{12}$ 가 되며 이는 $(\mathcal{I}_{12}, \mathcal{O}_{12}, \mathcal{N}_{12})$ 가 \mathcal{C} 의 분할이 아님을 의미한다. 모순. \square

분지한계법은 기술적으로 유리한 기준점이지만, 일종의 열거 해법이므로 계산 시간이 문제 크기와 지수적으로 늘어난다. 그리고 뒤에서 제안하는 근사 해법과 달리, 주어진 반복한계의 개수에 대한 정확성 보장이 없다. 또한 \mathcal{N} 에 학교가 많을 때, 선형 완화 문제 상한은 $v_{\mathcal{I}}[n']$ 보다 매우 높을 수도 있다. 이는 해법이 나무에 어느 정도 깊게 파고들어야 마디를 제외시킬 수 있음을 의미하며, 그때 계산 노력의 대부분은 이미 쏟은 것이다. 수리 실험에서, 알고리즘 2은(는) 약 $m = 20$ 개의 학교를 넘는 인스턴스에 대해 비효율적이었으며 이는 단순한 열거법에 비해 큰 개선이 아니다.

Algorithm 2: 엘리스의 문제를 위한 분지한계법.

Data: 효용 모수 $t \in (0, \infty)^m$, 합격 확률 $f \in (0, 1]^m$, 지원 비용 $g \in (0, \infty)^m$, 예산 $H \in (0, \infty)$.

- 1 뿌리 마디 $n_0 \leftarrow (\emptyset, \emptyset, \mathcal{C})$;
- 2 현재 하한 $L \leftarrow 0$ 및 최적해 $\mathcal{X} \leftarrow \emptyset$;
- 3 후보 집합 $\mathfrak{T} \leftarrow \{n_0\}$;
- 4 **while** 종료되지 않음 **do**
- 5 **if** $\mathfrak{T} = \emptyset$ **then return** \mathcal{X}, L ;
- 6 **else**
- 7 $n \leftarrow \arg \max \{v_{LP}^*[n] : n \in \mathfrak{T}\}$;
- 8 \mathfrak{T} 에서 n 을 제거한다;
- 9 **for** n 의 각 새끼 마디 n' **do**
- 10 **if** $L < v_{\mathcal{I}}[n']$ **then**
- 11 $L \leftarrow v_{\mathcal{I}}[n']$;
- 12 \mathcal{X} 을 n' 에 대응하는 \mathcal{I} -집합으로 수정한다;
- 13 **end**
- 14 **if** n' 이 앞 마디 아님 **then** \mathfrak{T} 에 n' 을 추가한다;
- 15 **end**
- 16 **for** $n'' \in \mathfrak{T}$ **do**
- 17 **if** $L > v_{LP}^*[n'']$ **then** \mathfrak{T} 에서 n'' 을 제거한다;
- 18 **end**
- 19 **end**
- 20 **end**

4.3 의사 다행 시간 동적 계획

본 절에서 일반성을 약간 제한해서 $g_j \in \mathbb{N}$ for $j = 1 \dots m$ 과 $H \in \mathbb{N}$ 임을 가정한다. 이때 엘리스의 문제를 위한 $O(Hm + m \log m)$ -시간 해법을 제시한다. 해법은 이진 배낭 문제를 위한 익숙한 동적 계획 해법과 비슷하다 (Dantzig 1957; Wikipedia, s.v. “Knapsack problem”). 알마의 문제와 달리, 여기에서 $H \leq m$ 임을 가정할 수 없으므로 이는 의사 다행 시간 해법이라고 한다 (Garey and Johnson 1979, § 4.2).

$j = 0 \dots m$ 와 $h = 0 \dots H$ 에 대해, $\{1, \dots, j\}$ 에 속한 학교만 사용하면서 최대한 h 의 지원 비용을 지불하는 최적 포트폴리오를 $\mathcal{X}[j, h]$ 라고하자. 또한 $V[j, h] = v(\mathcal{X}[j, h])$ 이다. $j = 0$ 혹은 $h = 0$ 이면 $\mathcal{X}[j, h] = \emptyset$ 이고 $V[j, h] = 0$ 이 되는 것은 분명하다. 편의상 $h < 0$ 이면 $V[j, h] = -\infty$ 이라고 정의하자.

남은 지표에 대해, $\mathcal{X}[j, h]$ 는 j 를 포함하거나 포함하지 않는다. j 를 포함하지 않는다면 $\mathcal{X}[j, h] = \mathcal{X}[j-1, h]$ 이다. $\mathcal{X}[j, h]$ 가 j 를 포함한다면, 그의 가치는 $(1 - f_j)v(\mathcal{X}[j, h] \setminus \{j\}) + f_j t_j$ 다. 따라서 $\mathcal{X}[j, h] \setminus \{j\}$ 는 남은 예산과 남은 학교에 대한 최적 포트폴리오다. 즉, $\mathcal{X}[j, h] = \mathcal{X}[j-1, h-g_j] \cup \{j\}$ 이다. 이 관찰에 따라 $j = 1 \dots m$ 와 $h = 1 \dots H$ 에 대해 다음 같은 Bellman 식을 얻는다.

$$V[j, h] = \max\{V[j-1, h], (1 - f_j)V[j-1, h-g_j] + f_j t_j\} \quad (22)$$

단, $-\infty \cdot 0 = -\infty$ 으로 처리한다. 그러면 $\mathcal{X}[j, h]$ 가 j 를 포함할 필수충분 조건이 $V[j, h] > V[j-1, h]$ 임을 관찰하면 대응되는 최적 포트폴리오를 쉽게 계산할 수 있으며 전역 최적해는 $\mathcal{X}[m, H]$ 이다. 아래 알고리즘은 이 계산을 수행하고 최적 포트폴리오 \mathcal{X} 를 출력한다.

정리 8 (알고리즘 3의 타당성). 알고리즘 3은(는) $O(Hm + m \log m)$ -시간 안의 엘리스 문제의 최적 포트폴리오를 출력한다.

증명. 위 논의에 따라 최적성이 성립한다. t 를 배열하는 시간은 $O(m \log m)$ 이다. 목병 단계는 2줄에

Algorithm 3: 정수 지원 비용의 엘리스 문제를 위한 동적 계획 해법.

Data: 효용 모수 $t \in (0, \infty)^m$, 합격 확률 $f \in (0, 1]^m$, 지원 비용 $g \in \mathbb{N}^m$, 예산 $H \in \mathbb{N}$.

- 1 t 의 순서대로 학교를 배열한다;
- 2 $V[j, h]$ -값으로 표를 채운다;
- 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}

서 $V[j, h]$ 의 표를 만드는 것이다. 각 원소를 단위 시간 안에 생성할 수 있으며 표의 크기가 $O(Hm)$ 이다. \square

4.4 완전 다행 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 $r[x] = 10^{-P} \lfloor 10^P x \rfloor$ denote the value of x rounded down to its nearest fixed-point representation. Since $\bar{U} = \sum_{j \in C} f_j t_j$ 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:

$$\mathcal{V} = \left\{ 0, 1 \times 10^{-P}, 2 \times 10^{-P}, \dots, r[\bar{U} - 1 \times 10^{-P}], r[\bar{U}] \right\} \quad (23)$$

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 $\{1, \dots, j\}$ 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

$$G[j, v] = \begin{cases} \infty, & t_j < v \\ \min\{G[j - 1, v], g_j + G[j - 1, v - \Delta_j(v)]\}, & t_j \geq v \end{cases} \quad (24)$$

$$\text{where } \Delta_j(v) = \begin{cases} r \left[\frac{f_j}{1-f_j} (t_j - v) \right], & f_j < 1 \\ \infty, & f_j = 1 \end{cases} \quad (25)$$

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

$$v(\mathcal{W}[j, v]) = (1 - f_j)v(\mathcal{W}[j, v] \setminus \{j\}) + f_j t_j. \quad (26)$$

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

$$G[j, v] = \min\{G[j - 1, v], g_j\}. \quad (27)$$

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  $j = m, m - 1, \dots, 1$  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 (24) and (25), if $j \in \mathcal{W}[j, v]$, then the true value of $(1 - f_j)v(\mathcal{W}[j - 1, v - \Delta_j(v)]) + f_j t_j$ 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 (26). 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

$$v(\mathcal{W}) \geq v'(\mathcal{W}) \geq v'(\mathcal{X}) \geq v(\mathcal{X}) - m10^{-P} \geq \left(1 - \frac{m^2 10^{-P}}{\bar{U}}\right) v(\mathcal{X}) \geq (1 - \varepsilon) v(\mathcal{X}) \quad (28)$$

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

$$|\mathcal{V}| = \bar{U} \times 10^P + 1 = \bar{U} \times 10^{\lceil \log_{10}(m^2/\varepsilon\bar{U}) \rceil} + 1 \leq \frac{m^2}{\varepsilon} \times \text{const.} \quad (29)$$

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 test the time complexity results established above.

5.1 실험 방법

The experimental procedure was as follows. First, to generate synthetic markets, we drew the t_j -values 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 $[0, 1]$. In the first experiment, which concerns Alma's problem, we set each $g_j = 1$ and $H = h = \lfloor m/2 \rfloor$. In the second experiment, which concerns Ellis's problem, each g_j is drawn uniformly from the set $\{5, \dots, 10\}$ and we set $H = \lfloor \frac{1}{2} \sum g_j \rfloor$. A typical instance is shown in Figure 2.

For each combination of the experimental variables, we generated 20 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 20 markets. Therefore, each cell of each table below represents a statistic over 60 computations. Where applicable, we do not count the time required to sort the entries of t .

5.2 결과 정리

In our first experiment, we compare the performance of Algorithm 1 for homogeneous-cost markets of various sizes when the set of candidate schools \mathcal{C} is stored as a list and as a binary max heap ordered by the $f_j t_j$ -values. The results appear in Table 2. Although using a max heap increases the nominal runtime from $O(hm)$ to $O(hm \log m)$, we hypothesized that in typical instances, the order of the heap would change only slightly between iterations, yielding an overall savings because of the lower costs of extracting the maximal school. Our results indicate that due to its lower overhead cost, the list implementation is faster; however, the ratio between the average times of the two implementations stays roughly constant at about 1.5, suggesting that a more effective heap implementation could be competitive in certain classes of problem instances. Overall, the rate of growth is quadratic in m , which accords with the $O(hm)$ time complexity result of Theorem 4.

In the second experiment, we turn to the general problem, and compare 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, agree with the time complexity analyses presented above. Overall, we found the exact dynamic program to be the fastest algorithm, a result that echoes the results of computational studies on knapsack problems. The branch-and-bound algorithm proved impractical for even medium-sized instances.

5.3 알고리즘의 이해

We chose to implement our algorithms in the Julia language (v1.7.0) because its system of type inference 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). The code is available at <https://github.com/maxkapur/OptimalApplication>.

학교의 개수 m	알고리즘 1 (목록 구현)	알고리즘 1 (힙 구현)
16	0.07 (0.01)	0.09 (0.02)
64	0.88 (0.16)	1.18 (0.21)
256	12.51 (2.20)	18.39 (2.90)
1024	281.79 (58.56)	387.97 (96.41)
4096	4661.75 (1152.54)	7218.04 (1733.91)

표 2: 알고리즘 1에서 \mathcal{C} 를 목록 또는 힙 구현으로 저장했을 때, 동일 지원 비용 입학 시장의 최적 지원 포트폴리오를 계산하는 시간. 각 m 에 대해 20개의 시장을 생성했으며 알고리즘을 3번 반복해서 그 중 최소 계산 시간을 기록했다. 표에서 20개의 인스턴스에 대한 평균 (평균편차) 시간을 밀리초 단위로 나타난다. 모든 경우, $h = m/2$.

학교의 개수 m	알고리즘 2: 분지한계법	알고리즘 3: 지원 비용 동적 계획	알고리즘 4: FPTAS, $\varepsilon = 0.5$	알고리즘 4: FPTAS, $\varepsilon = 0.05$
8	0.44 (0.42)	0.06 (0.02)	0.39 (0.16)	1.66 (0.64)
16	3298.08 (5285.45)	0.48 (0.16)	2.64 (0.96)	19.20 (7.05)
32	— (—)	2.30 (0.83)	19.01 (9.18)	184.00 (70.15)
64	— (—)	8.33 (3.52)	101.05 (59.47)	2089.30 (961.43)
128	— (—)	36.61 (15.36)	481.61 (210.68)	8626.29 (2615.38)
256	— (—)	205.57 (52.98)	3106.71 (957.95)	69291.93 (34881.04)
512	— (—)	1136.18 (396.24)	16613.99 (6241.99)	206020.31 (45217.42)

표 3: 4절에서 도출한 3개의 알고리즘을 사용할 때, 다양한 지원 비용으로 갖춘 입학 시장의 최적 또는 $(1 - \varepsilon)$ -최적 포트폴리오를 계산하는 시간. 분지한계법은 큰 시장에서 비실용적이다. 각 m 에 대해 20개의 시장을 생성했으며 알고리즘을 3번 반복해서 그 중 최소 계산 시간을 기록했다. 표에서 20개의 인스턴스에 대한 평균 (평균편차) 시간을 밀리초 단위로 나타난다.

The dynamic programs, namely Algorithms 3 and 4, were implemented using recursive functions and memoization rather than a full lookup table. Our implementation of Algorithm 4 also differed from that described in section 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 t_j -value by 2^P . These modifications yielded a substantial performance improvement without changing the fundamental algorithm design or complexity analysis.

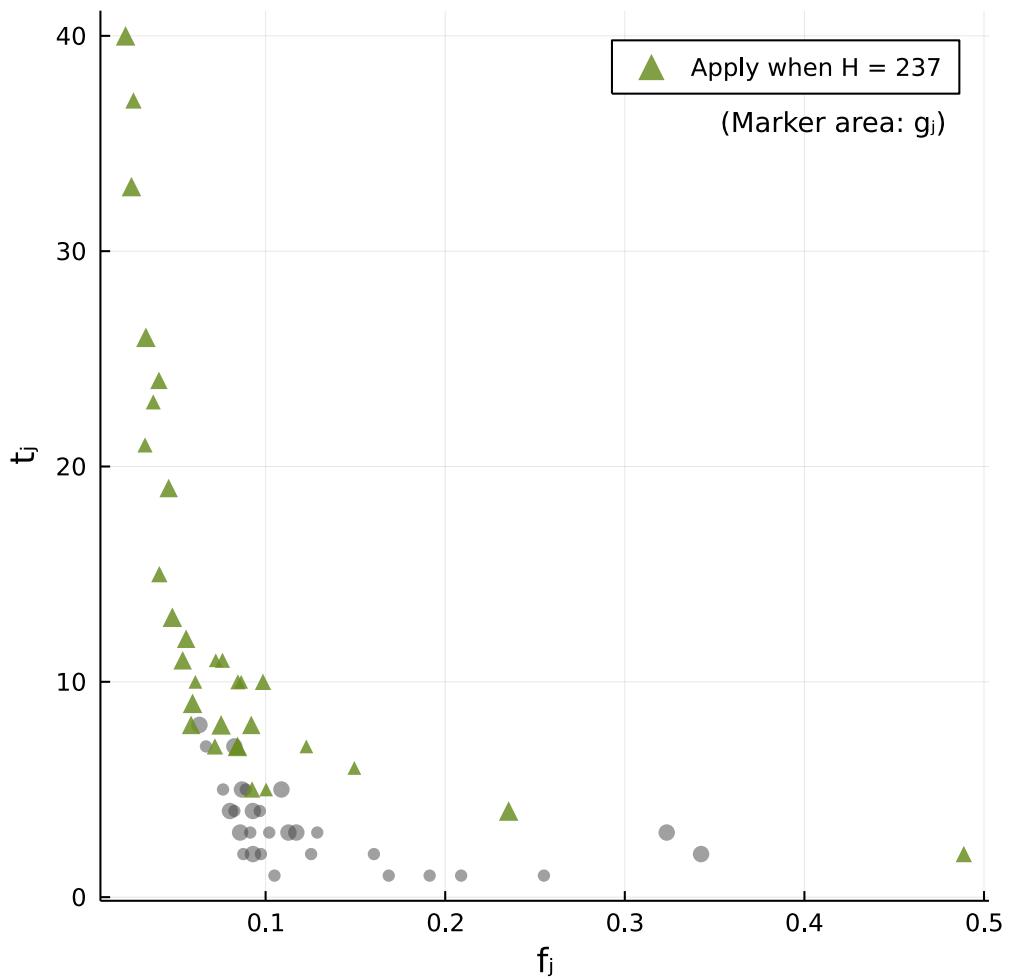


그림 2: $m = 60$ 개의 학교로 구성된 전형적인 무작위로 생성한 인스턴스와 해당 최적 포트폴리오. 지원 비용 g_j 는 $\{5, \dots, 10\}$ 에서 균일한 확률로 설정했으며 기호의 넓이와 비례된다. 최적 포트폴리오는 알고리즘 3(으)로 계산했다.

6 결론

7 참고문헌

- 김민희. 2015. “[대입 수시 전략] 총 6번의 기회 … ‘상향·소신·안정’ 분산 지원하라.” *중앙일보*, 8 월 26일. <https://www.joongang.co.kr/article/18524069>.
- 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>.
- 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.