The College Application Problem

Max Kapur

Seoul National University
Department of Industrial Engineering
Management Science/Optimization Lab
maxkapur@snu.ac.kr

June 12, 2022

Introduction

The optimal college application problem is a **novel combinatorial optimization problem**.

Maximize the expected maximum of a portfolio of random variables subject to a budget constraint.

Today's agenda:

- **Formulate** college application as an optimization problem.
- Our theoretical results, **solution algorithms**, and Julia implementation.

Methodological orientation

College application spans several areas of combinatorial optimization:

- Investment with uncertain payoff, search for the efficient frontier recall **portfolio allocation** models.
- Generalizes the **knapsack problem**: Integral packing constraint, NP-completeness.
- Objective is a **submodular set function**, but approximation results suggest college application is a relatively easy instance.


Model

The admissions process

Consider a **single student's** college application decision.

Market contains m **schools**, indexed by $\mathcal{C} = \{1 \dots m\}$. School j is named c_j.

Given a student's academic records, test scores, and demographic information, we can determine her **admissions probability** f_j at each school.

Let the **random variable** $Z_j \sim \text{Bernoulli}(f_j) = 1$ if student is admitted, 0 otherwise. Assume independent.

Let $\mathcal{X} \subseteq \mathcal{C}$ denote the set of schools, or **application portfolio**, to which a student applies.

Application fees, time to write essays, and/or legal limits **constrain** applicant behavior. We consider a single knapsack constraint $\sum_{j \in \mathcal{X}} g_j \leq H$ where g_j is called c_j's **application cost**.

Utility model

Let $t_j \geq 0$ denote the **utility** the student receives if she attends c_j. Wlog, $t_j \leq t_{j+1}$.

Let t_0 denote her utility if she doesn't get into college. Wlog, $t_0 = 0$ (see paper).

The student's overall utility is the t_j-value associated with the **best** school she gets into:

$$\text{Utility} = \max\{t_0, \max\{t_j Z_j : j \in \mathcal{X}\}\}$$

When the student's application portfolio is $\mathcal{X}$, we refer to her expected utility as the portfolio's **valuation** $v(\mathcal{X})$.

Unpacking the portfolio valuation function

To get $v(\mathcal{X})$ into a tractable form, let $p_j(\mathcal{X})$ denote the probability that the student **attends** c_j.

This happens if and only if she **applies** to c_j, is **admitted** to c_j, and is **not admitted** to any school she prefers to c_j:

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

Therefore,

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

Problem statement

Problem 1 (The college application problem)

$$\text{maximize} \quad v(\mathcal{X}) = \sum_{j \in \mathcal{X}} \left(f_j t_j \prod_{\substack{i \in \mathcal{X}: \\ i > j}} (1 - f_i) \right)$$

$$\text{subject to} \quad \mathcal{X} \subseteq \mathcal{C}, \quad \sum_{j \in \mathcal{X}} g_j \leq H$$

Problem 2 (The college application problem, INLP form)

$$\text{maximize} \quad v(x) = \sum_{j=1}^{m} \left(f_j t_j x_j \prod_{i > j} (1 - f_i x_i) \right)$$

$$\text{subject to} \quad x_j \in \{0, 1\}, j \in \mathcal{C}; \quad \sum_{j=1}^{m} g_j x_j \leq H$$

The status quo

Timothy Peck — May 7, 2021 — 5 — 12th Grade, College Lists

Safety, Target, & Reach Schools: How to Find the Right Ones

What's Covered:

- What Are Reach, Target, and Safety Schools?
- Factors that Impact Your Chances
- Elements of a Balanced College List

Creating a school list is an important-yet-tricky step in the college application process. A strategically constructed school list weighs your desire to attend reach schools—the institutions you dream about going to—along with safety schools where you're very likely to secure admission. Consequently, the ideal school list is balanced between reach, target, and safety schools, allowing you to shoot for the stars while also ensuring admission into at least one school.

What Are Reach, Target, and Safety Schools?

"Reach," "safety," and "target" are common terms used in college applications to describe the odds a student has of getting accepted at a particular institution. Understanding these terms, and which categories colleges fall into, is a critical step in the application process.

What is a Reach School?

Reach schools are colleges where you have less than a 25% chance of admission (this is your

The JoongAng

[대입 수시 전략] 총 6번의 기회 …'상향·소신·안정' 분산 지원하라

중앙일보 | 업데이트 2015.08.26 10:15 지면보기

전민희 기자 구독

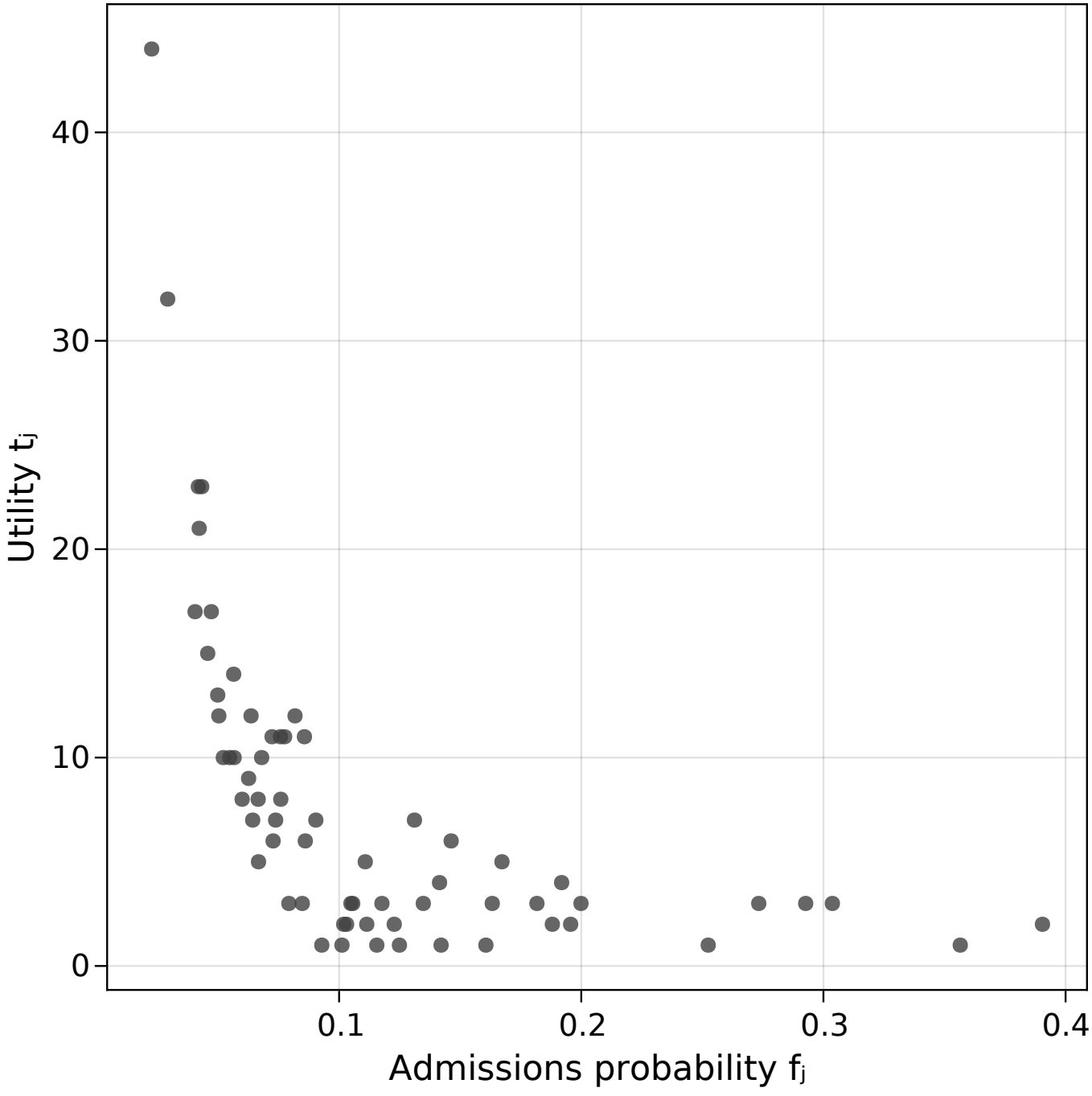
대학 최저학력기준 고려해 전략 지원
지난해 같은 전형 합격한 선배 내신 참고
수능 전 대학별고사 보는 곳은 최소화

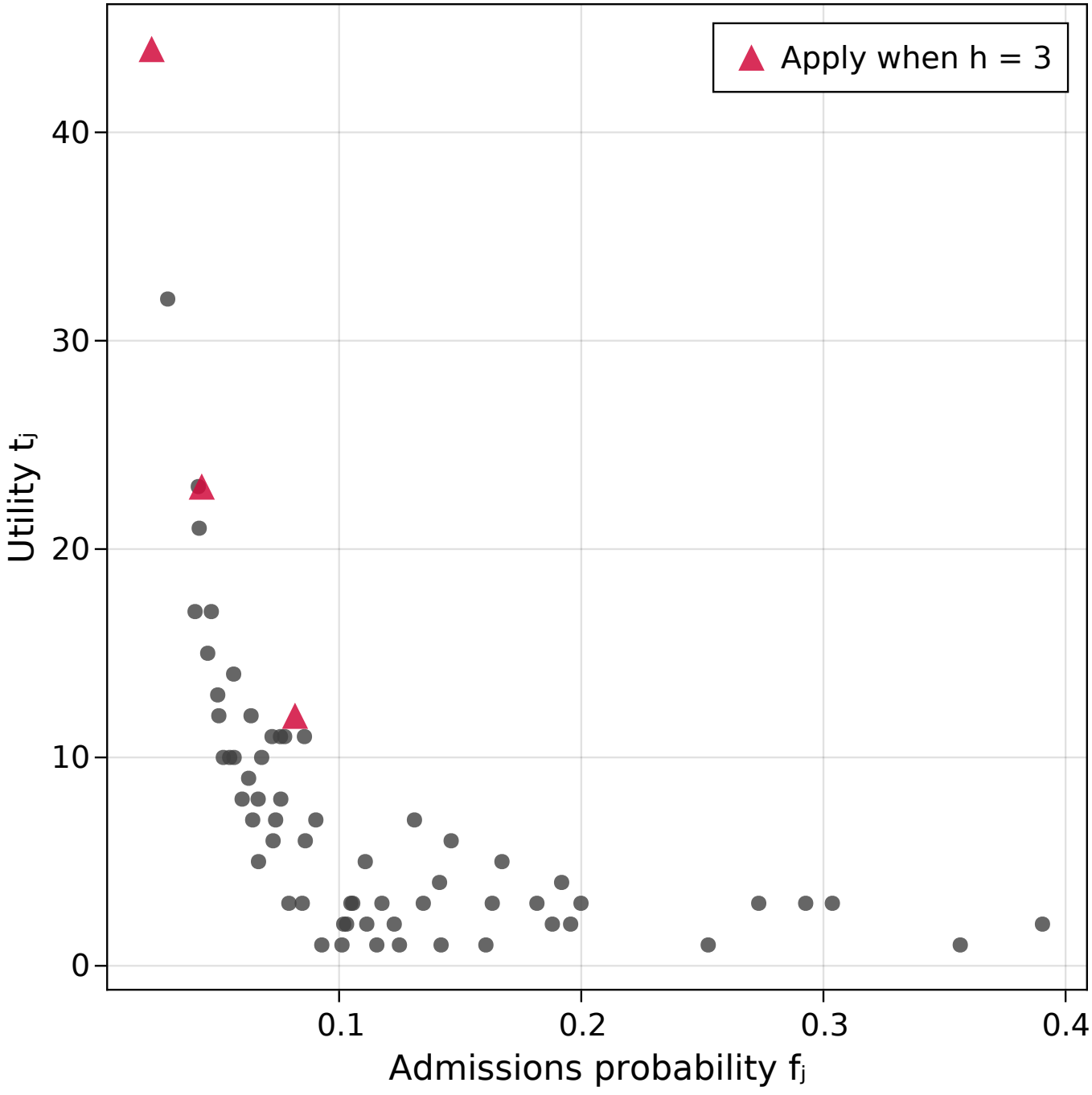
'지피지기 백전불태(知彼知己百戰不殆).' 적을 알고 나를 알면 백 번 싸워도 위태롭지가 않다는 뜻이다. 고대 중국의 병법서인 『손자』에 나온 말이지만 현대사회에서도 여러 가지 분야에서 회자된다. 그중 하나가 대학입시다. 특히 2주 앞으로 다가온 수시모집은 전형 종류가 다양해 '적'(모집전형) 을 알고, '나'(학생)에 대해 파악하는 게 무엇보다 중요하다.

자신의 학교생활기록부, 교과성적, 대학별고사 준비 상황, 예상 수능점수, 최저학력기준 통과 가능성에 대해 자세히 살핀 후 지원해야 합격률을 높일 수 있다. 수시모집 마무리 전략을 알아봤다.

논술전형도 학생부 성적 기준으로 지원

Do you trust the admissions consultant's advice?


40
30
20
10
0
Utility t_j
0.1
0.2
0.3
0.4
Admissions probability f_j


Apply when h = 3
Utility t_j
Admissions probability f_j
0
10
20
30
40
0.1
0.2
0.3
0.4

Existing solutions

In practice, most use **distributive heuristics** such as distributing applications evenly among reach, target, and safety schools. Turns out to be a **risk-averse** approach.

Another intuitive idea is the **linearization heuristic**. Since the expected utility associated with applying to c_j (alone) is $f_j t_j$, solve the knapsack problem

$$\text{maximize} \sum_{j \in \mathcal{X}} f_j t_j \qquad \text{subject to} \sum_{j \in \mathcal{X}} g_j \leq H$$

as a surrogate. But this solution can be **arbitrarily bad**.

Fu (2014) solved a similar problem by **enumeration**, which is intractable for $m \geq 20$ or so.

Our algorithms provide both **time and accuracy guarantees**.

Our algorithms

Overview of our algorithms

We consider two cases of the college application problem:

- **Alma's problem:** Special case in which $g_j = 1$, i.e. a cardinality constraint instead of knapsack.

  We provide an **exact, polynomial-time** solution via greedy algorithm.

- **Ellis's problem:** The general case, which is NP-complete.

  We provide four algorithms, including a **fully polynomial-time** $(1 - \varepsilon)$**-approximation algorithm**.

In general, submodular maximization over a knapsack constraint is $(1 - 1/e)$-inapproximable (Kulik et al. 2013), so the existence of an FPTAS for Ellis's problem is a significant result.

Alma's problem

. . . is the **special case** in which each $g_j = 1$.

Then H is simply a limit on the number of schools you can apply to. This case mirrors the main Korean admissions cycle, in which $h = 3$ and $m = 202$.

(In this case, we can assume that $H < m$; therefore, any polynomial in H is bound by a polynomial in m. We write h in lowercase to emphasize this fact.)

Fisher et al. (1978) showed that a greedy algorithm is $(1 - 1/e)$-optimal for monotone submodular maximization. We show that the same algorithm is *exact* for college application and improve its computation time.

Optimality of the greedy algorithm

The following **nestedness property** for Alma's problem implies the optimality of a **greedy algorithm** that adds schools one at a time, maximizing $v(\mathcal{X})$ at each addition.

Theorem 1

There exists a sequence of portfolios $\{\mathcal{X}_h\}_{h=1}^{m}$ satisfying the nestedness relation

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

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

Using a variable-elimination technique, we reduce the cost of computing $v(\mathcal{X})$ to amortized unit time, obtaining an $O(hm)$ algorithm.

Algorithms for Ellis's problem

By setting $f_j = \varepsilon$, we can make $v(\mathcal{X})$ arbitrarily close to a linear function. This means that knapsack reduces to Ellis's problem, and therefore Ellis's problem is NP-complete.

The general problem is **NP-complete** (reduction from knapsack). We offer four algorithms:

- A linear relaxation and **branch-and-bound scheme** allow us to interface with existing INLP solvers.
- A **dynamic program based on total expenditures**. Exact solution in $O(Hm + m \log m)$ time (pseudopolynomial).
- A different DP based on truncated portfolio valuations. $(1 - \varepsilon)$-optimal solution in $O(m^3/\varepsilon)$ time: an **FPTAS**!
- A **simulated annealing heuristic**. Fast, typically within 2% of optimality.

Algorithms for Ellis's problem

By setting $f_j = \varepsilon$, we can make $v(\mathcal{X})$ arbitrarily close to a linear function. This means that knapsack reduces to Ellis's problem, and therefore Ellis's problem is NP-complete.

The general problem is **NP-complete** (reduction from knapsack). We offer four algorithms:

- A linear relaxation and **branch-and-bound scheme** allow us to interface with existing INLP solvers. (see appendix)
- A **dynamic program based on total expenditures**. Exact solution in $O(Hm + m \log m)$ time (pseudopolynomial).
- A different DP based on truncated portfolio valuations. $(1 - \varepsilon)$-optimal solution in $O(m^3/\varepsilon)$ time: an **FPTAS**!
- A **simulated annealing heuristic**. Fast, typically within 2% of optimality.

Expenditures-based dynamic program

Let $V[j, h]$ denote the value of the optimal portfolio that uses only the schools $\{1, \ldots, j\}$ and costs no more than h.

Then we can use the following **Bellman equation** to compute all the $V[j, h]$-values recursively.

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

This equation works because the t_j-values are indexed in ascending order.

Filling a lookup table with the $V[j, h]$-values therefore costs $O(Hm + m \log m)$, and then computing $\mathcal{X}$ is trivial.

Very effective on typical instances where g_j are small integers.

Valuation-based dynamic program: Fixed-point math

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 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.

$\bar{U} = \sum_{j \in \mathcal{C}} f_j t_j$ is an upper bound on the valuation of any portfolio, so the set $\mathcal{V}$ of possible valuations possible in the fixed-point framework is finite:

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

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

Valuation-based dynamic program: Recursion relation

Let $G[j, v]$ denote the application cost of the least expensive portfolio that uses only schools $\{1, \ldots, j\}$ and has (rounded) valuation at least v. We argue that

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

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

Now we can fill a lookup table with the entries of $G[j, v]$ and compute an approximately optimal portfolio.

In the paper, we show that setting $P \leftarrow \left\lceil \log_{10}\left(m^2/\varepsilon\bar{U}\right)\right\rceil$ guarantees a $(1-\varepsilon)$-optimal portfolio, and that this yields a table of size $O(m^3/\varepsilon)$, meaning that this algorithm is **an FPTAS**.

Simulated annealing algorithm

SA is a familiar randomized technique for neighborhood search.

No accuracy guarantee, but computation time is essentially negligible $\Longrightarrow$ popular in real-time applications such as vehicle routing where a "good enough" solution is needed quickly.

Ingredients for SA:

- An **initial solution**. We use the linearization heuristic, and solve the associated knapsack problem with the greedy heuristic.
- A randomized procedure for **generating neighbors**. We add schools randomly until $\mathcal{X}$ becomes infeasible, then remove until feasibility restored.
- **Temperature parameters** determine how likely the algorithm is to change candidates. $T = 1/4$ and $r = 1/16$ chosen by grid search.

Computational results

Experimental procedure

Three computational experiments designed to test the efficiency and accuracy of our solution algorithms.

Recipe for an experimental instance with m schools:

- Each t_j drawn from an exponential distribution.
- f_j chosen to correlate with $1/t_j$ (creates reach/safety schools).
- Alma's problem: $h = \lfloor m/2 \rfloor$.
- Ellis's problem: g_j small integers, $H = \lfloor \frac{1}{2} \sum g_j \rfloor$.

All algorithms implemented and tested in Julia (v1.8.0b1). Code available on Github (Kapur 2022) and in Julia registry (OptimalApplication.jl).

Experiment 1

. . . compares **two data structures** for the greedy algorithm for Alma's problem.

50 markets in each cell, time recorded as best of three reps. Table shows mean (std) computation time in ms.

m	**Greedy alg. with list**	**Greedy alg. with heap**
16	0.00 (0.00)	0.01 (0.00)
64	0.03 (0.00)	0.08 (0.02)
256	0.15 (0.01)	0.97 (0.22)
1024	2.31 (0.05)	14.44 (1.66)
4096	37.85 (0.61)	245.74 (17.33)
16384	585.75 (2.11)	4728.59 (552.25)

$\arg\max$ operation at each iteration made heap an appealing data structure, but because *every* key is updated, the savings aren't worth the cost.

Experiment 2

. . . compares the speed of **guaranteed-accuracy** algorithms for Ellis's problem.

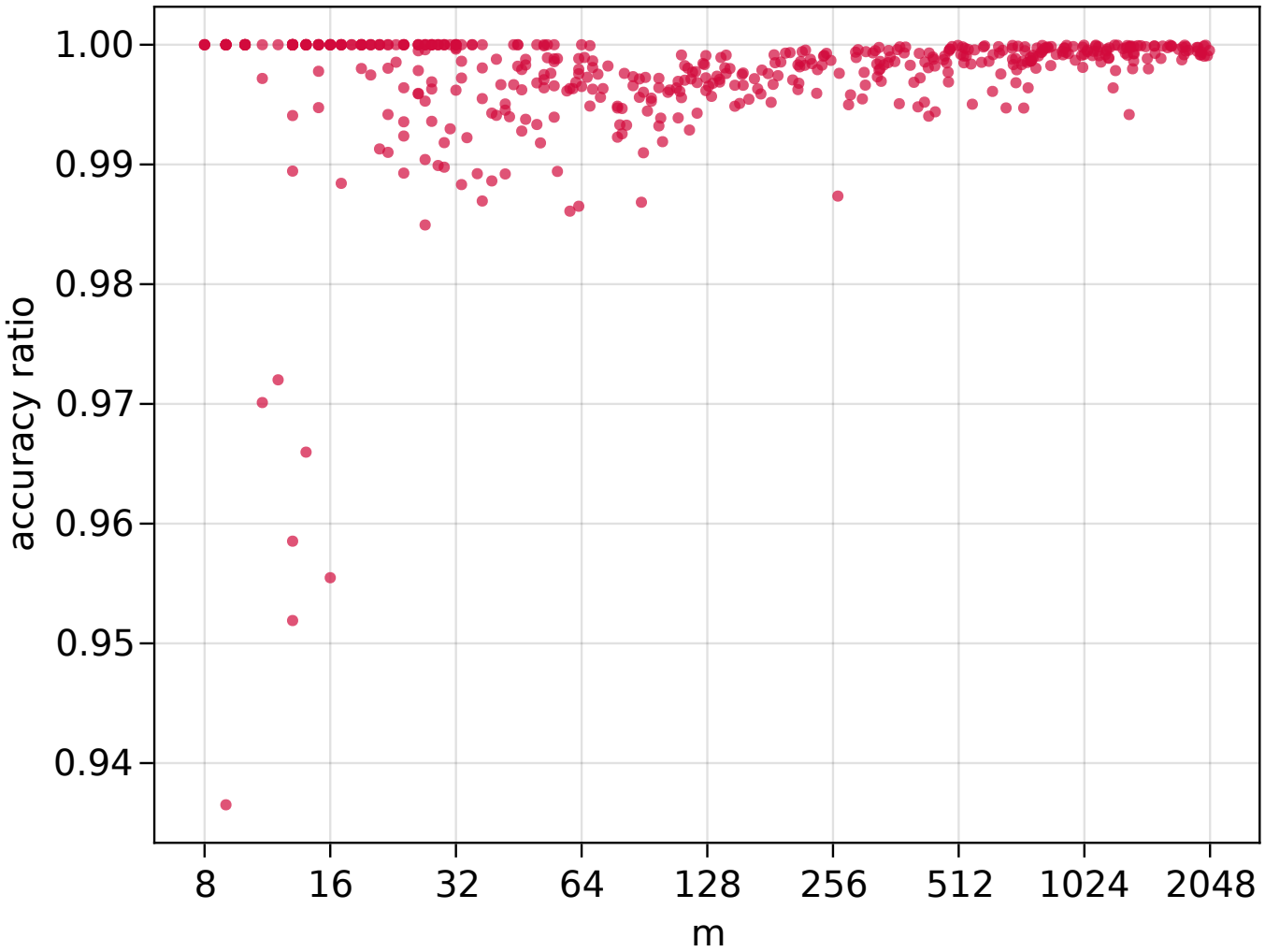
m	**Branch & bound**		**App. costs DP**		**FPTAS,** $\varepsilon = 0.5$		**FPTAS,** $\varepsilon = 0.05$	
8	0.04	(0.02)	0.01	(0.00)	0.05	(0.01)	0.21	(0.06)
16	0.22	(0.11)	0.07	(0.01)	0.43	(0.10)	3.15	(0.74)
32	166.20	(422.31)	0.31	(0.05)	2.38	(0.38)	33.38	(11.68)
64	—	(—)	1.36	(0.18)	15.32	(2.77)	405.73	(125.98)
128	—	(—)	6.52	(0.72)	84.50	(22.59)	2362.19	(1095.11)
256	—	(—)	31.23	(1.91)	1085.60	(1186.24)	22129.92	(6588.40)

Costs DP has a pronounced advantage because g_j are small integers.

Bottleneck for FPTAS is computer memory rather than time.

Experiment 3

. . . investigates the **empirical accuracy** of the simulated annealing heuristic in 500 synthetic instances.



Conclusion

Conclusion

"Maximax" form, integrality constraints make the college application problem **theoretically interesting**. Formally, it is a submodular maximization problem, but its approximability is more like knapsack (cf. Fisher et al. 1978; Kulik et al. 2013; Kellerer et al. 2004).

Solutions to the college application problem have **practical value**: US admissions consultants charge an average of $200/hr (Sklarow 2018)!

$\Rightarrow$ Open-sourcing our code for public benefit (Kapur 2022).

Possible extensions of our model:

- Explicit treatment of risk aversion as in the classic Markowitz (1952) model.
- Diversification constraints like Korea's application fields.
- Reduce memory usage of dynamic programs.

References I

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.

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

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.

Kapur, Max. 2022. "OptimalApplication." GitHub repository. https://github.com/maxkapur/OptimalApplication.jl.

References II

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

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

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

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.

Summary of algorithms

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

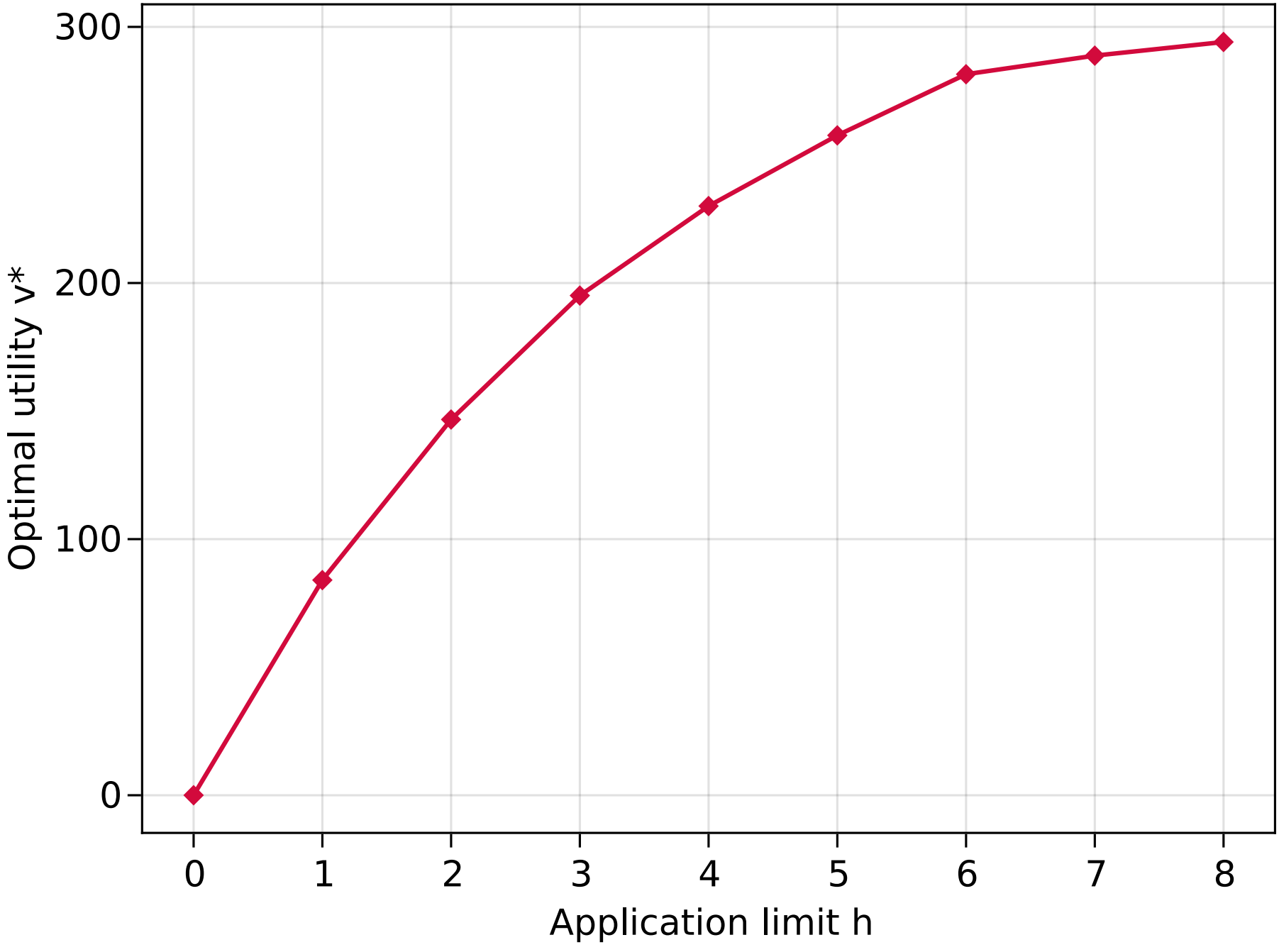
A small example

College data and optimal application portfolios for a fictional market with $m = 8$ schools.

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

By the nestedness property, the optimal portfolio when the application limit is h consists of the h schools having priority h or less.

Valuation function appears on next slide. Always concave.


300
200
100
0
Optimal utility v*
0
1
2
3
4
5
6
7
8
Application limit h

Branch-and-bound algorithm: LP relaxation

Our branch-and-bound framework is based on the following LP relaxation for Ellis's problem.

Problem 3 (LP relaxation for Ellis's problem)

$$\text{maximize} \quad v_{\mathrm{LP}}(x) = \sum_{j=1}^{m} f_j t_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{m} g_j x_j \leq H, \quad x \in [0,1]^m$$

We **tighten the LP bound** by recycling the variable-elimination technique from the proof of the nestedness theorem.

Problem 3 is a continuous knapsack problem, easy to solve (Dantzig 1957; Balas and Zemel 1980).

Branch-and-bound algorithm: Possible improvements

A straightforward implementation is OK on small instances ($m \leq 35$), but leaves **considerable room for improvement** using better node-selection heuristics.

In the associated Julia package, we provide **modular containers for subproblems and their LP relaxations** that can interface with a more sophisticated branch-and-bound solver.

A branch-and-bound algorithm may be necessary to handle more advanced constraint structures.