

# 대학 입학 지원 최적화 문제

Max Kapur

서울대학교  
산업공학과  
경영과학/최적화 연구실  
이메일: maxkapur@gmail.com  
지도 교수: 홍성필

June 23, 2022

## Abstract

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

**Keywords:** submodular maximization, knapsack problems, approximation algorithms

The full text of this paper is available in English at <https://github.com/maxkapur/CollegeApplication>.

## 요약

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

## Contents

<b>1 서론</b>	<b>3</b>
1.1 방법론적 지향 . . . . .	3
1.2 본 논문의 구성 . . . . .	5
<b>2 표기법과 예비 결과</b>	<b>5</b>
2.1 목적함수의 정제 . . . . .	5
2.2 변수 소거 기법 . . . . .	6
2.3 목적함수의 submodularity . . . . .	7
<b>3 동일한 지원 비용</b>	<b>7</b>
3.1 나이브 해법의 근사 성질 . . . . .	7
3.2 포함 사슬 관계 . . . . .	8
3.3 다향 시간 해법 . . . . .	9
3.4 지원의 수학 체감 . . . . .	10
3.5 작은 예 . . . . .	10
<b>4 동일하지 않은 지원 비용</b>	<b>11</b>
4.1 NP-completeness . . . . .	12
4.2 분지한계법 . . . . .	14
4.3 의사 다향 시간 동적 계획 . . . . .	16
4.4 완전 다향 시간 근사 해법 . . . . .	17
4.5 모의 담금질 휴리스틱 . . . . .	19
<b>5 계산 실험</b>	<b>20</b>
5.1 알고리즘의 구현 . . . . .	20
5.2 실험 방법 . . . . .	20
5.3 결과 정리 . . . . .	21
<b>6 결론과 향후 연구</b>	<b>21</b>
6.1 위험 회피를 모수화한 모형 . . . . .	23
6.2 시그널 전략과 matroid 제약 조건 . . . . .	24
6.3 동적 계획의 메모리 소요 절감 . . . . .	25
<b>A 부록</b>	<b>26</b>
A.1 정리 5의 기초적 증명 . . . . .	26
<b>참고문헌</b>	<b>27</b>

## 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 모수다. 각  $j = 1 \dots m$ 에 대해  $g_j > 0$ 는 비용 모수이며  $Z_j$ 는 확률  $f_j$ 를 가지는 서로 독립적인 Bernoulli 변수다. 각  $j = 0 \dots m$ 에 대해  $t_j \geq 0$ 는 효용 모수다.

다음 해석에 따라 이를 ‘대학 입학 지원 최적화 문제’라고 부른다.  $m$ 개의 대학교를 가지는 입학 시장을 고려하자.  $j$ 번째 학교의 이름은  $c_j$ 다. 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. 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 최댓값을 최대화하는 것이다. 위 문제의 최적해가  $\mathcal{X}$ 일 때, 학생의 최적 대학 지원 전략은  $\mathcal{X}$ 에 속한 학교로 지원하는 것이다.

대학 입학 지원 최적화 문제는 이론적인 화제일 뿐만이 아니다. 미래의 소득이 대학 입학 결과로 결정된다는 넓은 인식에 따라, (1)의 최적해는 금전적인 가치를 가지고 있다. 미국 입학 컨설팅 산업에서 대학 지원 서류 작성, 합격 확률 추정, 그리고 지원 학교를 선택하는 데에 자문하는 개인 상담가의 시간당 급료는 평균 200달러다 (Sklarow 2018). 한국에서, 메가스터디(megastudy.net)와 진학(jinhak.com)과 같은 입학 컨설팅 기업의 주된 수입원 중 인공지능을 활용하여 학생의 지원 전략을 최대화한다고 홍보하는 “모의 지원” 소프트웨어가 있다. 그러나, 학교 개별 합격 확률을 추정하는 것은 로지스틱 회귀 분석 모형의 익숙한 응용 사례(Acharya et al. 2019)이지만, 전체적인 지원 전략에 집중하여 최적화 문제로 모형화한 것은 본 연구의 새로운 기여로 보인다.

취직 전략과 같은 유사한 경쟁적 매칭 게임에도 위의 문제를 적용할 수 있다. 이때, 예산 제약 조건은 원서를 작성하는 시간이나 지원 개수에 대한 법적인 제한을 나타낼 수 있다.

### 1.1 방법론적 지향

대학 지원 문제는 다양한 방법론적인 우주를 걸치는 문제다. 확률적인 문제인 만큼 재정학 분야에 뿐만 아니라 포트폴리오 배분 모형과 비슷하다. 그러나 배낭 제약 조건은 대학 지원 문제를 NP-complete하게 만들며 조합적인 해법이 필요하다. 목적함수는 또한 submodular 집합 함수이지만, 본 연구가 제시하는 근사 해법 결과는 대학 지원 문제가 일반 submodular 함수 최대화 문제의 비교적 쉬운 경우임으로 해석할 수 있다.

미국 대학 입학 시장의 균형 분석에서, Fu (2014)는 (1)과(와) 비슷한 부문제를 직면하였으며 이를 “nontrivial portfolio problem”이라고 부른다 (226). 재정학 분야에서, 고전적 포트폴리오 배분 최적화 모형은 전체 자산에 대한 기대 총이익에서 위험회피 항을 뺀으로 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 학생의 효용을 각 지원 발송의 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 높으며 붙기 어려운 “상향” 지원 학교(reach school)와 선호도가 낮으며 붙기 쉬운 “안정” 지원 학교(safety school) 사이의 균형을 고려해야 한다 (전민희 2015). 마지막, 대학 지원의 조합적인 본질로

인해 연속적인 포트폴리오 최적화 문제에서 흔히 사용하는 기울기 해법으로는 풀기가 어렵다. Fu는 지원 비용을 제약 조건 대신 목적함수의 한 항으로 모형화했으며, 균형 모형의 모수를 추정하기 위해  $m = 8$ 이 되도록 학교들의 클러스터를 먼저 구성했다. 이는 열거법을 통해 문제 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 수 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 외 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  $H$ 로 모수화된 해의 순서를 유도하며 그의 원소들은 ‘포함 사슬 관계’(nestedness)로 연결된다. 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 아래서 submodular한 집합 함수를 최대화하는 문제가 대표적인 예다 (Fisher 외 1978; Assad 1985). 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 고정 소수점 산술을 활용한 완전 다항 시간 근사 해법(fully polynomial-time approximation scheme, FPTAS)을 제시한다.

마지막으로, (1)의 목적함수는 submodular하며 단조 증가하는 함수다. 그런데, 본 연구는 더 기초적인 분석 기술을 이용하며, 일반적인 submodular 최대화 알고리즘보다 좋은 근사 계수를 얻게 된다. 예를 들어 Fisher 외 (1978)의 잘 알려진 결과에 따라, 대학 지원 문제의  $g_j = 1$ 인 경우에서 탐욕 해법의 점근적 근사 계수가  $(1 - 1/e)$ 이 되며, 본 논문에서 같은 해법이 정확함을 보인다. 일반적인 문제에 대해서는, 배낭 제약식 위에서 submodular 집합 함수를 최대하는 문제에서, 다양한 완화 기술을 활용하면 거의 동등한 근사 계수를 이뤄 낼 수 있다 (Chekuri et al. 2014; Badanidiyuru and Vondrák 2014; Kulik et al. 2013). 이 문제를 위한 다항 시간 해법은  $1 - 1/e$ 보다 좋은 근사 계수를 가질 수 없음이 알려져 있다 (Nemhauser와 Wolsey 1978). 그러나 대학 지원 문제에서, FPTAS의 존재성은 그보다 강한 결과다.

## 1.2 본 논문의 구성

2절은 표기법과 일반성을 제한하지 않는 편의적 가정을 제시한다. 유용한 변수 소거 기술도 제시하고 목적함수가 submodularity함을 증명한다.

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 이용하여  $O(hm)$  시간 정확한 해법을 도출한다.

4절에서 각 학교의 지원 비용이 동일하지 않은 일반적인 상황을 다룬다. 이진 배낭 문제에서의 다항 변환을 통하여 포트폴리오 최적화 문제가 NP-complete함을 증명한다. 또한 3개의 일반적인 해법을 제안한다. 첫째는 분지한계법이다. 둘째는 총 지출액으로 탐색하는  $O(Hm + m \log m)$ 과 같은 의사 다항 시간 동적 계획 해법이다. 셋째는 실수값을 분수값으로 내림한 포트폴리오 가치를 탐색하는 또 다른 동적 계획 해법이다. 이를 통하여  $O(m^3/\varepsilon)$  시간에  $(1 - \varepsilon)$ -근사해를 출력하는 FPTAS를 얻는다. 넷째는 가상 인스턴스를 푸는 성능이 좋은 모의 담금질(simulated annealing) 기반 휴리스틱 알고리즘이다.

5절에서 위에서 도출한 타당성과 계산 복잡성을 확인하는 계산 실험 결과를 정리한다. 모의 담금질 휴리스틱의 정확도도 탐구한다.

결론에서, 해법을 개선할 수 있는 요소를 언급하고, 실제 지원 과정의 특징을 묘사하도록 원래 모형을 확장할 수 있는 기술 몇 가지 소개한다.

## 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 잊지 않고 각  $g_j \leq H$ ,  $\sum g_j > H$ , 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 제시한다.

### 2.1 목적함수의 정제

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 정의 한다. 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 필요충분 조건은  $c_j$ 에 지원하고,  $c_j$ 에서 합격하고,  $c_j$ 보다 선호하는 (즉, 지표가 더 높은) 학교에서 불합격은 것이다. 따라서  $j = 0 \dots m$ 에 대해

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

단, 공집합의 곱은 1이면  $f_0 = 1$ 이다. 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  $t_0 = 0$ 이라고 가정할 수 있음을 보이자.

**기본 정리 1.** 어떤  $\gamma \leq t_0$ 을 선택하고  $j \dots m$ 에 대해  $\bar{t}_j = t_j - \gamma$ 라고 하자. 그러면  $\mathcal{X}$ 를 막론하고  $v(\mathcal{X}; \bar{t}_j) = v(\mathcal{X}; t_j) - \gamma$ 가 성립한다.

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.

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

## 2.2 변수 소거 기법

본 항에서 남은 논문에서 활용되는 변수 소거 기술을 소개한다.<sup>1</sup> 학생이  $c_k$ 에 지원하기로 결심하고, 남은 결정 공간은 어떤 다른 학교에 지원하는 것으로 제한된다고 하자. 전체 지원 포트폴리오를  $\mathcal{X} = \mathcal{Y} \cup \{k\}$ 일 때,  $\mathcal{Y} \subseteq \mathcal{C} \setminus \{k\}$ 인 포트폴리오와  $\{k\}$  사이의 ‘보완성’을 평가하는 함수  $w(\mathcal{Y})$ 를 도출할 수 있다. 단,  $v(\mathcal{Y} \cup \{k\})$ 와  $w(\mathcal{Y})$  사이에서  $f_k t_k$  같은 상수 차이가 발생한다.

$w(\mathcal{Y})$ 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

$$v(\mathcal{Y} \cup \{k\}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \tilde{t}_j p_j(\mathcal{Y}). \quad (4)$$

$\tilde{t}$ 로 변환하면  $t_j$ 의 순서가 변하지 않기 때문에 (4)의 우변 그 자체가 포트폴리오 가치 함수다.  $t$ 를  $\tilde{t}$ 로,  $\mathcal{C}$ 를  $\mathcal{C} \setminus \{k\}$ 로 대체하면 대응하는 시장이 된다. 그다음에  $t_0 = 0$ 이 되도록  $\bar{t}_j = \tilde{t}_j - \tilde{t}_0$ 으로 정의하고  $w(\mathcal{Y})$ 를 다음처럼 얻을 수 있다.

$$w(\mathcal{Y}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \bar{t}_j p_j(\mathcal{Y}) = \sum_{j \in \{0\} \cup \mathcal{Y}} \tilde{t}_j p_j(\mathcal{Y}) - \tilde{t}_0 = v(\mathcal{Y} \cup \{k\}) - f_k t_k \quad (5)$$

두 번째 등호는 기본 정리 1에 따라 성립한다. 다음 정리는 이 변환의 타당성을 의미한다. 단,  $w(\mathcal{Y})$ 가 포트폴리오 가치 함수 형태를 가지는 것을 강조하기 위해  $w(\mathcal{Y})$  대신  $v(\mathcal{X}; \bar{t})$ 처럼 표기한다.

**기본 정리 2** ( $c_k$  소거 기법).  $\mathcal{X} \subseteq \mathcal{C} \setminus \{k\}$ 에 대해,  $v(\mathcal{X} \cup \{k\}; t) = v(\mathcal{X}; \bar{t}) + f_k t_k$ 다. 단,

$$\bar{t}_j = \begin{cases} (1 - f_k)t_j, & t_j \leq t_k \\ t_j - f_k t_k, & t_j > t_k. \end{cases} \quad (6)$$

증명. (6)의(가) 위에서 논의한 2개의 변환( $t$ 에서  $\tilde{t}$ 로, 그리고  $\tilde{t}$ 에서  $\bar{t}$ 로)의 합성임을 쉽게 확인할 수 있다.  $\square$

<sup>1</sup>이 유용한 변환을 발견한 임세호에게 감사합니다.

학생이 다수의 학교에 지원하기로 결심한 경우를 반영하기 위해, (6)의 변화를 반복적으로 적용할 수 있다.

### 2.3 목적함수의 submodularity

목적함수가 submodular함을 보이자. 이 결과는 주로 분류학적인 목표로 제시하며 건너뛰어도 연속 성에 지장을 주지 않는다. 일반적인 submodular 최대화 알고리즘보다 본 연구가 제시하는 대학 지원 최적화 해법이 더 좋으며, 뒤에서 submodularity를 이용하지 않고 기초적인 분석을 통해 해법의 타당성을 증명한다.

**정의 2** (Submodular한 집합 함수). 고려 집합  $\mathcal{C}$  그리고 함수  $v : 2^{\mathcal{C}} \mapsto \mathbb{R}$ 가 주어질 때,  $v(\mathcal{X})$  가 submodular한 집합 함수가 되는 필요충분 조건은, 모든  $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{C}$ 에 대해  $v(\mathcal{X}) + v(\mathcal{Y}) \geq v(\mathcal{X} \cup \mathcal{Y}) + v(\mathcal{X} \cap \mathcal{Y})$ 이 성립하는 것이다. 또한 모든  $\mathcal{X} \subset \mathcal{C}$ 와  $k \in \mathcal{C} \setminus \mathcal{X}$ 에 대해  $v(\mathcal{X} \cup \{k\}) - v(\mathcal{X}) \geq 0$  이면,  $v(\mathcal{X})$ 가 단조 증가하는 submodular한 집합 함수라고 부른다.

**정리 1.** 대학 지원의 포트폴리오 가치 함수  $v(\mathcal{X})$ 는 submodular한 집합 함수다.

증명.  $v(\mathcal{X})$ 가 단조 증가하는 것은 자명하다. Submodularity를 보이기 위해, Nemhauser와 Wolsey (1978)의 제의 2.1.iii을 적용하여 모든  $\mathcal{X} \subset \mathcal{C}$  그리고  $j \neq k \in \mathcal{C} \setminus \mathcal{X}$ 에 대해 다음이 성립함을 보이자.

$$v(\mathcal{X} \cup \{j\}) - v(\mathcal{X}) \geq v(\mathcal{X} \cup \{j, k\}) - v(\mathcal{X} \cup \{k\}) \quad (7)$$

기본 정리 2을(를) 적용하면, (6)에 따라  $\mathcal{X}$ 에 속한 학교를 반복적으로 소거하여 모두  $\bar{t}$ 로 갖춘 포트 폴리오 함수  $w(\mathcal{Y})$ 를 얻을 수 있으며, 모든  $\mathcal{Y} \subseteq \mathcal{C} \setminus \mathcal{X}$ 에 대해  $w(\mathcal{Y}) = v(\mathcal{X} \cup \mathcal{Y}) + \text{const.}$ 가 성립한다. 따라서 (7)은(는) 다음 부등식과 동등하다.

$$w(\{j\}) - w(\emptyset) \geq w(\{j, k\}) - w(\{k\}) \quad (8)$$

$$\iff w(\{j\}) + w(\{k\}) \geq w(\{j, k\}) \quad (9)$$

$$\iff E[\bar{t}_j Z_j] + E[\bar{t}_k Z_k] \geq E[\max\{\bar{t}_j Z_j, \bar{t}_k Z_k\}] \quad (10)$$

마지막 부등식은 분명히 참이다.  $\square$

## 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 다향 시간 해법을 도출한다. 기본 정리 1을 (를) 적용하여 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 최적해다.

$$\begin{aligned} & \text{maximize} \quad v(\mathcal{X}) = \sum_{j \in \mathcal{X}} \left( f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right) \\ & \text{subject to} \quad \mathcal{X} \subseteq \mathcal{C}, \quad |\mathcal{X}| \leq h \end{aligned} \quad (11)$$

### 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 매력적 으로 보일 수 있지만 사실은 최적해가 아니다.

**정의 3** (알마의 문제를 위한 나이브 해법). 기대 효용  $f_j t_j$ 가 가장 큰  $h$ 개의 학교로 지원한다.

Blum 외 (1973)의 PICK 알고리즘을 사용하면 위의 해법의 계산 시간이  $O(m)$ 이다.

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

**정리 2** (나이브 해법의 정확성). 지원 제한이  $h$ 일 때, 최적 포트폴리오가  $\mathcal{X}_h$ 이고 기대 효용  $f_j t_j$ 가 가장 큰  $h$ 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,

$$\begin{aligned} v(\mathcal{X}_h) &= E\left[\max_{j \in \mathcal{X}_h} \{t_j Z_j\}\right] \leq E\left[\sum_{j \in \mathcal{X}_h} \{t_j Z_j\}\right] \leq E\left[\sum_{j \in \mathcal{T}_h} \{t_j Z_j\}\right] \\ &= h E\left[\frac{1}{h} \sum_{j \in \mathcal{T}_h} \{t_j Z_j\}\right] \leq h E\left[\max_{j \in \mathcal{T}_h} \{t_j Z_j\}\right] = hv(\mathcal{T}_h). \end{aligned} \quad (12)$$

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

다음 예는 근사 계수가 타이트(tight)함을 보인다.

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

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

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.

$$v(\mathcal{X}_h) = \sum_{j=h+1}^m \left( f_j t_j \prod_{j'=j+1}^m (1 - f_{j'}) \right) = \sum_{j=1}^h (1 - \varepsilon)^j \approx h \quad (13)$$

따라서  $\varepsilon$ 에 0에 가까워지면서  $v(\mathcal{T}_h)/v(\mathcal{X}_h) \rightarrow 1/h$ 이 된다. ( $\mathcal{X}_h$ 의 최적성은 정리 12의 (가) 제시하는 상한을 실천하기 때문에 성립한다.)

나이브 해법은 최적 해법이 아니지만  $O(hm)$  시간에 최적해를 구할 수 있다.

### 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).  $\mathcal{X}_h$ 가 지원 제한  $h$ 에 대한 최적 포트폴리오가 되면서 다음 포함 사슬 관계를 만족하는 포트폴리오 수열  $\{\mathcal{X}_h\}_{h=1}^m$ 이 존재한다.

$$\mathcal{X}_1 \subset \mathcal{X}_2 \subset \dots \subset \mathcal{X}_m \quad (14)$$

증명.  $h$ 에 대해 귀납법을 적용한다. 기본 정리 1에 따라  $t_0 = 0$ 이라고 가정하자.

(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  $t_k \leq t_l$ 이라고 가정할 수 있다. 최적성에 따라 다음이 성립한다:

$$v(\mathcal{X}_1) = f_j t_j > v(\{k\}) = f_k t_k \quad (15)$$

그리고

$$\begin{aligned}
 v(\mathcal{X}_2) &= f_k(1 - f_l)t_k + f_lt_l > v(\{j, l\}) \\
 &= f_j(1 - f_l)t_j + (1 - f_j)f_lt_l + f_jf_l \max\{t_j, t_l\} \\
 &\geq f_j(1 - f_l)t_j + (1 - f_j)f_lt_l + f_jf_lt_l \\
 &= f_j(1 - f_l)t_j + f_lt_l \\
 &\geq f_k(1 - f_l)t_k + f_lt_l = v(\mathcal{X}_2)
 \end{aligned} \tag{16}$$

이는 모순이다.

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

$k \notin \mathcal{X}_h$ 인 경우를 고려하자. 모순을 보이기 위해  $v(\mathcal{Y}_h) < v(\mathcal{X}_h)$  그리고  $v(\mathcal{X}_{h+1}) > v(\mathcal{X}_h \cup \{k\})$ 라고 가정하면 다음이 성립한다.

$$\begin{aligned}
 v(\mathcal{X}_{h+1}) &= v(\mathcal{Y}_h \cup \{k\}) \\
 &= (1 - f_k)v(\mathcal{Y}_h) + f_k t_k \\
 &\leq (1 - f_k)v(\mathcal{X}_h) + f_k E[\max\{t_k, X_h\}] \\
 &= v(\mathcal{X}_h \cup \{k\})
 \end{aligned} \tag{17}$$

모순이다.

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.

기본 정리 2을(를) 적용하여  $c_k$ 를 소거하고 남은  $t_j$ -값을 (6)에 따라  $\bar{t}_j$ 로 변환한다. 그러면  $\mathcal{Y} \subseteq \mathcal{C} \setminus \{k\}$ 인 포트폴리오와  $\{k\}$  사이의 보완성을 평가하는 함수  $w(\mathcal{Y})$ 를 얻는다.  $w(\mathcal{Y})$  그 자체가 포트폴리오 가치 함수이며  $\bar{t}_0 = 0$ 이므로, 귀납법 가설에 따라  $\mathcal{Y}_{h-1} \subset \mathcal{Y}_h$ 이고 증명이 완성된다.  $\square$

### 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 시간은  $O(m \log m)$ 이다. 모든  $h$ 개의 반복 단계에서  $k$ 의 후보자의 개수는  $O(m)$ 이며 (3)을(를) 사용하면  $v(\mathcal{X} \cup \{k\})$ 를 계산하는 시간은  $O(h)$ 다. 따라서 이 해법의 계산 시간이  $O(h^2m + m \log m)$ 임을 알 수 있다.

기본 정리 2에서 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 소거하고 남은 학교의  $t_j$ -값을 (6)에 따라 수정한다. 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 큰 학교로 이루어진다. 따라서 각 반복 단계에서 (6)을 (를) 이용하여  $t_j$ -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. 알고리즘 1은(는) 얼마가 지원하는  $h$ 개의 학교를 배열(array)  $\mathbf{x}$ 로 출력한다. 그의 원소들은 진입하는 순서대로 등장한다. 즉,  $h = m$ 으로 알고리즘을 돌리면 크기가  $h$ 인 최적 포트폴리오는  $\mathcal{X}_h = \{\mathbf{x}[1], \dots, \mathbf{x}[h]\}$ 와 같다. 또한 배열  $\mathbf{v}$

의 원소들은 해당 포트폴리오의 가치다.

---

**Algorithm 1:** 알마의 문제를 위한 최적 포트폴리오 알고리즘.

---

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

```

1  $\mathcal{C} \leftarrow \{1 \dots m\};$ 
2  $X, V \leftarrow$  빈  $h$ -배열;
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    $X[i] \leftarrow k;$ 
7   if  $i = 1$  then  $V[i] \leftarrow f_k t_k$  else  $V[i] \leftarrow V[i-1] + f_k t_k;$ 
8   for  $j \in \mathcal{C}$  do
9     if  $t_j \leq t_k$  then  $t_j \leftarrow (1 - f_k) t_j$  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)  $O(hm)$  시간에 알마의 문제를 위한 최적 지원 포트폴리오를 출력한다.

증명. 최적성은 정리 3의 증명에 따라 성립한다. 각  $h$ 개의 반복 단계에서 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.  $\square$

$\mathcal{C}$ 는 배열 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 정의하며, 각 반복 단계의 마지막 단계에서 힙을 비우고 다시 힙화(heapify)하면  $O(hm)$  계산 시간을 유지할 수 있다. 그러나 (5절에서 결과를 제시할) 수리 실험에서 배열 구현으로 구성한 알고리즘이 더 빨랐다. 효용 모수를 수정하면서  $k$ 를 구함으로  $\arg \max\{\}$  연산의 계산 비용을 거의 제거할 수 있기 때문이다.

### 3.4 지원의 수학 체감

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,

$$v(\mathcal{X}_h) - v(\mathcal{X}_{h-1}) \geq v(\mathcal{X}_{h+1}) - v(\mathcal{X}_h). \quad (18)$$

증명. 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. 최적성에 따라,  $v(\mathcal{X}_h) - v(\mathcal{X}_{h-1}) \geq v(\mathcal{X}_{h-1} \cup \{k\}) - v(\mathcal{X}_{h-1})$ . Submodularity 그리고 포함 사슬 관계 성질에 따라,  $v(\mathcal{X}_{h-1} \cup \{k\}) - v(\mathcal{X}_{h-1}) \geq v(\mathcal{X}_h \cup \{k\}) - v(\mathcal{X}_h) = v(\mathcal{X}_{h+1}) - v(\mathcal{X}_h)$ .  $\square$

(기초적인 증명은 § A.1에서 제시한다.) 위 결과는  $\mathcal{X}_h$ 가 어떤 시장의 최적  $h$ -포트폴리오일 때,  $v(\mathcal{X}_h)$ 가  $O(h)$  함수라고 의미한다. 다시 말해 알마 문제의 지원 예산에는 수학 체감 성질이 있다. 예 1에서  $v(\mathcal{X}_h)$ 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 몇 반복 단계가 나타난다.  $f_j, t_j$ , 그리고 그 곱의 값은  $C$ 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 반복 단계에서 최적해에 추가하는 학교는  $f_j t_j$ -값이 가장 큰 학교이며 이를 밑줄로 강조한다.

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

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 역 순열(inverse permutation)이며 이는 해당 학교가 최적 포트폴리오에 포함되는 최소한  $h$ -값을 의미한다. 그럼 1은 최적 포트폴리오의 가치를  $h$ 의 함수로 나타낸다. 곡선의 오목성은 정리 5의 결과를 시사한다.

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

## 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 기본 정리 1을(를) 적용하여  $t_0 = 0$ 임을 가정하자.

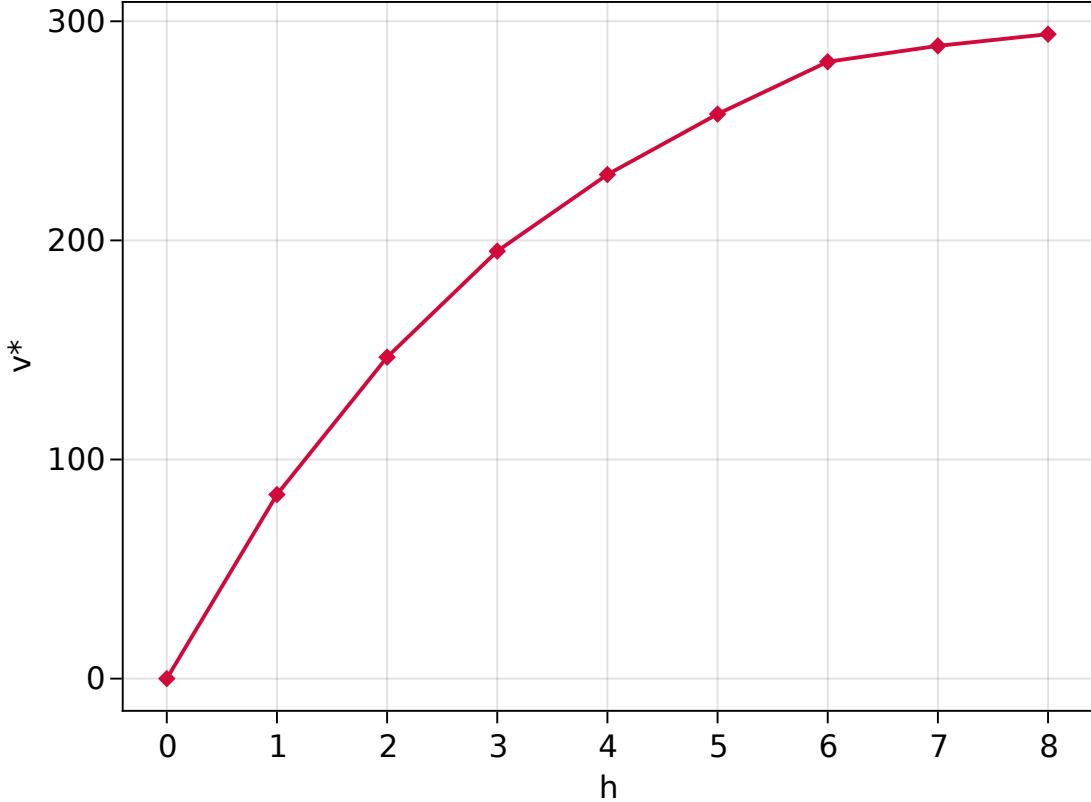


그림 1:  $m = 8$ 개의 학교로 구성된 가상 입학 시장에서, 각 지원 제한  $h$ 에 해당하는 최적 포트폴리오의 가치  $v^* = v(\mathcal{X}_h)$ . 곡선의 오목성은 정리 5의 결과를 시사한다.

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

$$\begin{aligned} \text{maximize } & v(\mathcal{X}) = \sum_{j \in \mathcal{X}} \left( f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right) \\ \text{subject to } & \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j \in \mathcal{X}} g_j \leq H \end{aligned} \tag{19}$$

본 절에서 이 문제가 NP-complete함을 보이고 3가지 해법을 제시한다: 정확한 분지한계법 해법, 정확한 동적 계획 해법, 완전 다항 시간 근사 해법(fully polynomial-time approximation scheme, FPTAS), 그리고 모의 담금질(simulated annealing) 휴리스틱.

#### 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-complete하며 이를 이진 배낭 문제에서의 변환을 통해 증명할 수 있다. 배낭 문제의 NP-completeness가 이미 알려져 있다 (Garey and Johnson 1979, § 3.2.1).

**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 필요충분 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 필요충분 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*하다.

증명. 문제가 NP에 속하는 것은 자명하다.

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,

$$\begin{aligned} \sum_{j \in \mathcal{X}} u_j &= \sum_{j \in \mathcal{X}} f_j t_j > \sum_{j \in \mathcal{X}} \left( f_j t_j \prod_{\substack{j' \in \mathcal{X}: \\ j' > j}} (1 - f_{j'}) \right) = v(\mathcal{X}) \\ &= \sum_{j \in \mathcal{X}} \left( u_j \prod_{\substack{j' \in \mathcal{X}: \\ j' > j}} (1 - \delta) \right) \geq (1 - \delta)^m \sum_{j \in \mathcal{X}} u_j \\ &\geq (1 - m\delta) \sum_{j \in \mathcal{X}} u_j \geq \sum_{j \in \mathcal{X}} u_j - m\delta U_{\max} = \sum_{j \in \mathcal{X}} u_j - 1. \end{aligned} \tag{20}$$

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 필요충분 조건은  $v(\mathcal{X}) \geq U - 1$ 이다.  $V = U - 1$ 으로 정의하여 변환을 완성하면 증명이 완료된다.  $\square$

알마 문제를 위한 탐욕 해법의 직관적인 확장은  $(v(\mathcal{X} \cup \{k\}) - v(\mathcal{X}))/g_k$ -값이 가장 큰 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 2.**  $t = (10, 2021)$ ,  $f = (0.1, 0.1)$ ,  $g = (1, 500)$ , 그리고  $H = 500$ 이라고 하자. 그러면 탐욕 해법은  $\mathcal{X} = \{1\}$ 을 출력하며 이는 분명히 최적해가 아니다.

엘리스 문제의 좀 더 좋은 근사해를 찾기 위해 다음 배낭 문제를 (NP-complete함을 무시하고) 대리 문제로 푸는 것에 매력은 있다.

$$\text{maximize } \sum_{j \in \mathcal{X}} f_j t_j \quad \text{subject to } \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j \in \mathcal{X}} g_j \leq H \tag{21}$$

그러나 이러한 해법은 사실 alma 문제의 나이브 해법(정의 3)의 단순한 일반화임을 알 수 있다. 엘리스 문제에서 최적 포트폴리오에 크기가  $m - 1$ 이 될 수 있으므로 배낭 문제 해법의 근사 계수는 (12)과(와) 유사하듯이  $1/(m - 1)$ 이다. 타이트한 예는 다음과 같다.

**예 3.** 엘리스 문제의 다음 인스턴스를 고려하자. 단,  $H = m - 1$ 이다.

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

가능해  $\mathcal{Y} = \{1, \dots, m-1\}$ 와  $\mathcal{X} = \{m\}$ 에 대해  $\sum_{j \in \mathcal{Y}} f_j t_j = \sum_{j \in \mathcal{X}} f_j t_j = 1$ 이므로 배낭 문제 해법은  $\mathcal{Y}$ 를 선택할 수 있다. 그의 가치는  $v(\mathcal{Y}) = 1/(m-1)$ 이다. 그러나  $\mathcal{X}$ 는 최적해이며 그의 가치는  $v(\mathcal{X}) = 1$ 이다.

탐욕 해법 또는 배낭 문제 기반 해법으로 엘리스 문제의 정확한 최적해를 구할 수 없음을 의미한다.

## 4.2 분지한계법

정수 최적화의 고전적인 기법 중 분지한계법(branch and bound)이 있다. 이는 하나 이상의 결정 변수를 고정하고 해당 부문제를 탐색하는 해법이며, 목적함수의 상한(bound)을 활용하여 기존 해보다 좋은 해를 생성할 수 없는 결정 나무의 가지(branch)를 제외하여 결정 공간을 줄인다 (Martello and Toth 1990; Kellerer 외 2004). 본 항에서, 엘리스 문제의 정수 모형과 그의 상한이 되는 선형 계획을 제시한다. 알고리즘 1에서 개발한  $t_j$ -값의 변환을 활용하여 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  $j \in \mathcal{X}$ 과 동치다. 그러면 엘리스의 문제가 다음 비선형 정수 최적화 문제와 동등함을 볼 수 있다.

**문제 5** (엘리스의 문제를 위한 비선형 정수 계획).

$$\begin{aligned} & \text{maximize} \quad v(x) = \sum_{j=1}^m \left( f_j t_j x_j \prod_{i>j} (1 - f_i x_i) \right) \\ & \text{subject to} \quad \sum_{j=1}^m g_j x_j \leq H, \quad x_i \in \{0, 1\}^m \end{aligned} \tag{22}$$

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

**문제 6** (엘리스의 문제를 위한 선형 완화 문제).

$$\begin{aligned} & \text{maximize} \quad v_{\text{LP}}(x) = \sum_{j=1}^m f_j t_j x_j \\ & \text{subject to} \quad \sum_{j=1}^m g_j x_j \leq H, \quad x \in [0, 1]^m \end{aligned} \tag{23}$$

문제 6은(는) “연속 배낭 문제”라고 불리며 탐욕 알고리즘을 적용하면  $O(m \log m)$  시간에 쉽게 풀 수 있다 (Dantzig 1957).  $O(m)$  알고리즘도 알려져 있다 (Balas와 Zemel 1980, § 2).

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,  $\mathcal{I}$ 는  $\sum_{j \in \mathcal{I}} g_j \leq H$ 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. 첫 번째 부문제는 다음 같은 문제 5의 인스턴스다.

$$\begin{aligned} & \text{maximize} \quad v(x) = \gamma + \sum_{j \in \mathcal{N}} \left( f_j \bar{t}_j x_j \prod_{\substack{i \in \mathcal{N}: \\ i > j}} (1 - f_i x_i) \right) \\ & \text{subject to} \quad \sum_{j \in \mathcal{N}} g_j x_j \leq \bar{H}; \quad x_j \in \{0, 1\}, \quad j \in \mathcal{N}. \end{aligned} \tag{24}$$

두 번째 부문제는 대응하는 문제 6 인스턴스다.

$$\begin{aligned} \text{maximize } & w_{\text{LP}}(x) = \gamma + \sum_{j \in \mathcal{N}} f_j \bar{t}_j x_j \\ \text{subject to } & \sum_{j \in \mathcal{N}} g_j x_j \leq \bar{H}; \quad x_j \in [0, 1], \quad j \in \mathcal{N} \end{aligned} \quad (25)$$

각 부문제에서  $\bar{H} = H - \sum_{j \in \mathcal{I}} g_j$ 는 잔여 예산을 의미한다.  $\mathcal{I}$ 에 속한 학교에 (6)의 변환을 적용하여 모수  $\gamma$ 와  $\bar{t}$ 를 얻을 수 있다. 즉, 각  $j \in \mathcal{I}$ 에 대해  $\gamma$ 를  $f_j \bar{t}_j$  만큼 증가시키고, 시장에서  $j$ 를 소거하고, (6)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 고정하고 다른 새끼 마디에서  $x_j = 0$ 으로 고정한다. 일반적으로  $j$ 는 임의의 학교가 될 수 있지만 탐욕적인 휴리스틱으로  $f_j \bar{t}_j / g_j$ 의 비율이 가장 큰 학교를 선택한다. 이런 식으로 새끼를 생성하면 각 마디의  $\mathcal{I}$ -집합은 자기 어미 마디의  $\mathcal{I}$ -집합과 최대 한 학교로만 다르다. 따라서 새로운 마디를 생성할 때, (6)을(를) 한 번만 적용하면 상수  $\gamma$ 와 변화된  $\bar{t}_j$ -값을 얻을 수 있다.

이례적인 경우 2가지 있다. 첫 번째는,  $\mathcal{N}$ 에 속한 모든 학교에 대해  $g_j > \bar{H}$ 이면 가능성은 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. 두 번째는,  $\mathcal{N} = \emptyset$ 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 (6)에 적용하여  $\{\bar{t}_3, \bar{t}_4, \bar{t}_5\} = \{(1-f_2)t_3, (1-f_2)t_4, (1-f_2)t_5\} = \{30, 40, 50\}$ 과  $\bar{\gamma} = f_2 t_2 = 20$ 을 얻는다.  $c_5$ 를 소거하기 위해  $\bar{t}$ 에 (6)에 다시 적용하여  $\{\bar{t}_3, \bar{t}_4\} = \{\bar{t}_3 - f_5 \bar{t}_5, \bar{t}_4 - f_5 \bar{t}_5\} = \{5, 15\}$ 와  $\gamma = \bar{\gamma} + f_5 \bar{t}_5 = 35$ 를 얻는다. 마지막으로,  $\bar{H} = H - g_2 - g_5 = 3$ 이다. 이제 (24)과(와) (25)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 큰 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 (25)의 최댓값  $v_{\text{LP}}^*[n]$ 을 기록한다.  $\mathcal{I}$ 는 예산에 가능한 포트폴리오이므로  $v_{\mathcal{I}}[n]$ 은 원래 목적함수 최댓값의 하한이 된다. 또한 정리 3의 증명 과정에서 보였듯, (24)의 목적함수는 함수  $v(\mathcal{I} \cup \mathcal{X})$ 과 동치다. 이는  $v_{\text{LP}}^*[n]$ 은  $\mathcal{I}$ 를 부분집합으로 포함하며  $\mathcal{O}$ 에 속한 학교를 포함하지 않은, 즉 이 가지에 있는 모든 포트폴리오의 가치에 대한 상한임을 의미한다. 따라서 새끼 마디  $n_2$ 를 생성했을 때, 만약 그의 목적 함숫값  $v_{\mathcal{I}}[n_2]$ 이 어떤 다른 마디  $n_1$ 에 대응하는  $v_{\text{LP}}^*[n_1]$ 보다 크다면,  $n_1$ 과 그의 모든 후손을 무시할 수 있다.

$\mathfrak{T}$ 에 뿌리 마디  $n_0 = (\emptyset, \emptyset, \mathcal{C})$ 를 넣어서 알고리즘을 초기화한다. 각 반복 단계에서  $v_{\text{LP}}^*[n]$ -값이 가장 큰 마디  $n \in \mathfrak{T}$ 을 선택한다.  $n$ 의 새끼를 생성하고 후보 집합에서 제거한다. 그다음에  $n$ 의 각 새끼 마디  $n'$ 을 검증하고 나무에 추가한다. 새끼 마디 중에 새로운 최적해를 발견하면 이를 가장 좋은 후보로 표하고  $v_{\mathcal{I}}[n'] > v_{\text{LP}}^*[n'']$ 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.

---

**Algorithm 2:** 엘리스의 문제를 위한 분지한계법.

---

**Input:** 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 제거;
- 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 수정;
- 13       **end**
- 14       **if**  $v_{LP}^*[n] > L$ 이고  $n'$ 이 잎 마디 아님 **then**  $\mathfrak{T}$ 에  $n'$ 을 추가;
- 15       **end**
- 16       **for**  $n'' \in \mathfrak{T}$  **do**
- 17         **if**  $L > v_{LP}^*[n'']$  **then**  $\mathfrak{T}$ 에서  $n''$ 을 제거;
- 18       **end**
- 19     **end**
- 20 **end**

---

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 두 번 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 그중 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 제외할 수 있음을 의미하며, 그때 계산 노력의 대부분은 이미 쏟은 것이다. 수리 실험에서, 알고리즘 2은(는) 약  $m = 35$ 개의 학교를 넘는 인스턴스에 대해 비효율적이었으며 이는 단순한 열거법에 비해 큰 개선이 아니다. 하지만, 커버 부등식을 도입하여 해법을 개선할 가능성이 보인다 (Wolsey 1998, § 9.3).

### 4.3 의사 다행 시간 동적 계획

본 절에서 일반성을 약간 제한하여  $g_j \in \mathbb{N}$  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 이전 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 다향 시간 해법이라고 한다 (Garey and Johnson 1979, § 4.2). 그러나 지원 비용이 작은 정수가 되는 전형적인 대학 지원 문제 인스턴스에 대해 매우 효율적인 해법이다.

$j = 0 \dots m$ 와  $h = 0 \dots H$ 에 대해,  $\{1, \dots, j\}$ 에 속한 학교만 사용하면서 지원 지출액이  $h$ 을 넘지 않는 최적 포트폴리오를  $\mathcal{X}[j, h]$ 라고 하자. 또한  $V[j, h] = v(\mathcal{X}[j, h])$ 다.  $j = 0$  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  $\mathcal{X}[j, h] = \mathcal{X}[j-1, h]$ 다.  $\mathcal{X}[j, h]$ 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,  $\mathcal{X}[j, h] = \mathcal{X}[j-1, h-g_j] \cup \{j\}$ 다. 이 관찰에 따라  $j = 1 \dots m$ 와  $h = 1 \dots H$ 에 대해 다음 같은 Bellman 식을 얻는다.

$$V[j, h] = \max\{V[j-1, h], (1 - f_j)V[j-1, h-g_j] + f_j t_j\} \quad (26)$$

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

---

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

---

**Input:** 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  $V[j, h]$ 의 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**  $\mathcal{X}$

---

**정리 8** (알고리즘 3의 타당성). 알고리즘 3은(<는)  $O(Hm + m \log m)$  시간의 엘리스 문제의 최적 포트폴리오를 출력한다.

증명. 위 논의에 따라 최적성이 성립한다.  $t$ 를 배열하는 시간은  $O(m \log m)$ 이다. 병목 단계는 2줄에서  $V[j, h]$ 의 표를 만드는 것이다. 각 원소를 단위 시간에 생성할 수 있으며 표의 크기가  $O(Hm)$ 이다.  $\square$

#### 4.4 완전 다향 시간 근사 해법

엘리스의 문제는 배낭 문제와 같이, 가치가 가장 높은 포트폴리오의 비용 대신 비용이 가장 낮은 포트폴리오의 가치를 탐색하는 보완적인 동적 계획이 존재한다. 이를 기반으로  $O(m^3/\varepsilon)$  시간과 공간을 사용하는 엘리스 문제의 FPTAS를 도출할 수 있다.

포트폴리오의 근사적 가치를 정확도  $P$ 로 구성된 고정소수점 십진수(fixed-point decimal)로 나타내자. 다,  $P$ 는 소수점 뒤에 등장하는 숫자의 수다. 이때  $x$ 를 가장 가까운 고정소수점 십진수로 내림한 것을  $r[x] = 10^{-P} \lfloor 10^P x \rfloor$ 라고 하자. 임의의 포트폴리오의 가치가  $\bar{U} = \sum_{j \in C} f_j t_j$ 를 넘을 수 없으며, 모든 포트폴리오 가치의 고정소수점 십진수 근사가 실제값보다 작게 나타내도록 한다.

따라서 고정소수점 환경에서 발생할 수 있는 포트폴리오 가치로 이루어진 집합  $\mathcal{V}$ 는 유한 집합이다.

$$\mathcal{V} = \left\{ 0, 1 \times 10^{-P}, 2 \times 10^{-P}, \dots, r[\bar{U} - 1 \times 10^{-P}], r[\bar{U}] \right\} \quad (27)$$

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

본 항 나머지에서, 다른 언급이 없으면 포트폴리오의 “가치”라고 하면 고정소수점 환경 안에서의 가치를 의미하며 이것이 근삿값임을 유념한다. 근사 오차는 나중에 동적 계획의 타당성을 증명할 때 처리한다.

정수  $0 \leq j \leq m$  그리고  $v \in [-\infty, 0) \cup \mathcal{V}$ 에 대해,  $\{1, \dots, j\}$ 에 속한 학교만 사용하면서 최소한  $v$ 의 가치를 가지는 포트폴리오가 존재한다면 그중 가장 비용이 낮은 포트폴리오를  $\mathcal{W}[j, v]$ 라고 하자. 그의 비용을  $G[j, v] = \sum_{j \in \mathcal{W}[j, v]} g_j$ 라고 하자. 단,  $\mathcal{W}[j, v]$ 가 존재하지 않으면  $G[j, v] = \infty$ 으로 처리한다.  $v \leq 0$ 이면  $\mathcal{W}[j, v] = \emptyset$ 이고  $G[j, h] = 0$ 이며,  $j = 0$ 이고  $v > 0$ 이면  $G[j, h] = \infty$ 인 것이 자명하다. 남은 지표(즉  $j, v > 0$ )는 다음을 만족한다.

$$G[j, v] = \begin{cases} \infty, & t_j < v \\ \min\{G[j-1, v], g_j + G[j-1, v - \Delta_j(v)]\}, & t_j \geq v \end{cases} \quad (28)$$

$$\text{where } \Delta_j(v) = \begin{cases} r \left[ \frac{f_j}{1-f_j} (t_j - v) \right], & f_j < 1 \\ \infty, & f_j = 1 \end{cases} \quad (29)$$

$t_j < v$ 인 경우, 가능한 포트폴리오는 효용이  $v$ 보다 작은 학교로 구성되므로 그의 가치는  $v$ 를 넘을 수가 없고  $\mathcal{W}[j, v]$ 가 정의되지 않는다.  $t_j \geq v$ 인 경우,  $\min\{\}$ 의 첫 입력값은  $j$ 를 제외하고  $\mathcal{W}[j-1, v]$ 로 지원하는 것이 가능한 선택임을 의미한다. 반면에  $j \in \mathcal{W}[j, v]$ 라면 다음이 성립한다.

$$v(\mathcal{W}[j, v]) = (1 - f_j)v(\mathcal{W}[j, v] \setminus \{j\}) + f_j t_j \quad (30)$$

이는 부분 포트폴리오  $\mathcal{W}[j, v] \setminus \{j\}$ 가 최소한  $v - \Delta$ 의 가치를 가져야 한다고 의미한다. 단,  $\Delta$ 는  $v = (1 - f_j)(v - \Delta) + f_j t_j$ 를 만족한다.  $f_j < 1$ 일 때, 이 식의 해는  $\Delta = \frac{f_j}{1-f_j}(t_j - v)$ 다. 이값을 내림함으로써  $\mathcal{W}[j, v]$ 가 ‘최소한’  $v - \Delta$ 임을 보장한다.  $t_j \geq v$ 이고  $f_j = 1$ 일 때, 단일 원소 집합  $\{j\}$ 에 대해  $v(\{j\}) \geq v$ 이므로

$$G[j, v] = \min\{G[j-1, v], g_j\}. \quad (31)$$

이때  $\Delta_j(v) = \infty$ 으로 정의하면  $g_j + G[j-1, v - \Delta_j(v)] = g_j + G[j-1, v - \infty] = g_j$ 와 같은 적절한 값이 나온다.

각 지표에 대해  $G[j, v]$ 를 계산한 다음에,  $\mathcal{W}[j, v]$ 가  $j$ 를 포함할 필요충분 조건인  $G[j, v] < G[j-1, v]$ 를 적용하면 해당 최적 포트폴리오를 구할 수 있다. 그러면 가능하면서 최대한 목적 함숫값  $\max\{w : G[m, w] \leq H\}$ 와 해당 포트폴리오를 계산하면 엘리스 문제의 근사해를 얻을 수 있다.

**정리 9** (알고리즘 4의 타당성). 알고리즘 4은(는)  $O(m^3/\varepsilon)$  시간에 엘리스의 문제를 위한  $(1 - \varepsilon)$ -근사 최적 포트폴리오를 출력한다.

증명. (최적성.) 알고리즘 4의 출력이  $\mathcal{W}$ 이며 실제 최적해가  $\mathcal{X}$ 라고 하자.  $v(\mathcal{X}) \leq \bar{U}$ 이고, 모든 단일 원소 포트폴리오가 가능해이므로  $\mathcal{X}$ 는 평균적인 단일 원서 포트폴리오보다 가치 높을 수밖에 없다. 즉,  $v(\mathcal{X}) \geq \bar{U}/m$ 다.

(28)과(와) (29)(으)로 정의된 점화식에 따라  $\Delta_j(v)$ 를 내림하기 때문에,  $j \in \mathcal{W}[j, v]$ 이면  $(1 - f_j)v(\mathcal{W}[j-1, v - \Delta_j(v)]) + f_j t_j$ 의 실제값은  $\mathcal{W}[j, v]$ 의 고정소수점 가치인  $v$ 보다 클 수 있는데 최대한  $10^{-P}$  만큼의 차이가 발생한다. 또한 이 오차는  $\mathcal{W}$ 에 학교를 추가할 때마다 가산적으로 누적되는 데 추가하는 학교의 개수는 최대한  $m$ 이다. 따라서 알고리즘의 4줄에서 기록하는  $\mathcal{W}$ 의 고정소수점 가치가  $v'(\mathcal{W})$ 일 때,  $v(\mathcal{W}) - v'(\mathcal{W}) \leq m10^{-P}$ 이 성립한다.

---

**Algorithm 4:** 엘리스의 문제를 위한 완전 다향 시간 근사 해법.

---

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

**Parameters:** 허용치  $\varepsilon \in (0, 1)$ .

- 1  $t$ 의 순서대로 학교를 배열한다;
- 2 정확도  $P \leftarrow \lceil \log_{10}(m^2/\varepsilon\bar{U}) \rceil$ ;
- 3  $G[j, h]$ 의 값으로 표를 채운다;
- 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**  $\mathcal{X}$

---

$\mathcal{X}$ 의 원소를 지표 순서대로 추가했을 때, 각 학교를 추가하면 (30)에 따라 가치를 수정하고 가장 가까운  $10^{-P}$ 의 배수로 내림한 값으로  $v'(\mathcal{X})$ 를  $v'(\mathcal{W})$ 와 유사하게 정의할 수 있다. 위와 같은 논리에 따라  $v(\mathcal{X}) - v'(\mathcal{X}) \leq m10^{-P}$ 이다. 또한  $\mathcal{W}$ 가 고정소수점 환경에서 최적이므로  $v'(\mathcal{W}) \geq v'(\mathcal{X})$ 임을 알 수 있다.

위 관찰을 적용하면 다음 부등식을 도출할 수 있다.

$$v(\mathcal{W}) \geq v'(\mathcal{W}) \geq v'(\mathcal{X}) \geq v(\mathcal{X}) - m10^{-P} \geq \left(1 - \frac{m^2 10^{-P}}{\bar{U}}\right) v(\mathcal{X}) \geq (1 - \varepsilon) v(\mathcal{X}) \quad (32)$$

이는 정리에서 말한 근사 계수다.

(계산 시간.) 3줄에서 표를 채우는 것이 병목 단계이며 표의 크기는  $m \times |\mathcal{V}|$ 다. 그러면

$$|\mathcal{V}| = \bar{U} \times 10^P + 1 = \bar{U} \times 10^{\lceil \log_{10}(m^2/\varepsilon\bar{U}) \rceil} + 1 \leq \frac{m^2}{\varepsilon} \times \text{상수} \quad (33)$$

는  $O(m^2/\varepsilon)$ 이므로 시간 복잡성은 정리와 같다.  $\square$

계산 시간이  $m$ 과  $1/\varepsilon$ 의 다향식으로 제한되므로 알고리즘 4은(는) 엘리스의 문제를 위한 FPTAS다 (Vazirani 2001).

알고리즘 3과(와) 알고리즘 4을(를) 표 대신 재귀 함수 기반으로 구성할 수 있다. 그러나 각 재귀 함수가 스스로를 두 번 호출하므로 지수적으로 늘어나는 호출 수를 방지하기 위해 각 지표에 대응하는 합수값을 표에 저장하거나 다른 식으로 기록해야 한다.

## 4.5 모의 담금질 휴리스틱

모의 담금질은 비볼록 최적화 문제를 위한 인기 많은 지역 탐색 휴리스틱이다. 선천적인 확률성 때문에 모의 담금질 해법에는 보장된 정확도가 없다. 반면에 구현하기 쉽고 계산 비용이 적으면서 최적에 가까운 해를 구하는 경향이 있다. 본 항에서, 엘리스의 문제를 위한 모의 담금질 해법을 제시한다. Wolsey (1998, § 12.3.2)가 제시한 정수 계획 모의 담금질 해법이 바탕이 된다.

초기 해를 선정하기 위해, 예산이 고갈될 때까지 예산이 고갈될 때까지  $f_j t_j / g_j$ -값이 감소하는 순서대로 학교를 추가하는 탐욕 휴리스틱을 도입한다. 예 2에서 보였듯이 탐욕 해의 근사 계수는 무한히 작아질 수 있지만, 전형적인 인스턴스에서는 좋은 초기 해다. 그다음으로, 후보의 가능해  $\mathcal{X}$

의 이웃  $\mathcal{X}_n$ 을 생성하기 위해  $\mathcal{X}$ 를 복사하고 불가능해질 때까지 학교를 추가하고, 가능성이 복구될 때까지  $\mathcal{X}$ 의 원소를  $\mathcal{X}_n$ 에서 제거한다. 만약  $v(\mathcal{X}_n) \geq v(\mathcal{X})$ 이면, 후보를 이웃으로 대체한다. 아니면,  $\exp(v(\mathcal{X}_n) - v(\mathcal{X})/T)$ 의 확률로 대체하며, 여기서  $T$ 는 “온도 모수”라고 불린다. 이를  $N$ 번 반복한 다음에 가장 좋은  $\mathcal{X}$ 를 출력한다.

---

**Algorithm 5:** 엘리스의 문제를 위한 모의 담금질.

---

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

**Parameters:** 초기 온도  $T \geq 0$ , 온도 절감 모수  $r \in (0, 1]$ , 반복 단계 개수  $N$ .

- 1 예산이 고갈될 때까지 학교를  $f_j t_j / g_j$ 가 감소하는 순서대로  $\mathcal{X}$ 에 추가;
- 2 **for**  $i = 1 \dots N$  **do**
- 3      $\mathcal{X}_n \leftarrow \text{copy}(\mathcal{X})$ ;
- 4      $\mathcal{X}_n$ 이 불가능해질 때까지  $\mathcal{C} \setminus \mathcal{X}$ 에서 학교를 무작위로 추가;
- 5      $\mathcal{X}_n$ 의 가능성이 복구될 때까지  $\mathcal{X}$ 에 속한 학교를  $\mathcal{X}_n$ 에서 제거;
- 6      $\Delta = v(\mathcal{X}_n) - v(\mathcal{X})$ ;
- 7     **if**  $\Delta \geq 0$  **then**  $\mathcal{X} \leftarrow \mathcal{X}_n$  **else**  $\exp(\Delta/T)$ 의 확률로  $\mathcal{X} \leftarrow \mathcal{X}_n$ ;
- 8      $T \leftarrow rT$ ;
- 9 **end**
- 10 **return** 가장 좋은  $\mathcal{X}$

---

제의 2. 알고리즘 5의 계산 시간은  $O(Nm)$ 이다.

계산 실험 결과에 따라, 알고리즘 5의 출력은 보통 최적에 아주 가깝다.

## 5 계산 실험

본 절에서 위에서 도출한 알고리즘의 현실성을 탐구하고자 계산 실험의 결과를 제시한다.

### 5.1 알고리즘의 구현

모든 알고리즘을 줄리아 (Julia, v1.8.0b1) 언어로 구현했다. 줄리아는  $t_j$ -값이 정수 또는 부동소수점 수와 같은 특수한 경우에 대해 컴파일러가 자동으로 최적화하는 모수적 자료형(parametric data type) 기능이 있어서 유용한 선택이었다. 또한 병렬 연산을 쉽게 관리할 수 있는 매크로를 제공하므로 계산 실험의 규모를 증가할 수 있었다 (Bezanson 외 2017). DataStructures.jl 패키지(v0.18.11, [github.com/JuliaCollections](https://github.com/JuliaCollections))도 활용했다. 실험 코드는 [github.com/maxkapur/OptimalApplication](https://github.com/maxkapur/OptimalApplication)에서 공유한다.

동적 계획 해법인 3과(와) 알고리즘 4을(를) 재귀 함수와 사전 기록으로 구현했다 (Cormen 외 1990, § 16.6). 알고리즘 4은(는) 4.4항에서 설명한 것과 달리, 포트폴리오의 가치를 십진수 대신 이진수로 나타냈으며  $P$ 와  $\mathcal{V}$ 의 정의를 적절히 수정했다. 그리고 고정소수점 대신  $\mathcal{V}$ 의 모든 원소를  $2^P$ 으로 곱하여 정수로 변환했다. 이런 변경은 기본 알고리즘 설계와 계산 시간 분석에 영향을 주지 않는데 성능이 많이 개선된다.

### 5.2 실험 방법

총 3개의 수리 실험을 진행했다. 실험 1과 실험 2에서, 각각 알마의 문제와 엘리스의 문제를 위한 해법의 비교적 계산 시간을 판단하고자 한다. 실험 3에서, 정확한 해법에 비한 모의 담금질 휴리스틱의 현실적 정확도를 탐구한다.

가산 시장을 생성하기 위해  $t_j$ -값을 평균이 10인 지수 분포에서 독립적으로 관측하고 정수로 올림했다. 그다음에,  $t_j$ 와  $f_j$ 가 서로 부분적으로 반비례하게 만들도록  $f_j = 1/(t_j + 10Q)$ 로 정의했다. 단,  $Q$ 는  $[0, 1]$ 에서 균일한 확률로 관측한다. 알마의 문제를 고려하는 실험 1에서,  $h = \lfloor m/2 \rfloor$ 으로 처리했다. 엘리스의 문제를 고려하는 실험 2와 실험 3에서, 모든  $g_j$ 를  $\{5, \dots, 10\}$ 에서 균일한 확률로 관측했으며  $H = \lfloor \frac{1}{2} \sum g_j \rfloor$ 으로 처리했다. 실제 대학 지원 비용인 50과 100달러 사이에 10 달러의 어떤 배수이므로 합리적인 비용 분포다. 그럼 2에서 전형적인 인스턴스가 나타난다.

실험 1과 실험 2의 실험 변수로는 시장의 크기  $m$ , 해법 선택, 그리고 (알고리즘 4인 경우) 허용치  $\varepsilon$ 을 고려했다. 실험 변수의 각 조합에 대해, 50개의 시장을 생성했으며 최적 포트폴리오를 3번 계산한 것 중 가장 빠른 것으로 계산 시간으로 기록했다. 50개의 시장에 대한 평균과 표준편차를 표에 나타낸다. 따라서 아래 표의 각 칸은 150개의 계산에 대한 통계다.  $t$ 를 배열하는 시간은 고려하지 않았다. 그리고 모의 담금질 휴리스틱의 계산 시간이 무시해도 될 정도이므로 실험 2에서 제외했다.

실험 3에서, 500개의 시장을 생성했으며, 크기는 대수가 균일하게 분포되도록  $m = 2^3$ 과  $2^{11}$  사이에서 무작위로 선택했다. 각 시장에 알고리즘 5(을)를 적용하여 휴리스틱한 해를 구하고 알고리즘 3(을)를 적용하여 정확한 해를 구한 다음의 최적성 비율을 계산했다. 모의 담금질의 반복 단계 개수는  $N = 500$ 이며, 격자 탐색을 이용한 예비 실험 결과에 따라 온도 모수를  $T = 1/4$  그리고  $r = 1/16$ 로 설정했다.

### 5.3 결과 정리

실험 1에서, 동일한 지원 비용으로 정의된 크기가 다양한 시장에 대해 알고리즘 1의 성능을 고려했다. 이를  $\mathcal{C}$ 를 배열 혹은  $f_j t_j$ -값에 따라 배열된 힙 구현으로 구성하는 경우로 나눴으며 실험 결과는 표 2에서 등장한다. 실험 결과에 따라 배열 구현이 더 빠른 해법이었다.

실험 2에서 일반적인 문제를 고려한다. 정확한 해법 (알고리즘 2 및 알고리즘 3) 그리고 허용치를 0.5와 0.05로 설정한 근사 해법 (알고리즘 4)의 성능을 비교한다. 실험 결과는 표 2에서 나타나며 위에서 도출한 시간 복잡성 결과와 대략 같다. 정확한 동적계획 해법이 전책적으로 가장 빠르며 FPTAS가 약간 느린 것으로 나온 결과는 배낭 문제 해법을 비교한 기존 연구와 비슷하다 (Martello 와 Toth 1990, § 2.10). 분지한계법은 크기가 중간인 인스턴스에도 비효율적이었다. 알고리즘 3의 좋은 실험적 성능은 부분적으로 가상 인스턴스의 구조 덕이다. 가상 인스턴스의 지원 비용이 작은 정수이고  $H$ 가  $m$ 과 선형식으로 비례하는 것은 알고리즘의 기대 계산 시간이  $O(m^2)$ 임을 의미한다. 그러나 전형적인 학생의 지원 예산은 최대한 열 몇 개의 학교에 지원할 수 있게 하며  $m$ 에 대해 상수이므로, 실제 대학 지원 문제 인스턴스에 적용하면 알고리즘 3이 더욱 뚜렷한 상대적 우위를 발휘할 가능성이 있다.

실험 3은 가상 인스턴스를 통해 모의 담금질 휴리스틱의 현실 성능을 탐구했으며, 실험 결과는 그림 3에 도시하였다. 모든 인스턴스에서 최적의 10% 이내에 해를 구했으며, 대다수의 경우는 2% 이내로 달했다. 휴리스틱의 성능은  $m \leq 16$ 이 되는 작은 시장에서 가장 낮았으며, 정확한 해법도 효율적으로 풀 수 있는 시장 크기이므로 염려할 이유가 없어 보인다.

학교의 개수 $m$	알고리즘 1 (배열 구현)	알고리즘 1 (힙 구현)
16	0.00 (0.00)	0.01 (0.00)
64	0.01 (0.00)	0.08 (0.02)
256	0.06 (0.00)	0.96 (0.26)
1024	0.82 (0.03)	14.27 (2.28)
4096	12.87 (0.71)	230.77 (18.75)
16384	199.48 (1.77)	3999.19 (283.48)

표 2: (실험 1.) 단위: ms. 알고리즘 1에서  $\mathcal{C}$ 를 배열 또는 힙 구현으로 저장했을 때, 동일 지원 비용 입학 시장의 최적 지원 포트폴리오를 계산하는 시간. 각  $m$ 에 대해 50개의 시장을 생성했으며 알고리즘을 3번 반복하여 그중 최소 계산 시간을 기록했다. 표에서 50개의 인스턴스에 대한 평균 (표준편차) 시간이 나타난다. 모든 경우,  $h = m/2$ .

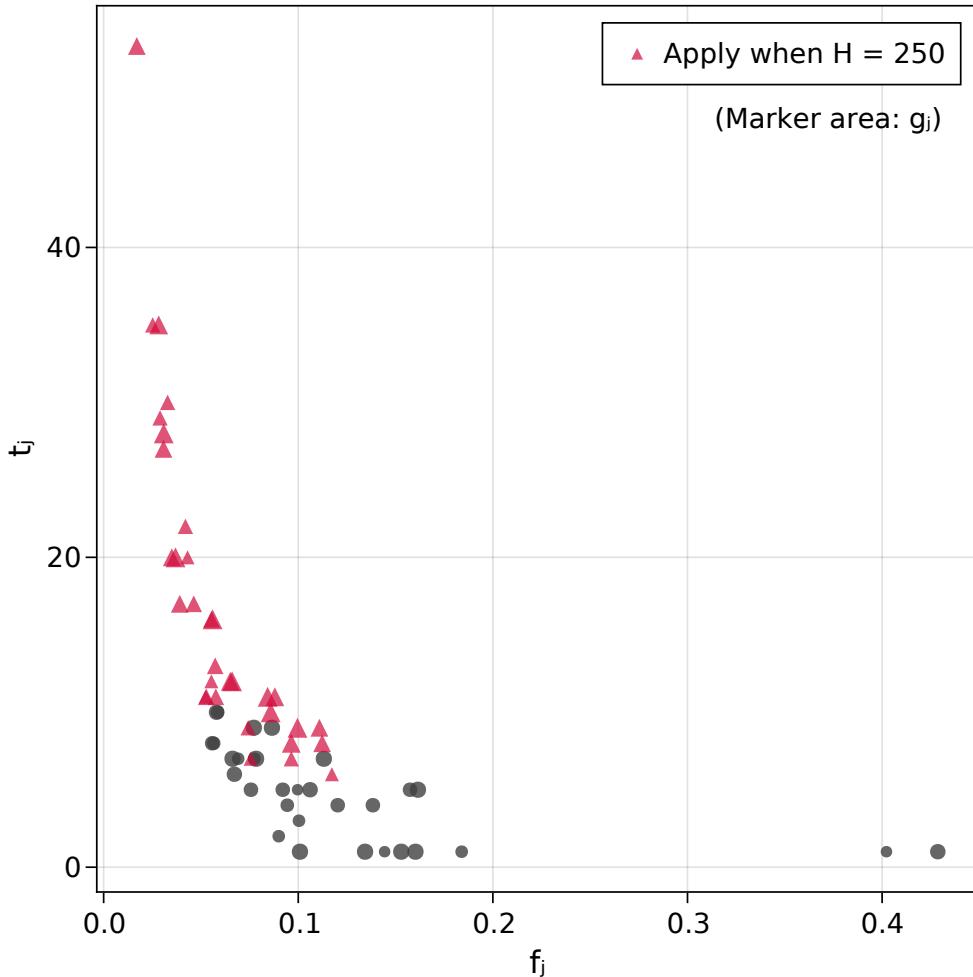


그림 2:  $m = 64$ 개의 학교로 구성된 전형적인 무작위로 생성한 인스턴스와 해당 최적 포트폴리오. 지원 비용  $g_j$ 는  $\{5, \dots, 10\}$ 에서 균일한 확률로 선정했으며 기호의 넓이와 비례된다. 최적 포트폴리오는 알고리즘 3(으)로 계산했다.

학교의 개수 $m$	알고리즘 2: 분지한계법	알고리즘 3: 지출액 동적 계획	알고리즘 4: FPTAS, $\varepsilon = 0.5$	알고리즘 4: FPTAS, $\varepsilon = 0.05$
8	0.02 (0.01)	0.01 (0.00)	0.05 (0.01)	0.16 (0.04)
16	0.10 (0.04)	0.07 (0.02)	0.39 (0.10)	2.59 (0.62)
32	12.43 (9.88)	0.29 (0.05)	2.15 (0.29)	33.07 (10.72)
64	— (—)	1.24 (0.16)	13.24 (2.22)	339.64 (100.38)
128	— (—)	6.20 (0.64)	79.92 (20.18)	2042.63 (749.71)
256	— (—)	30.63 (2.28)	818.50 (632.98)	18949.50 (3533.88)

표 3: (실험 2.) 단위: ms. 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 50개의 시장을 생성했으며 알고리즘을 3번 반복하여 그중 최소 계산 시간을 기록했다. 표에서 50개의 인스턴스에 대한 평균 (표준편차) 시간이 나타난다.

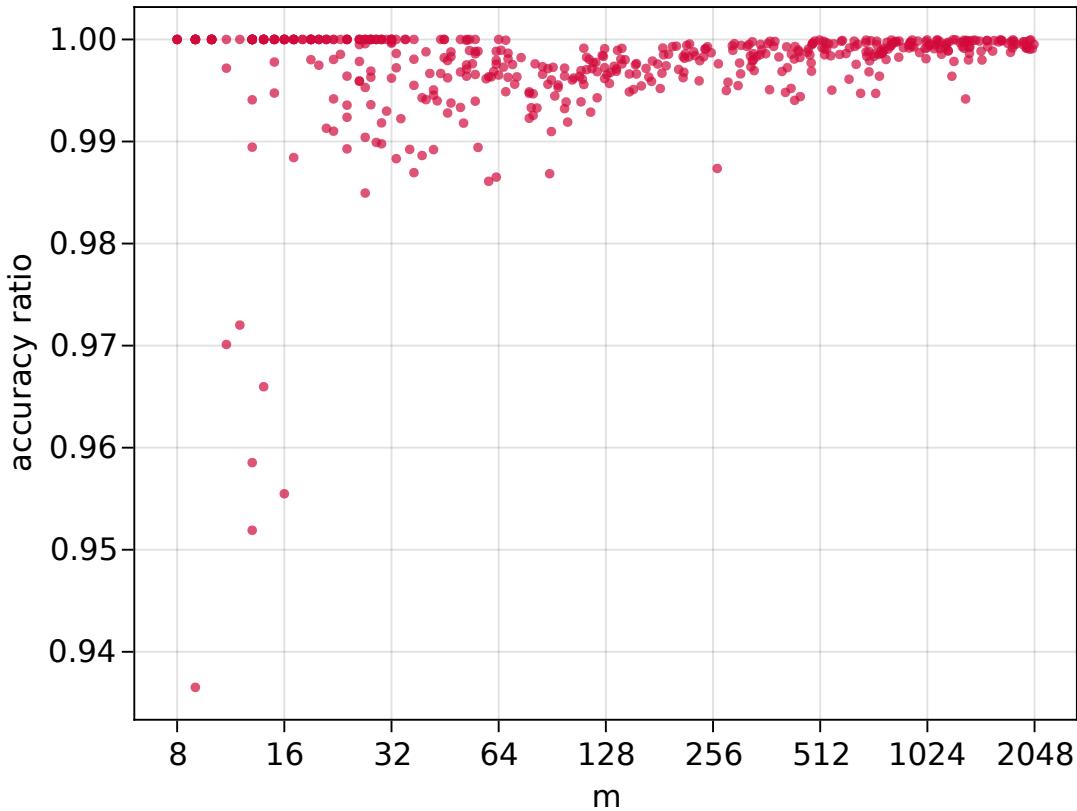


그림 3: (실험 3.) 크기가 다양한 시장에 적용한 모의 담금질 (알고리즘 5) 해법이 달성하는 최적성 비율. 모의 담금질의 모수는  $N = 500, T = 1/4, r = 1/16$ 이며, 최적 포트폴리오는 알고리즘 3(으)로 계산했다.

## 6 결론과 향후 연구

본 연구는 새로운 조합 최적화 문제를 제안했으며 이를 대학 입학 지원 최적화 문제라고 부른다. Submodular한 목적함수를 가지는 자산 배분 혹은 배낭 문제로 볼 수 있다. 모든 대학의 지원 비용이 동일한 특수한 경우, 탐욕 알고리즘으로 다행 시간에 풀 수 있음을 보였으며 최적해가 예산 제약식에 대한 포함 사슬 관계를 가지기 때문이다. 일반적인 문제는 NP-complete하다. 4가지 해법을 제시했으며 이론적으로 가장 효율적인 알고리즘은  $O(m^3/\varepsilon)$  시간에  $(1 - \varepsilon)$ -근사해를 계산하는 FPTAS다. 그 반면에 전형적인 대학 지원 문제 인스턴스에 대해, 지원 비용 지출액 기반 동적 계획 해법은 실행하기 쉬울뿐더러 계산 시간이 상당히 짧다. 계산 실험에서 모의 담금질 기반 휴리스틱 해법도 좋은 성능을 발휘했다. 표 4에서 본 논문이 다운 알고리즘을 정리한다.

이 문제를 확장할 수 있는 향후 연구 방향 3가지를 제안할 수 있다.

### 6.1 위험 회피를 모수화한 모형

익숙한 Markowitz 포트폴리오 최적화 모형에서는 명시적인 위험 회피 항이 등장하며, 본 모형은 maximax 목적함수에 내포된 경쟁적인 학교와 안정 지원 학교 사이의 균형에 의한 위험 관리 요소만 고려했다. 그러나 기본 목적함수인  $v(\mathcal{X}) = E[X]$ 에 표준편차 폐널티  $\beta \geq 0$ 를 다음처럼 도입할 수

알고리즘	논문 내 위치	문제	제한	정확도	계산 시간
나이브	정의 3	동일한 지원 비용	없음	( $1/h$ )-근사	$O(m)$
탐욕 해법	알고리즘 1	동일한 지원 비용	없음	정확	$O(hm)$
분지한계법	알고리즘 2	일반 문제	없음	정확	$O(2^m)$
지출액 동적 계획	알고리즘 3	일반 문제	$g_j$ 정수	정확	$O(Hm + m \log m)$
FPTAS	알고리즘 4	일반 문제	없음	( $1 - \varepsilon$ )-근사	$O(m^3/\varepsilon)$
모의 담금질	알고리즘 5	일반 문제	없음	0-근사	$O(Nm)$

표 4: 본 논문에서 다룬 해법의 요약.

있다:

$$\begin{aligned}
 v_\beta(\mathcal{X}) &= E[X] - \beta \text{Var}(X) \\
 &= E[X] - \beta (E[X^2] - E[X]^2) \\
 &= \sum_{j \in \{0\} \cup \mathcal{X}} \left( f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right) - \beta \sum_{j \in \{0\} \cup \mathcal{X}} \left( f_j t_j^2 \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right) + \beta v(\mathcal{X})^2 \quad (34) \\
 &= v(\mathcal{X}; \tau) + \beta v(\mathcal{X}; t)^2
 \end{aligned}$$

단,  $\tau_j = t_j - \beta t_j^2$ . 이의 첫 번째 항 그 자체가 포트폴리오 가치 함수이며, 두 번째 항은 포트폴리오 가치 함수의 단조 변환이다. 따라서 위에서 제시한 알고리즘을 서브루틴으로 사용하면 효율적 투자선 (efficient frontier)을 탐색할 수 있어 보인다. 이때 다양한  $\alpha \geq 0$ 에 대해 예산 제약식과  $v(\mathcal{X}; \tau) \geq \alpha$  아래서  $v(\mathcal{X}; t)$ 을 최대화하는 것이 알고리즘의 요점이다.

위험 회피를 모수화한 모형은 실제 지원 행동을 더 정확하게 반영할 수 있다. “상향·소신·안정” 지원 학교를 대략 균일하게 분산하여 지원하는 것이 최적이란 통념이다 (전민희 2015). 입학 시장의 크기에 비해 지원 예산이 작은 경우, 본 연구에서 제시하는 알고리즘은 비슷한 전략을 추천한다 (항 3.5의 예제 참고). 그러나 포트폴리오에 이미 삽입한 학교를 반영한 한계 가치를 나타내도록 효용 모수를 할인하는 수식 (6)을 살펴보면, 효용이 높은 학교보다 효용이 낮은 학교에 부과되는 폐널티가 큰 것을 알 수 있다. 그 결과는 지원 예산이 커질수록 본 모형의 최적 포트폴리오가 안정 지원 학교보다 상향 지원 학교를 편드는 것이다. (그림 2에서 뚜렷이 보이는 현상이다.)

학생의 지원 행동( $\mathcal{X}$ )을 관찰하고 학생이 인식하는 대학교의 질( $t_j$ )을 추정하는 경제 측정학 연구에서 본 논문의 모형을 그대로 응용할 수 있다. 실무 연구 결과에 따르면 위험 회피의 다양성은 인간 의사 결정 다양성의 주된 원인이 되며, 교육 선택과 취업 활동은 특히 그렇다 (Kahneman 2011; Hartlaub와 Schneider 2012; van Huijsen과 Alessie 2019). 따라서 경제 측정학 맥락에서, 위험 회피 모수를 내생 변수로 도입하면 원래 모형의 묘사력을 많이 개선할 수 있다.

## 6.2 시그널 전략과 matroid 제약 조건

향후 연구에 두 번째 방향은 모형에 시그널 전략을 도입하는 것이다. 한국 정시 입학 과정의 온라인 지원 양식에는 가군, 나군, 다군이란 3가지 선다형 칸을 채워서 지원하는 학교를 선택한다. 대부분의 학교는 모든 3개의 칸에 등장하지 않고 그중 1~2개만 가능하다. 따라서 학생이 지원할 수 있는 학교의 ‘수’일 뿐만 아니라 가능한 지원 포트폴리오에 같이 들어갈 수 있는 학교의 ‘조합’도 제한된다. 이는 학생이 상위권 대학만 지원하지 않도록 만들어지기 때문에 다각화(diversification) 제약식으로

볼 수 있다. 그러나 대학이 어떤 군에 속하는 것은 대학 순위일뿐더러 입학 전형 유형도 의미한다. 가군과 나군은 관례적으로 학생이 선호하는 대학을 선택하는 것이며 다군보다 수능 점수의 반영 비율이 높은 편이다. 또한 군 2개에 포함되는 학교도 있으며 입학 전형이 다르면서 평행한 모집 과정 2개를 운영한다. 가령, 어떤 학생이 면접 능력에 비교 우위가 있으면 경쟁적인 학교에 다군으로 지원하면 합격 확률을 높일 수도 있다. 이런 맥락에서 최적의 지원 전략은 미묘한 문제이며 한국 SNS에서 자주 발생하는 화젯거리다.

미국 입학 과정의 유사한 특징은 조기 전형(early decision)이라고 불린다. 이는 학생이 합격하면 진학한다고 약속하고 지원하는 것을 의미한다. 조기 전형으로 지원하면 입학할 관심을 시그널할 수 있으므로 합격 확률을 높일 수 있지만, 학교가 학생을 모집하려고 장학금을 수여할 동기가 약화할 수 있으므로 학생의 효용 추정이 달라진다.

대학 지원 문제의 정수 모형(문제 5)에서 위의 시그널 전략들을 어렵지 않게 도입하는 방식은 각  $c_j$ 를  $c_{j+}$ 와  $c_{j-}$ 로 나누는 것이다. 그다음에 학생이  $c_j$ 에 높은 관심을 시그널하면 (즉, 가군으로 선택하거나 조기 전형으로 지원하면)  $x_{j+} = 1$ 이며 관심을 시그널 하지 않고 지원하면 (즉 가군 혹은 나군으로 지원하거나 조기 전형 없이 지원하면)  $x_{j-} = 1$ 이 되는 이진 결정 변수를 정의한다. 해당 합격 확률  $f_{j+}$ 와  $f_{j-}$  그리고 효용  $t_{j+}$ 와  $t_{j-}$ 가 알려져 있으면, 다음 같은 논리 제약식을 더하면 모형을 완성할 수 있다.

$$x_{j+} + x_{j-} \leq 1, \quad j = 1 \dots m \quad \text{그리고} \quad \sum_{j=1}^m x_{j+} \leq 1 \quad (35)$$

모두 배낭 제약식이므로 Kulik 외 (2013)의 submodular 최대화 알고리즘을 통해 이 문제의  $(1 - 1/e - \varepsilon)$ -근사해를 구할 수 있다. 또한 (알마의 문제처럼) 예산 제약식이 집합 크기 제약식이 되면, 논리 제약식을 더한 모형의 해집합이 matroid가 되며, Calinescu 외 (2011)의 알고리즘이 대안 해법이 된다. 단, 그의 근사 계수는 여전히  $1 - 1/e - \varepsilon$ 이다.

본 연구의 주된 결과는 대학 지원 문제의 목적함수가 선형 함수가 아니지만, 최적화 관점에서 선형 함수와 비슷한 성질을 가진다는 것으로 해석할 수 있다. 집합 크기 제약 아래서 탐욕 해법이 정확하고, 배낭 제약 아래서 FPTAS가 존재하기 때문이다. 그런데 선형 함수는 matroid 제약 아래서 최적화하는 문제도 탐욕 해법으로 풀 수 있다 (Edmonds 1971). 따라서 대학 지원 목적함수와 선형 함수의 유사성이 적당히 깊으면 위에서 논한 matroid 제약 아래서 이를 최적화하는 문제를 탐욕 해법으로 풀 수 있을 것이다. 예비 수리 실험을 진행한 결과로 이 성질이 사실처럼 보이지만, matroid가 아주 광범위한 클래스이므로 철저한 실험이 어렵다. ( $\S.2.2$ 에서 제시된 변수 기법의 증명 과정에서  $|\mathcal{X}| = |\mathcal{Y}|$ 일 때  $\mathcal{X} \cup \{k\}$ 가 가능해면  $\mathcal{Y} \cup \{k\}$ 도 가능해라고 가정하기 때문에 포함 사슬 관계 성질을 matroid에 자동으로 확장할 수 있는 것이 아니다.)

### 6.3 동적 계획의 메모리 소요 절감

수리 실험 결과에 따르면 동적 계획 해법 성능의 병목 요소는 계산 시간이 아니라 메모리 소모량이다. 이 알고리즘의 메모리 소요를 절감하면 상당히 큰 문제를 풀 수 있다.

추상적인 관점에서, 알고리즘 3과(와) 알고리즘 4은(는) 최적해를  $Z[N, C]$ 로 표현하는 2차원 동적 계획이다. 이때  $N$ 은 결정 변수의 수,  $C$ 는 제약식의 모수, 그리고  $Z[n, c]$ 는 제약 모수가  $c \leq C$  일 때 첫  $n \leq N$ 개의 결정 변수만 이용하는 최적 목적 함수값이다. 알고리즘의 구조는  $Z[n, c]$ 를  $Z[n-1, c]$ 와 어떤  $c' \leq c$ 에 대응하는  $Z[n-1, c']$ 의 함수로 표현하고 그 반복 관계로 탐색하는 것이다. (알고리즘 3에서  $Z$ 는 최대 포트폴리오 가치,  $N = m$ , 그리고  $C = H$ 이며, 알고리즘 4에서  $Z$ 는 최소한 지원 지출액,  $N = m$ , 그리고  $C = |\mathcal{V}| \propto m^2/\varepsilon$ 이다.)

이러한 형태의 동적 계획을 표 혹은 사전 기록으로 구현하면 최적해를 출력하는 것은  $O(NC)$  시간과  $O(NC)$  공간을 소요한다. Kellerer 외 (2004, §3.3)는 동적 계획  $Z$ 를  $O(NC)$  시간과  $O(N+C)$  공간 안에 최적해를 구하는 분할 정복(divide and conquer) 알고리즘으로 변환하는 일반적인 방법을 제시하며 매우 유익하다. 그러나 이를 이용할 수 있는 조건 중, 목적함수는 어떤 기술적인 의미에서

가산적으로 분할되어야 하며 이를 대학 지원 문제에 적용하기 어렵다.

## A 부록

### A.1 정리 5의 기초적 증명

*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,

$$\begin{aligned}
 2v(\mathcal{X}_h) &= v(\mathcal{X}_{h-1} \cup \{j\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
 &\geq v(\mathcal{X}_{h-1} \cup \{k\}) + v(\mathcal{X}_{h-1} \cup \{j\}) \\
 &= v(\mathcal{X}_{h-1} \cup \{k\}) + (1 - f_j)v(\mathcal{X}_{h-1}) + f_j E[\max\{t_j, X_{h-1}\}] \\
 &= v(\mathcal{X}_{h-1} \cup \{k\}) - f_j v(\mathcal{X}_{h-1}) + f_j E[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\
 &\geq v(\mathcal{X}_{h-1} \cup \{k\}) - f_j v(\mathcal{X}_{h-1} \cup \{k\}) + f_j E[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\
 &= (1 - f_j)v(\mathcal{X}_{h-1} \cup \{k\}) + f_j E[\max\{t_j, X_{h-1}\}] + v(\mathcal{X}_{h-1}) \\
 &= v(\mathcal{X}_{h-1} \cup \{j, k\}) + v(\mathcal{X}_{h-1}) \\
 &= v(\mathcal{X}_{h+1}) + v(\mathcal{X}_{h-1}).
 \end{aligned} \tag{36}$$

첫 번째 부등식은  $\mathcal{X}_h$ 의 최적성에 따르며, 두 번째 부등식은  $\mathcal{X}_{h-1}$ 에  $j$ 를 더하면 가치가 증가할 수밖에 없기 때문이다.

$t_k \geq t_j$ 인 경우는 유사하다:

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

□

## 참고문헌

- 전민희. 2015. “[대입 수시 전략] 총 6번의 기회 … ‘상향·소신·안정’ 분산 지원하라.” 중앙일보, 8 월 26일. <https://www.joongang.co.kr/article/18524069>.
- Acharya, Mohan S., Asfia Armaan, and Aneeta S. Antony. 2019. “A Comparison of Regression Models for Prediction of Graduate Admissions.” In *Second International Conference on Computational Intelligence in Data Science*. <https://doi.org/10.1109/ICCID.2019.8862140>.
- Assad, Arjang. 1985. “Nested Optimal Policies for Set Functions with Applications to Scheduling.” *Mathematics of Operations Research* 10 (1): 82–99.
- Badanidiyuru, Ashwinkumar and Jan Vondrák. 2014. “Fast Algorithms for Maximizing Submodular Functions.” In *Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms*, 1497–1514. <https://doi.org/10.1137/1.9781611973402.110>.
- Balas, Egon and Eitan Zemel. 1980. “An Algorithm for Large Zero-One Knapsack Problems.” *Operations Research* 28 (5): 1130–54. <https://doi.org/10.1287/opre.28.5.1130>.
- Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2017. “Julia: A Fresh Approach to Numerical Computing.” *SIAM Review* 59: 65–98. <https://doi.org/10.1137/141000671>.
- Blum, Manuel, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. 1973. “Time Bounds for Selection.” *Journal of Computer and System Sciences* 7 (4): 448–61. [https://doi.org/10.1016/S0022-0000\(73\)80033-9](https://doi.org/10.1016/S0022-0000(73)80033-9).
- Calinescu, Gruia, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2011. “Maximizing a Monotone Submodular Function Subject to a Matroid Constraint.” *SIAM Journal on Computing* 40 (6): 1740–66. <https://doi.org/10.1137/080733991>.
- Carraway, Robert, Robert Schmidt, and Lawrence Weatherford. 1993. “An Algorithm for Maximizing Target Achievement in the Stochastic Knapsack Problem with Normal Returns.” *Naval Research Logistics* 40 (2): 161–73. <https://doi.org/10.1002/nav.3220400203>.
- Chekuri, Chandra, Jan Vondrák, and Rico Zenklusen. 2014. “Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes.” *SIAM Journal on Computing* 43 (6): 1831–79. <https://doi.org/10.1137/110839655>.
- Cormen, Thomas, Charles Leiserson, and Ronald Rivest. 1990. *Introduction to Algorithms*. Cambridge, MA: The MIT Press.
- Edmonds, Jack. 1971. “Matroids and the Greedy Algorithm.” *Mathematical Programming* 1: 127–36. <https://doi.org/10.1007/BF01584082>.
- Dantzig, George B. 1957. “Discrete-Variable Extremum Problems.” *Operations Research* 5 (2): 266–88.
- Dean, Brian, Michel Goemans, and Jan Vondrák. 2008. “Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity.” *Mathematics of Operations Research* 33 (4): 945–64. <https://doi.org/10.1287/moor.1080.0330>.
- Kellerer, Hans, Ulrich Pferschy, and David Pisinger. 2004. *Knapsack Problems*. Berlin: Springer.
- Kulik, Ariel, Hadas Shachnai, and Tami Tamir. 2013. “Approximations for Monotone and Non-monotone Submodular Maximization with Knapsack Constraints.” *Mathematics of Operations Research* 38 (4): 729–39. <https://doi.org/10.1287/moor.2013.0592>.
- Fisher, Marshall, George Nemhauser, and Laurence Wolsey. 1978. “An Analysis of Approximations for Maximizing Submodular Set Functions—I.” *Mathematical Programming* 14: 265–94.
- Fu, Chao. 2014. “Equilibrium Tuition, Applications, Admissions, and Enrollment in the College

- Market.” *Journal of Political Economy* 122 (2): 225–81. <https://doi.org/10.1086/675503>.
- Garey, Michael and David Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman and Company.
- Hartlaub, Vanessa and Thorsten Schneider. 2012. “Educational Choice and Risk Aversion: How Important Is Structural vs. Individual Risk Aversion?” *SOEPpapers on Multidisciplinary Panel Data Research*, no. 433. [https://www.diw.de/documents/publikationen/73/diw\\_01.c.394455.de/diw\\_sp0433.pdf](https://www.diw.de/documents/publikationen/73/diw_01.c.394455.de/diw_sp0433.pdf).
- Kahneman, Daniel. 2011. *Thinking, Fast and Slow*. New York: Macmillan.
- Markowitz, Harry. 1952. “Portfolio Selection.” *The Journal of Finance* 7 (1): 77–91. <https://www.jstor.org/stable/2975974>.
- Martello, Silvano and Paolo Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. New York: John Wiley & Sons.
- Meucci, Attilio. 2005. *Risk and Asset Allocation*. Berlin: Springer-Verlag, 2005.
- Nemhauser, George and Laurence Wolsey. 1978. “Best Algorithms for Approximating the Maximum of a Submodular Set Function.” *Mathematics of Operations Research* 3 (3): 177–88. <https://doi.org/10.1287/moor.3.3.177>.
- Parker, R. Gary and Ronald L. Rardin. 1988. *Discrete Optimization*. San Diego: Academic Press.
- Rozanov, Mark and Arie Tamir. 2020. “The Nestedness Property of the Convex Ordered Median Location Problem on a Tree.” *Discrete Optimization* 36: 100581. <https://doi.org/10.1016/j.disopt.2020.100581>.
- Sklarow, Mark. 2018. *State of the Profession 2018: The 10 Trends Reshaping Independent Educational Consulting*. Technical report, Independent Educational Consultants Association. <https://www.iecaonline.com/wp-content/uploads/2020/02/IECA-Current-Trends-2018.pdf>.
- Sniedovich, Moshe. 1980. “Preference Order Stochastic Knapsack Problems: Methodological Issues.” *The Journal of the Operational Research Society* 31 (11): 1025–32. <https://www.jstor.org/stable/2581283>.
- Steinberg, E. and M. S. Parks. 1979. “A Preference Order Dynamic Program for a Knapsack Problem with Stochastic Rewards.” *The Journal of the Operational Research Society* 30 (2): 141–47. <https://www.jstor.org/stable/3009295>.
- Van Huizen, Thomas and Rob Alessie. 2019. “Risk Aversion and Job Mobility.” *Journal of Economic Behavior & Organization* 164: 91–106. <https://doi.org/10.1016/j.jebo.2019.01.021>.
- Vazirani, Vijay. 2001. *Approximation Algorithms*. Berlin: Springer.
- Wolsey, Laurence. 1998. *Integer Programming*. New York: John Wiley & Sons.