Robust Admission Via Two-Stage Stable Matching Under Ranking Uncertainty

Gustavo Angulo* Matías Giddings† Pablo Marshall‡

Abstract

We study a two-stage admission and assignment problem under uncertainty arising in university admission systems. In the first stage, applicants are admitted to an initial two-year program. In the second stage, admitted applicants are assigned to degree programs through an articulation mechanism subject to capacity constraints. Uncertainty stems from the academic performance of admitted applicants within the initial program, as this exit ranking determines the sequence in which students are processed by the articulation mechanism. Admission decisions must therefore be robust to all possible exit orders, giving rise to a robust stable matching problem.

We introduce a notion of robust compatibility for sets of admitted applicants and establish some of its properties. Building on a standard integer programming formulation of stable matching, we derive an adversarial formulation that determines whether a given admission set remains compatible under worst-case realizations of the exit order. This adversarial model underpins a sequential admission algorithm that incrementally constructs a robustly compatible set of applicants. Computational experiments on real admission data from three consecutive cohorts show that computational effort is highly unevenly distributed across iterations. In particular, certifying acceptance decisions is substantially more expensive than certifying rejections, with overall running time dominated by a small number of late iterations in highly congested regimes. We also address how the algorithm has been applied in practice and highlight the benefits in first-year retention rate driven by the proposed mechanism.

Keywords: OR in Education, Robust Stable Matching, College Admission, Uncertain Preferences

*Corresponding author: gangulo@uc.cl, School of Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile 7820436.

†matias.giddings@uc.cl, School of Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile 7820436.

‡pmarshall@uc.cl, School of Management and Bachillerato Inicia, Pontificia Universidad Católica de Chile, Santiago, Chile 7820436.

1 Introduction

Universities may admit students to a preparatory or general program before they choose or enter a specific degree program (Bordon and Fu, 2015). When access to these subsequent programs is subject to capacity constraints, admission decisions must account for the feasibility of later assignments. A sufficient total number of places does not resolve this problem if demand concentrates on a few programs and students consider only a limited set of alternatives. Institutions must therefore coordinate admission with subsequent assignment, taking account of the information available when each decision is made.

We study a two-stage admission problem in which applicants declare a short, strictly ordered list of degree programs. In the first stage, the university selects a cohort using these declarations and a fixed *entrance ranking*. In the second stage, it assigns the admitted students to their declared programs, a process that we call *articulation*. This assignment follows a common *exit ranking* determined by academic performance in the initial program: students are processed in that order and each takes her most preferred program with an available place.

The difficulty is that the exit ranking is unknown when admission decisions are taken, and a cohort that can be fully assigned under one ranking may leave students unmatched under another. We call a cohort *robustly compatible* if articulation assigns every student to a declared program under every exit ranking. Our objective is to select such a cohort while giving lexicographic priority to applicants with higher entrance rankings. Robust compatibility gives admitted students a guarantee that is independent of a particular forecast of their relative performance, without assigning probabilities to rankings or requiring them to be equally likely. Full robustness thus provides a conservative benchmark for admission rules that allow a controlled risk of unsuccessful articulation.

Our work is motivated by *Bachillerato Inicia*, a preparatory program created in 2024 by the Pontificia Universidad Católica de Chile (PUC) for graduates of technical-professional secondary education. Applicants declare three degree programs, and the admission mechanism studied here has been used in the 2024, 2025, and 2026 admission cycles. The application illustrates how concentrated demand can constrain admission even when aggregate articulation capacity exceeds the planned intake. We use it as a case study to evaluate the proposed method and examine its implementation.

We establish basic structural properties of robust compatibility, including its preservation under taking subsets, appending programs to preference lists, and increasing capacities. We

then prove that deciding robust compatibility is co-NP-complete. This difficulty persists when testing whether one student can be added to an already robustly compatible cohort, even with exactly three declared programs per student and capacities at most two. To perform this test, we develop an adversarial integer programming formulation based on stable matching constraints. A layered representation of priority relations yields a formulation with quadratically many variables and constraints when preference lists have bounded length. Solving this model either certifies robust compatibility or identifies an exit ranking under which some admitted student remains unmatched.

The adversarial model serves as the decision procedure in a sequential admission algorithm. Applicants are considered in entrance-ranking order and accepted whenever their inclusion preserves robust compatibility. The hereditary structure of the feasible family ensures that this procedure returns its lexicographically maximal member. This objective can produce a smaller cohort than maximizing cardinality, and we discuss the complexity and formulation implications of the latter problem. Because the submitted lists determine admission as well as assignment, the combined mechanism also raises incentive questions. We show that it is not strategy-proof by constructing a deviation that benefits an applicant under every exit ranking.

Computational experiments on the three admission cohorts examine the performance of the algorithm and the effects of extending preference lists and expanding program capacities. They show that computational effort is concentrated in a small number of late acceptance decisions, which require certifying compatibility across all exit rankings. The implementation study documents repeated admission rounds, students' subsequent preference changes, and observed retention and assignment outcomes. It also compares entrance and exit rankings for the 2025 cohort, reporting substantial reordering and providing empirical context for the uncertainty faced at admission.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature. Section 3 defines the robust admission problem, and Section 4 studies its structural properties, incentive implications, and computational complexity. Section 5 develops the adversarial formulation and the admission algorithm. Section 6 presents the case study and computational results, and Section 7 discusses the implementation experience. Section 8 concludes.

2 Literature review

The design of centralized admission mechanisms is grounded in the theory of stable matching, initiated by the seminal work of Gale and Shapley (1962). They introduced the concept of stability, defined as the absence of blocking pairs, and the Deferred Acceptance algorithm, which guarantees finding a stable matching. This framework was established as the canonical theoretical foundation for two-sided matching markets by Roth and Sotomayor (1990) and later adapted to school choice by Abdulkadiroğlu and Sönmez (2003), who demonstrated that student-optimal stable mechanisms eliminate justified envy and provide strategy-proofness for students. While these classical models assume that agent preferences and institutional priorities are fixed and known, recent literature has increasingly focused on environments characterized by uncertainty and dynamic changes.

Early departures from the deterministic model focused on incomplete information regarding agent preferences. Roth (1989) showed that stability and strategy-proofness are generally incompatible when preferences are private. This spurred a line of research investigating stability concepts under various information structures, ranging from Bayesian settings to informationally constrained environments (Chakraborty et al., 2010; Liu et al., 2014; Ehlers and Massó, 2015; Bikhchandani, 2017). In these environments, the primary challenge is often incentivizing truthfulness or defining stability when agents cannot perfectly identify blocking pairs. Our setting differs as the uncertainty is not about hidden preferences, but about future priorities (rankings) that simply do not exist at the time of the initial decision.

A critical distinction in modern matching literature is between robustness against strategic manipulation and robustness against structural perturbations. Kojima (2011) formalized robust stability as immunity to combined manipulation strategies (misreporting followed by blocking) and showed that such robustness is achievable if and only if priority structures are acyclic. Afacan (2012) extended this notion to group deviations and established an impossibility result: no mechanism can satisfy group robust stability, even under acyclic priorities, motivating weaker robustness notions. In contrast, Mai and Vazirani (2018) and more recently Gangam et al. (2026) study robust stable matchings from a structural perspective, seeking matchings that remain stable across multiple realized instances of preferences. Gangam et al. (2026) demonstrate that while the set of robust matchings retains a lattice structure under simple local changes, this structure collapses under general, multi-agent perturbations. These results indicate that tractability can deteriorate under broad uncertainty, motivating optimization-based approaches when seeking robust solutions. Other structural approaches

include the work of Genc et al. (2017) on supermatches that are resilient to breakups, and Chen et al. (2021) on d-robustness against swap perturbations. Furthermore, Aziz et al. (2020) showed that verifying stability guarantees can be NP-complete under uncertain preferences.

Our work is most closely related to the emerging field of two-stage stable matching, where decisions are split temporally. Gajulapalli et al. (2020) and Bredereck et al. (2020) study how to adapt stable matchings to evolving preferences or new arrivals without disrupting existing assignments. Most notably, Faenza et al. (2024, 2025) have recently introduced a framework for two-stage stochastic stable matching. Their model optimizes a first-stage matching to minimize the expected cost of recourse (reassignment) after uncertainty is revealed, reducing the problem to minimum cuts in rotation digraphs.

Our problem concerns robust feasibility: the first-stage decision selects a set of students for which a complete stable assignment exists under every exit ranking. The second-stage matching may adapt to the realized ranking. Thus, robustness applies to the admitted set, while stability and complete assignment are required in every scenario. This differs from optimizing a first-stage matching to minimize expected recourse costs (Faenza et al., 2024). To our knowledge, this admission problem under global uncertainty in a common priority order has not been explicitly studied.

3 The robust admission problem

Consider a set of applicants S seeking admission to an initial preparatory program, which after a first stage of study allows students to articulate to a set of degree programs P. Each applicant $s \in S$ has declared a strictly ordered subset of programs $P_s \subseteq P$, representing their preferences. Preferences are denoted by $\rhd_s$, where $q \rhd_s p$ means that applicant s prefers program q over p. Conversely, each program $p \in P$ has a set of interested applicants $S_p = \{s \in S : \ p \in P_s\}$ and a predetermined number of available slots c_p. We assume that students prefer any program in their list to being left without a seat, denoted by $\emptyset$, and thus for all $s \in S$ we extend $\rhd_s$ to $P_s \cup \{\emptyset\}$ so that $p \rhd_s \emptyset$ for all $p \in P_s$. Similarly, we assume that any program $p \in P$ prefers to fill its slots with any student in S_p rather than leaving empty seats.

First-stage problem (ex-ante admission). The objective is to select a set $S^* \subseteq S$ that is lexicographically maximal with respect to the entrance ranking among all sets for which every admitted applicant can articulate under every exit ranking.

Second-stage problem (ex-post articulation). Consider the set of admitted applicants $S^* \subseteq S$ who are ready for articulation and must be assigned to one of their declared undergraduate programs. An articulation mechanism proceeds as follows: first, students in S^* are ranked according to their exit ranking, represented by a strict priority order $\blacktriangleright$. Students are reviewed sequentially, starting from the highest-ranked student. For each student s, programs in their preference list P_s are checked in descending order of priority $\rhd_s$. The student is assigned to the highest-priority program that still has available slots. Once a program reaches its capacity, it no longer accepts additional students.

Definition 1. For any $S^* \subseteq S$, the set of all strict total orders $\blacktriangleright$ over S^* is denoted by $\mathcal{R}(S^*)$.

We interpret any $\blacktriangleright \in \mathcal{R}(S^*)$ as an exit ranking or common preferences over S^* used by all programs in the articulation mechanism.

Definition 2. We say that $S^* \subseteq S$ is compatible with exit ranking $\blacktriangleright \in \mathcal{R}(S^*)$ if all students in S^* are matched to an acceptable program by the articulation mechanism.

Definition 3. We say that $S^* \subseteq S$ is robustly compatible if, for every exit ranking $\blacktriangleright \in \mathcal{R}(S^*)$, S^* is compatible with $\blacktriangleright$.

Example 1 (Robust vs. non-robust compatibility)**.** *Consider four programs* $P = \{A, B, C, D\}$ *with capacities* $(c_A, c_B, c_C, c_D) = (2, 1, 1, 1)$ *and an admitted cohort* $S^* = \{1, 2, 3, 4, 5\}$*. Each student* $s \in S^*$ *has submitted a strict preference ranking* $\rhd_s$ *over three declared programs, shown in Table 1. We examine how different realizations of the exit ranking affect the compatibility of the same admitted set.*

Table 1: Applicants' declared preference rankings.

s	$\rhd_s$
1	$A \rhd B \rhd C$
2	$A \rhd B \rhd D$
3	$C \rhd D \rhd B$
4	$A \rhd C \rhd D$
5	$B \rhd D \rhd A$

Exit ranking $\alpha: \ 2 \blacktriangleright 4 \blacktriangleright 5 \blacktriangleright 1 \blacktriangleright 3$**.** *Students 2 and 4 are processed first and occupy the two available seats in program A. Student 5 is then assigned to program B. When student 1*

is considered, both A and B are already full, so she is assigned to her third choice, C. Finally, student 3 is processed. Since C is full, she is assigned to her second choice, D. Thus, under ranking α, all students in S^* are successfully articulated, and S^* is compatible with α.

Exit ranking *β : $2 \blacktriangleright 4 \blacktriangleright 5 \blacktriangleright 3 \blacktriangleright 1$. Students 2 and 4 again fill program A, and student 5 is assigned to program B. However, student 3 now appears before student 1 and is assigned to program C. When student 1 is processed last, all three programs in her declared list, A, B, and C, are already at capacity. Although program D still has an available seat, student 1 did not declare it. She therefore remains unmatched, and the same cohort S^* is incompatible with ranking β.*

This example illustrates that compatibility of an admitted set may depend on the realized exit ranking. Indeed, while S^* is compatible with some rankings, it fails to be compatible with others. By contrast, if the admitted set is reduced to $\{1, 2, 3, 4\}$, then all four students can be matched for every one of the $4!$ possible exit rankings. To see this, consider the last processed student. If the last student is 1, 2, or 4, then the total capacity of her declared programs is four, so three preceding assignments cannot fill all of them. If the last student is 3, then the first two students among $\{1, 2, 4\}$ occupy the two seats in A, and the third occupies either B or C; hence program D remains available for student 3. Therefore, $\{1, 2, 3, 4\}$ is robustly compatible, whereas admitting student 5 violates robustness.

It can be observed that the articulation mechanism corresponds to a simple serial dictatorship, which yields a unique assignment given $S^* \subseteq S$ and exit ranking $\blacktriangleright \in \mathcal{R}(S^*)$. Moreover, such an assignment is a stable matching. This connection allows us to draw tools from the rich literature on the subject. In the following section, we formalize the model.

4 Definitions and basic properties

Definition 4. A pair $(s, p) \in S \times P$ is admissible if $p \in P_s$, or equivalently in our case, if $s \in S_p$. The set of admissible pairs is denoted by A, and for $S^* \subseteq S$, its restriction to $S^* \times P$ is denoted by $A(S^*)$.

Definition 5. For $\mu \subseteq A$, we write $\mu(s) = \{p \in P : (s, p) \in \mu\}$ for $s \in S$ and $\mu(p) = \{s \in S : (s, p) \in \mu\}$ for $p \in P$. A matching is a set $\mu \subseteq A$ such that $|\mu(s)| \leq 1$ for all $s \in S$ and $|\mu(p)| \leq c_p$ for all $p \in P$. If $\mu(s) = \{p\}$, we simply write $\mu(s) = p$. For $S^* \subseteq S$, an S^*-matching is a matching $\mu \subseteq A(S^*)$.

Definition 6. Given an S^*-matching $\mu \subseteq A(S^*)$ and $\blacktriangleright \in \mathcal{R}(S^*)$, a pair $(s,p) \in A(S^*) \setminus \mu$ blocks μ under $\blacktriangleright$ if (i) $\mu(s) = \emptyset$ or $p \rhd_s \mu(s)$ and (ii) $|\mu(p)| < c_p$ or $|\{t \in \mu(p) : \ s \blacktriangleright t\}| \geq 1$. If μ has no blocking pair under $\blacktriangleright$, μ is a $\blacktriangleright$-stable matching.

In other words, a matching μ is not $\blacktriangleright$-stable if there exists an admissible pair (s,p), unmatched to each other by μ, such that s prefers to be matched to p (either because she is unmatched or matched to a program she prefers less than p) and p is willing to accept s (either because it has empty slots or one of them is occupied by another student it prefers less than s under $\blacktriangleright$).

In general, there can be multiple stable matchings when programs have different priorities over students. However, the situation simplifies substantially when programs share a common preference profile as in our case. Proposition 2 will be instrumental in our developments.

Proposition 2 (Irving et al. (2008))**.** *For any $S^* \subseteq S$ and $\blacktriangleright \in \mathcal{R}(S^*)$, the assignment given by the articulation mechanism is the only possible $\blacktriangleright$-stable matching.*

Proposition 3 below shows that the collection of robustly compatible sets is down-monotone or closed under inclusion. This property ensures that if an admitted student declines to enroll, a situation that can occur in actual admission processes, the remaining admitted students still form a robustly compatible set.

Proposition 3. *Let $S', S^* \subseteq S$ be such that $S' \subseteq S^*$ and S^* is robustly compatible. Then, S' is robustly compatible.*

Proof. Fix an arbitrary exit ranking $\blacktriangleright' \in \mathcal{R}(S')$ and consider any extension $\blacktriangleright^* \in \mathcal{R}(S^*)$ such that $s \blacktriangleright^* t$ for all $s \in S'$ and $t \in S^* \setminus S'$. Note that the execution of the articulation mechanism over S' under $\blacktriangleright'$ is equivalent to halting its execution over S^* under $\blacktriangleright^*$ once all students in S' are processed. Since S^* is robustly compatible, the latter matches all students in S^*, in particular, those in S'. Since $\blacktriangleright' \in \mathcal{R}(S')$ was arbitrary, S' is robustly compatible. □

Propositions 4 and 5 below show that as students gain more flexibility in choosing and articulating to programs, robust compatibility is not lost.

Proposition 4. *Let $S^* \subseteq S$ be robustly compatible. If students append acceptable programs to the end of their lists, S^* remains robustly compatible.*

Proof. Let $\blacktriangleright \in \mathcal{R}(S^*)$. If all students in S^* are matched by the articulation mechanism under $\blacktriangleright$, none of them will be matched to a new acceptable program added at the end of their lists.

Therefore, the articulation mechanism yields the same matching and, in particular, S^* remains compatible with $\blacktriangleright$. Since this holds for any $\blacktriangleright \in \mathcal{R}(S^*)$, S^* remains robustly compatible. $\square$

Proposition 5. *Let $S^* \subseteq S$ be robustly compatible. If no program loses slots, S^* remains robustly compatible.*

Proof. Let $\blacktriangleright \in \mathcal{R}(S^*)$. It is known that the student-optimal stable matching cannot harm any student when slots increase (Roth and Sotomayor (1990)). By Proposition 2, the uniqueness of the stable matching implies it is student-optimal. Therefore, if all students in S^* are matched by the articulation mechanism under $\blacktriangleright$, they remain matched when slots increase, that is, S^* remains compatible with $\blacktriangleright$. Since this holds for any $\blacktriangleright \in \mathcal{R}(S^*)$, S^* remains robustly compatible. $\square$

We next establish Proposition 6, which will allow us to strengthen the IP formulation to be presented in Section 5.

Proposition 6. *Let $S^* \subsetneq S$ be robustly compatible and let $s \in S \setminus S^*$. Then, for every $\blacktriangleright \in \mathcal{R}(S^* \cup \{s\})$, the articulation mechanism over $S^* \cup \{s\}$ leaves at most one student unmatched.*

Proof. Fix an arbitrary exit ranking $\blacktriangleright \in \mathcal{R}(S^* \cup \{s\})$ and let $\hat{\mu} \subseteq A(S^* \cup \{s\})$ be the unique $S^* \cup \{s\}$-matching produced by the articulation mechanism under $\blacktriangleright$. Let $\mu \subseteq A(S^*)$ be the S^*-matching produced by the articulation mechanism under the restriction of $\blacktriangleright$ to S^*. By robust compatibility of S^*, μ matches all students in S^*. Let u be the number of unmatched students produced by $\hat{\mu}$, i.e., $u = |\{t \in S^* \cup \{s\} : \ \hat{\mu}(t) = \emptyset\}|$.

We compare the two executions of the mechanism (on S^* and on $S^* \cup \{s\}$) along the common order $\blacktriangleright$. Consider the first student $t_1 \in S^*$ in the order $\blacktriangleright$ having different assignments between the two executions. If no such student exists, then all students in S^* receive the same program under μ and $\hat{\mu}$. Hence, $\hat{\mu}$ matches all students in S^* and $u \leq 1$ trivially. If t_1 exists, note that necessarily $s \blacktriangleright t_1$. Since t_1 is matched under μ, she has some assigned program $p_0 = \mu(t_1) \in P_{t_1}$ when only students in S^* are considered. Because t_1 is the earliest student whose assignment differs, t_1 must find p_0 full in the execution on $S^* \cup \{s\}$ at the moment she is processed, and moreover, her seat in p_0 must have been taken by s. Now, in the execution on $S^* \cup \{s\}$, student t_1 either becomes unmatched, or she is assigned to some program $p_1 \in P_{t_1}$ such that $p_0 \rhd_{t_1} p_1$. If t_1 becomes unmatched, after processing t_1, all programs have the same number of remaining slots in both executions (because the only change has been p_0 accepting s instead of t_1). Therefore, from that point on, both executions

coincide and thus $u = 1$. Otherwise, t_1 is assigned to $p_1 \neq p_0$, consuming one seat of p_1 that, in the execution on S^*, was either left empty or allocated later to some student of S^*. If it was left empty in the execution on S^*, then this reassignment does not change the assignment of remaining students, so $u = 0$ in this case. If it was allocated later to some student, then there exists a set of students whose assignments are modified as a consequence, because the displaced student must move to another acceptable program, and so on.

Formally, define a sequence of distinct students $t_1, t_2, \ldots$ in S^* as follows: for $i \geq 1$, having defined t_i and her new assignment $p_i := \hat{\mu}(t_i)$, let t_{i+1} be the earliest student (in the order $\blacktriangleright$) whose assignment under μ equals p_i but who cannot be assigned to p_i in the execution on $S^* \cup \{s\}$ due to the seat taken by t_i. If no such student exists, the sequence terminates. By construction, the mechanism differs from μ only along this single chain: each t_i takes (in the enlarged instance) a seat that was previously used (in the restricted instance) by t_{i+1}, and all other students in S^* keep their original assignment. Since the chain is triggered by the insertion of a single student s, the chain can terminate in one of two ways: (i) some t_i is reassigned to a program that had an empty seat in the restricted instance and no student becomes unmatched, or (ii) the last student in the chain has no acceptable program with remaining capacity and becomes unmatched. In either case, at most one student can end up unmatched and thus $u \leq 1$. Since $\blacktriangleright \in \mathcal{R}(S^* \cup \{s\})$ was arbitrary, the result holds for every exit ranking. □

Since declared lists determine both admission and subsequent articulation, applicants may have an incentive to misreport them. Let each student s have a true strict preference $\succ_s$ over $P \cup \{\emptyset\}$, where $\emptyset$ denotes not being admitted, and call her report truthful if it lists the programs she prefers to $\emptyset$ in that order, truncated to its first k entries when lists are limited to k programs. To compare reports, we regard the exit ranking as an exogenous total order on S, restricted to the admitted set that each report profile produces. The mechanism is strategy-proof if, for every student, every true preference, every profile of reports of the other students, and every exit ranking, truthful reporting yields an outcome at least as good as any other report. The following example shows that this fails, through a single deviation that is strictly profitable under every exit ranking.

Example 7. *Let $P = \{A, B, C, D\}$, all programs with one slot, and let the entrance ranking be $1, 2, 3, 4$, from highest to lowest priority, with lists $A \rhd_1 B \rhd_1 D$, $A \rhd_2 B \rhd_2 D$, and $C \rhd_3 A \rhd_3 B$. Student 4 has true preference $A \succ_4 B \succ_4 C \succ_4 D \succ_4 \emptyset$, so her truthful list of length three is $A \rhd_4 B \rhd_4 C$. The set $\{1, 2, 3\}$ is robustly compatible: student 3 always*

receives C, and students 1 and 2 receive A and B in some order. Under the truthful report, the exit ranking $1, 2, 3, 4$ leaves student 4 unmatched, so she is rejected. If instead she declares $A \rhd_4 B \rhd_4 D$, student 3 still receives C and students 1, 2, and 4 receive A, B, and D in some order, so $\{1, 2, 3, 4\}$ is robustly compatible. Student 4 is then admitted and articulated to A, B, or D, each of which she prefers to being rejected, whatever the exit ranking.

We finally address the computational complexity of deciding robust compatibility. Since all programs share the exit ranking, the articulation mechanism is a serial dictatorship: students are processed in the order given by $\blacktriangleright$, and each one takes her most preferred program with an available slot. Cechlárová et al. (2017, Corollary 3) prove that deciding whether a specified agent can receive no object under some processing order is NP-complete, even when every preference list contains at most two objects and every object has capacity at most two.

Theorem 8. *Deciding whether a given set $S^* \subseteq S$ is robustly compatible is co-NP-complete, even when every student lists exactly three programs and every program has capacity at most two. Hardness holds even if, in addition, a student $s \in S^*$ is specified and $S^* \setminus \{s\}$ is promised to be robustly compatible.*

Proof. If S^* is not robustly compatible, an exit ranking under which some student is left unmatched is a certificate that can be verified in polynomial time by running the articulation mechanism. Hence the problem belongs to co-NP.

For hardness, consider an instance of the problem described above: a set N of agents, a set O of objects with capacities at most two, a preference list of at most two objects for each agent, and a distinguished agent $a \in N$. We first delete any objects of capacity zero and remove them from all lists, without changing any assignment under any processing order. All remaining capacities are therefore one or two. Let $\ell \in \{0, 1, 2\}$ be the length of the resulting list of a, and set $h = 3 - \ell$.

We retain the objects in O as programs with their capacities and the agents in N as students, denoting the distinguished student by $s = a$. For each $i \in N \setminus \{a\}$, we append new programs, listed by no other student, to her list until it contains exactly three entries. We also introduce programs q_j, r_j, r'_j and students g_j, for $j = 1, \ldots, h$, with preferences $q_j \rhd_{g_j} r_j \rhd_{g_j} r'_j$. The list of s is the resulting list of a followed by $q_1, \ldots, q_h$, in this order. All newly introduced programs are distinct and have capacity one. Thus, every student lists exactly three programs.

Let $S' = (N \setminus \{a\}) \cup \{g_1, \ldots, g_h\}$ and $S^* = S' \cup \{s\}$. Every student in S' lists a program that no other student, including s, lists. Hence every student in S' is matched under every

exit ranking, both with and without s. In particular, S' is robustly compatible, and s is the only student who can be left unmatched in S^*.

Observe that, under any exit ranking of S^*, the students in N receive within O exactly the objects they receive in the original instance under the restriction of the ranking to N. Indeed, the students $g_1, \ldots, g_h$ never consume capacity in O, and every student in N ranks the objects in her original list above all programs appended to that list. The observation follows by induction along the restricted processing order.

Suppose that a receives no object in the original instance under an order σ of N. Consider the exit ranking $g_1, \ldots, g_h, \sigma$. Each g_j takes q_j, and the students in N receive within O the same assignments as under σ. Consequently, when $s = a$ is processed, all programs in her list are full, so she is left unmatched.

Conversely, suppose that some exit ranking of S^* leaves a student unmatched. This student must be $s = a$; in particular, she receives no object in O. By the preceding observation, a receives no object in the original instance under the restriction of the ranking to N.

Thus, S^* is not robustly compatible if and only if a can receive no object in the original instance, and the construction is polynomial. □

5 An optimization problem for robust compatibility

Consider the second-stage problem over students $S^* \subseteq S$ and exit ranking $\blacktriangleright \in \mathcal{R}(S^*)$. For each admissible pair $(s, p) \in A(S^*)$, let x_{sp} be a binary decision variable indicating whether student s is assigned to program p. Let $X(S^*)$ be the set of vectors x that represent an S^*-matching, that is, that satisfy

$$\sum_{p \in P_s} x_{sp} \leq 1 \quad \forall s \in S^* \tag{1}$$

$$\sum_{s \in S_p^*} x_{sp} \leq c_p \quad \forall p \in P \tag{2}$$

$$x_{sp} \in \{0, 1\} \quad \forall (s, p) \in A(S^*), \tag{3}$$

where $S_p^* = S_p \cap S^*$. Constraints (1) and (2) ensure each student is assigned to at most one program and program capacities are respected, respectively, while (3) impose binary constraints. Note that solutions in $X(S^*)$ may not necessarily respect the students' preference order. To enforce this, it is enough to include the stability constraints from an IP formulation of the many-to-one stable matching problem. Following Baïou and Balinski (2000), we

consider

$$c_p x_{sp} + c_p \sum_{q \in P_s : q \rhd_s p} x_{sq} + \sum_{t \in S_p^* : t \blacktriangleright s} x_{tp} \geq c_p \quad \forall (s,p) \in A(S^*). \tag{4}$$

Constraints (4) ensure, for each admissible pair (s,p), that either s is assigned to p or to a more preferred program according to $\rhd_s$, or p is full with higher-ranked students according to $\blacktriangleright$. Since the ranking $\blacktriangleright$ is common to all programs in the current setting, by Proposition 2, there is a unique $x^{\blacktriangleright} \in X(S^*)$ satisfying (4).

Remark. When $c_p = 1$ for all $p \in P$, we recover the classical one-to-one or marriage problem. In that case, by relaxing constraints (3) to nonnegativity only, we obtain a polytope whose vertices are binary vectors (Vate (1989); Rothblum (1992)). However, this is no longer the case for general c_p since the polytope might have fractional vertices (Baïou and Balinski (2000)). Of course, as Rothblum (1992) suggests, we can split each program p into c_p copies with a unique available slot each, thus reducing the many-to-one to the one-to-one setting and recovering integrality of extreme points. We are, however, increasing the number of variables and constraints involved in the formulation, which can become rather expensive to solve if the numbers are large.

We introduce a robust formulation by parameterizing the stability constraints. Define a binary vector $z^{\blacktriangleright}$, representing the exit ranking $\blacktriangleright$, with $z^{\blacktriangleright}_{ts} = 1$ if and only if $t \blacktriangleright s$ for $s,t \in S^*,\ s \neq t$. Then, constraints (4) become

$$c_p x_{sp} + c_p \sum_{q \in P_s : q \rhd_s p} x_{sq} + \sum_{t \in S_p^* :\ t \neq s} z^{\blacktriangleright}_{ts} x_{tp} \geq c_p \quad \forall (s,p) \in A(S^*). \tag{5}$$

We now treat $z^{\blacktriangleright}$ as an adversarial decision vector z rather than a fixed parameter. The feasible set $Z(S^*)$ for exit rankings z corresponds to the set of linear orderings on S^*, which is described by

$$z_{ts} + z_{st} = 1 \quad \forall s,t \in S^* :\ s \neq t \tag{6}$$

$$z_{st} + z_{tu} \leq z_{su} + 1 \quad \forall s,t,u \in S^* :\ s \neq t,\ t \neq u,\ u \neq s \tag{7}$$

$$z_{st} \in \{0,1\} \quad \forall s,t \in S^* :\ s \neq t. \tag{8}$$

Constraints (6) and (7) enforce totality, antisymmetry and transitivity, while constraints (8) define the variable domain.

We seek a set $S^* \subseteq S$ that guarantees compatibility across all possible exit rankings

$z \in Z(S^*)$. We verify (in)compatibility by solving the optimization problem

$$\begin{aligned} m(S^*) = \min \quad & \sum_{(s,p)\in A(S^*)} x_{sp} \\ \text{s.t.} \quad & c_p x_{sp} + c_p \sum_{q\in P_s : q \rhd_s p} x_{sq} + \sum_{t\in S^*_p:\ t\neq s} z_{ts}x_{tp} \geq c_p \quad \forall (s,p) \in A(S^*) \qquad (9) \\ & x \in X(S^*) \\ & z \in Z(S^*). \end{aligned}$$

The set S^* is not robustly compatible if and only if $m(S^*) < |S^*|$, indicating that there exists an exit ranking $\blacktriangleright \in \mathcal{R}(S^*)$, represented by some $z^* \in Z(S^*)$, such that the unique matching $x^* = x^*(z^*) \in X(S^*)$ satisfying (9) leaves at least one student not assigned to a program.

To obtain a linear IP, we linearize the bilinear term $z_{ts}x_{tp}$ in (9) using auxiliary variables w_{spt} along with the constraints

$$w_{spt} \leq x_{tp} \quad \forall (s,p) \in A(S^*),\ t \in S^*_p :\ s \neq t \qquad (10)$$

$$w_{spt} \leq z_{ts} \quad \forall (s,p) \in A(S^*),\ t \in S^*_p :\ s \neq t \qquad (11)$$

$$w_{spt} \geq x_{tp} + z_{ts} - 1 \quad \forall (s,p) \in A(S^*),\ t \in S^*_p :\ s \neq t \qquad (12)$$

$$w_{spt} \in \{0,1\} \quad \forall (s,p) \in A(S^*),\ t \in S^*_p :\ s \neq t, \qquad (13)$$

transforming (9) into

$$c_p x_{sp} + c_p \sum_{q\in P_s : q \rhd_s p} x_{sq} + \sum_{t\in S^*_p:\ t\neq s} w_{spt} \geq c_p \quad \forall (s,p) \in A(S^*). \qquad (14)$$

Defining $W(S^*)$ as the solution set of (10)–(14), we have that $m(S^*)$ is given by the optimal value of (RobIP) defined by

$$\begin{aligned} \text{(RobIP)} \qquad \min \quad & \sum_{(s,p)\in A(S^*)} x_{sp} \\ \text{s.t.} \quad & x \in X(S^*) \\ & z \in Z(S^*) \\ & (x,z,w) \in W(S^*). \end{aligned}$$

We slightly modify the formulation of (RobIP) to improve computational performance. First, note that constraints (13) can be relaxed to nonnegativity only and that constraints (12) can be dropped altogether: for fixed binary x and z, the largest value of $\sum_{t\in S^*_p:\ t\neq s} w_{spt}$

allowed by (10)–(11) is precisely $\sum_{t \in S_p^*:\ t \neq s} z_{ts} x_{tp}$, so the set of pairs (x, z) for which (14) holds for some $w \geq 0$ is unchanged. We denote by $W'(S^*)$ the resulting relaxation of $W(S^*)$.

Second, we replace the description (6)–(8) of $Z(S^*)$, which involves a cubic number of transitivity constraints, by a formulation with quadratically many variables and constraints in which the exit ranking is encoded by placing students into layers. Let $L = \{1, \ldots, |S^*|\}$. For each $s \in S^*$ and $l \in L$, let y_{sl} be a binary variable indicating whether student s is placed in layer l, and consider the constraints

$$z_{st} + z_{ts} \leq 1 \quad \forall s, t \in S^* :\ s \neq t \tag{15}$$

$$\sum_{l \in L} y_{sl} = 1 \quad \forall s \in S^* \tag{16}$$

$$\sum_{l \in L} l\, y_{sl} + 1 \leq \sum_{l \in L} l\, y_{tl} + |S^*|\,(1 - z_{st}) \quad \forall s, t \in S^* :\ s \neq t \tag{17}$$

$$y_{sl} \in \{0, 1\} \quad \forall s \in S^*,\ l \in L. \tag{18}$$

Let $Z'(S^*)$ be the set of vectors (z, y) satisfying (8) and (15)–(18).

To interpret $Z'(S^*)$, given a binary vector z, let D_z be the digraph on S^* having an arc (s, t) whenever $z_{st} = 1$, and, given a binary vector y satisfying (16), let $\lambda(s) = \sum_{l \in L} l\, y_{sl}$ be the layer of student s. Constraints (17) state that $z_{st} = 1$ implies $\lambda(s) < \lambda(t)$, whereas they are vacuous when $z_{st} = 0$ since $\lambda(s) - \lambda(t) + 1 \leq |S^*|$ always holds. Hence, every arc of D_z goes from a lower to a strictly higher layer, and D_z is acyclic. Conversely, if D_z is acyclic, placing each student in the layer given by her position in a topological ordering of D_z yields a vector y satisfying (16)–(18). Therefore, the projection of $Z'(S^*)$ onto the z-space is exactly the set of incidence vectors of acyclic subgraphs of the complete digraph on S^*, whose inclusion-wise maximal elements are precisely the linear orderings in $Z(S^*)$. We also note that (15) is implied by (17) for binary vectors, but not by its linear relaxation, and we keep it to strengthen the latter.

Replacing $Z(S^*)$ and $W(S^*)$ in (RobIP), we obtain

$$\begin{aligned} \text{(RobIP}') \qquad \min \ & \sum_{(s,p) \in A(S^*)} x_{sp} \\ \text{s.t.} \ & x \in X(S^*) \\ & (z, y) \in Z'(S^*) \\ & (x, z, w) \in W'(S^*). \end{aligned}$$

Although $Z'(S^*)$ allows the adversary to choose an arbitrary acyclic digraph instead of a linear ordering, the optimal value is not affected.

Proposition 9. *For any $S^* \subseteq S$, the optimal value of (RobIP′) equals $m(S^*)$.*

Proof. Let $m'(S^*)$ denote the optimal value of (RobIP′). Since every linear ordering of S^* is an acyclic digraph, for any $z \in Z(S^*)$ there exists y such that $(z, y) \in Z'(S^*)$, and thus $m'(S^*) \leq m(S^*)$. Conversely, let (x, z, y, w) be an optimal solution of (RobIP′). As D_z is acyclic, it admits a topological ordering $\blacktriangleright \in \mathcal{R}(S^*)$, whose incidence vector $z^{\blacktriangleright} \in Z(S^*)$ satisfies $z^{\blacktriangleright} \geq z$ componentwise. Since z appears in $W'(S^*)$ only as an upper bound in (11), the vector $(x, z^{\blacktriangleright}, w)$ belongs to $W'(S^*)$ and is therefore feasible for (RobIP) with (13) relaxed and (12) dropped, whose optimal value is $m(S^*)$. Hence, $m'(S^*) = \sum_{s \in S^*} \sum_{p \in P_s} x_{sp} \geq m(S^*)$. □

The proof also shows that any solution of (RobIP′) with objective value below $|S^*|$ yields an explicit exit ranking, namely any topological ordering of D_z, under which S^* is not compatible.

The ranking formulation uses $O(|S^*|^2)$ variables and constraints, compared with the $O(|S^*|^3)$ transitivity constraints (7). On the other hand, constraints (17) are big-M constraints in the spirit of those proposed by Miller et al. (1960) for the traveling salesman problem, and the linear relaxation of $Z'(S^*)$ projected onto the z-space can be weaker than that of (6)–(8), which implies the 3-dicycle inequalities known to be facet-defining for the linear ordering polytope (Grötschel et al., 1985). Nevertheless, in preliminary experiments (RobIP′) was computationally more effective than formulations based on (6)–(7), whether these constraints were included explicitly or generated dynamically. We thus use (RobIP′) for the admission algorithm.

When S^* extends a robustly compatible set by one student, Proposition 6 gives $m(S^*) \in \{|S^*| - 1, |S^*|\}$. For these extension tests, we add

$$\sum_{s \in S^*} \sum_{p \in P_s} x_{sp} \geq |S^*| - 1 \tag{19}$$

to strengthen the LP relaxation and certify rejection as soon as an integer solution of value $|S^*| - 1$ is found. This outperformed treating the test as a feasibility problem.

The acceptance/rejection algorithm (Algorithm 1) identifies a robust subset of students from an initial set $S = \{1, \ldots, n\}$, ordered by their entrance ranking. The algorithm starts with an empty set S^* and sequentially considers each applicant s, computing $m(S^* \cup \{s\})$. If adding applicant s maintains robust compatibility, indicated by $m(S^* \cup \{s\}) = |S^*| + 1$, the applicant is accepted into S^*. Otherwise, she is rejected. This sequential process ensures robustness against any possible exit ranking.

Theorem 8 shows that the test performed at each iteration of Algorithm 1 is intractable in the worst case, and that such instances do arise along the algorithm: if the students in

Algorithm 1 Acceptance/Rejection Algorithm for Robust Admission

Require: Applicants $S = \{1, \ldots, n\}$ ordered by entrance ranking; function $m(\cdot)$ returning the minimum number of matched students over all exit rankings

Ensure: A robustly compatible subset $S^* \subseteq S$

Initialize $S^* \leftarrow \emptyset$
for each applicant $s \in S$ in entrance-ranking order **do**
    Compute $m(S^* \cup \{s\})$
    if $m(S^* \cup \{s\}) = |S^*| + 1$ **then**
        $S^* \leftarrow S^* \cup \{s\}$
    else
        Reject applicant s
    end if
end for
return S^*

S' precede s in the entrance ranking, all of them are accepted and the test for s is precisely the hard instance in the proof. The complementary problem, deciding whether some exit ranking leaves a student unmatched, is NP-complete, and (RobIP$'$) searches precisely for such a ranking. This is what motivates the use of integer programming, in contrast to the polynomial-time algorithms of Cechlárová et al. (2017, Corollary 2) for the case of unit capacities and lists with at most two programs.

We close this section with a comment on the set returned by Algorithm 1. By Proposition 3, robust compatibility is preserved under taking subsets, so Algorithm 1 is the greedy procedure on a hereditary family, and it returns its lexicographically maximal member with respect to the entrance ranking: a student is rejected only if her admission is incompatible with the admission of the higher-ranked students already accepted. This set need not have maximum cardinality. For instance, if programs p, q, and r have capacity one and students 1, 2, and 3 list (p), (p, r), and (p, q), respectively, the algorithm admits student 1 alone, whereas $\{2, 3\}$ is robustly compatible. Seeking a robustly compatible set of maximum cardinality may therefore override the entrance ranking, admitting lower-ranked applicants at the expense of higher-ranked ones, which is the kind of priority violation the mechanism is meant to avoid. Its decision version asks whether a robustly compatible set of cardinality at least k exists. It is co-NP-hard, as the case $k = |S|$ shows. Moreover, unless NP = co-NP, Theorem 8 rules out constructing in polynomial time a rational mixed-integer linear program of poly-

nomial encoding length whose projection onto the admission variables is exactly the family of robustly compatible sets, since fixing these variables would reduce robust compatibility to mixed-integer linear programming feasibility, which belongs to NP. A constraint-generation approach could use (RobIP$'$) to identify and exclude non-robust integer candidate sets. We leave this problem for future research.

6 Numerical experiments

6.1 Case study: *Bachillerato Inicia*

PUC has more than 30,000 undergraduate students and admits them through the centralized system that serves nearly 50 Chilean universities (Bordon and Fu (2015); Ríos et al. (2021); DEMRE (2026)). Over the past 15 years, the university has implemented several admission pathways for students with educational gaps, including students from low socioeconomic backgrounds, graduates of public secondary schools, Indigenous peoples, migrants, and individuals with disabilities, in line with the inclusion programs coordinated by the Chilean Ministry of Education and DEMRE (Ríos et al. (2021); MINEDUC (2026a)). In Chile, as in many Latin American and European countries, upper secondary students choose between an academically oriented track and a technical-professional track that prepares them for the labor market (Salas-Velasco (2024)). In 2024, graduates of the technical-professional track represented 27% of all upper secondary graduates (MINEDUC (2026b)), and a growing share of them aspire to continue into higher education. *Bachillerato Inicia*, created in 2024, is a two-year program aimed exclusively at these graduates, providing access to the regular undergraduate degree programs of PUC.

Admission to the program follows a special process that combines secondary school grade point average with assessments of intra- and interpersonal skills; the resulting score defines the entrance ranking. At the time of application, students declare three programs ordered by preference. Demand is highly skewed, and a few programs attract far more first preferences than their capacity. In the 2026 cycle, 248 applicants reached the final stage of the selection process for 125 places, with 305 articulation places distributed over 51 programs; if the 125 best-ranked applicants had been admitted, their first preferences would have been concentrated in 29 of the 51 programs. Articulation assignments are determined at the end of the first year, using the students' academic performance ranking.

6.2 Data

We consider data from the 2024, 2025, and 2026 admission cycles of *Bachillerato Inicia*. Table 2 reports the size of each instance in terms of the number of applicants, the number of programs, and the total number of available articulation slots. We have that across the three years, the instances exhibit a clear growth in scale. The number of applicants increases substantially from 146 in 2024 to 248 in 2026, while the number of programs grows more moderately, from 40 to 51. In contrast, the total number of slots remains relatively stable across years, increasing only slightly from 273 to 305. This divergence between applicant growth and relatively flat capacity levels anticipates increased congestion and competition in later cohorts.

Table 2: Instances sizes.

Year	Applicants	Programs	Slots
2024	146	40	273
2025	206	50	303
2026	248	51	305

Figure 1 displays, for the 2026 instance, the distribution of applicants' declared preferences across programs, disaggregated by first, second, and third choices, together with the available number of slots per program. Programs are sorted in decreasing order of total demand, revealing a highly skewed demand profile: a small subset of programs attracts a disproportionately large number of applicants, while the majority of programs receive relatively low demand. Moreover, the most demanded programs accumulate substantial mass not only from first choices but also from second and third choices, suggesting that these programs play a central role as both top preferences and fallback options. Conversely, programs in the lower-demand tail tend to receive very few first-choice applications and are mostly supported by lower-ranked preferences, if at all. The red line representing slots shows that capacities are relatively homogeneous compared to demand, and do not scale proportionally with popularity. As a result, several highly demanded programs face demand that substantially exceeds their capacity, while many low-demand programs operate with slack capacity. Years 2024 and 2025 share almost identical patterns (skewed demand profile, mismatch between demand concentration, and relatively flat capacities), and we thus omit their figures.

Figure 2 depicts, for the 2026 year, the program co-occurrence graph induced by applicants' preference lists. Nodes represent programs, and an undirected edge connects two programs

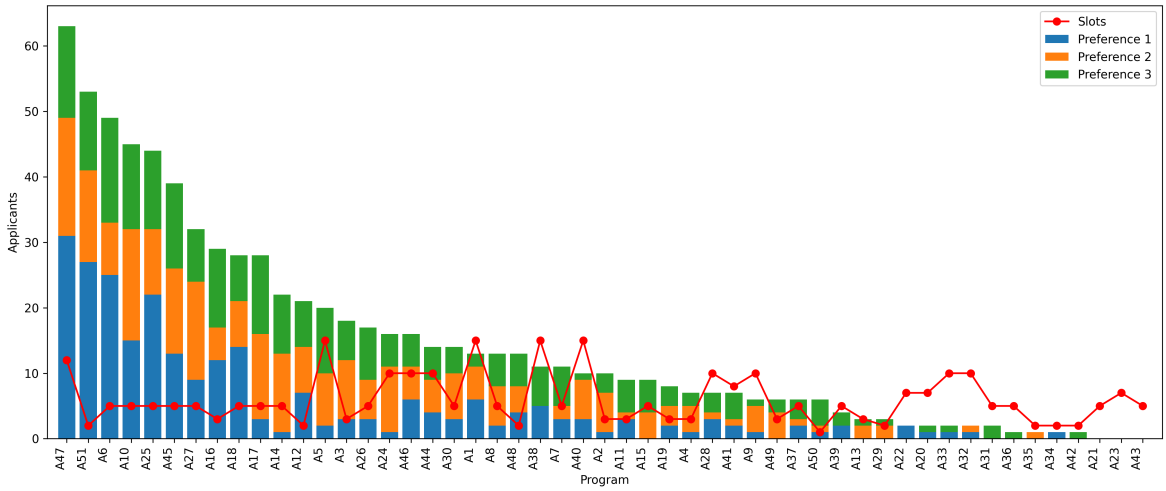

Figure 1: Student preferences and available program slots.

whenever there exists at least one applicant who listed both programs among their preferences. Moreover, edge width is proportional to the frequency with which the connected programs jointly appear in applicants' lists. We observe that the graph exhibits a highly connected structure. Apart from a small number of isolated nodes corresponding to programs that do not appear in applicants' lists, the graph contains a single giant connected component encompassing almost all programs. In particular, there are no nontrivial connected components that would allow the instance to be decomposed into smaller, independent subproblems. This connectivity reflects substantial overlap in applicants' preference sets, indicating that programs compete for applicants not only directly but also indirectly through chains of shared preferences. As a consequence, allocation decisions at one program can propagate through the entire system, affecting the stability of assignments elsewhere. From a computational perspective, this structural property implies that the overall admission problem cannot be simplified via graph-based decomposition techniques in an obvious way and must be solved as a series of increasingly large instances. As with demand, for 2024 and 2025, the overall topology of the co-occurrence graphs remains stable: they also exhibit dense cores with a few peripheral or isolated programs. We thus omit the corresponding figures.

6.3 Results

Algorithm 1 was implemented in Python 3.12, using COPT 8.0.7 as IP solver (Ge et al. (2022)). The computational experiments were run under Linux on a desktop computer with an AMD Ryzen 9 9950X processor and 64 GB of RAM. Table 3 summarizes the main outcomes

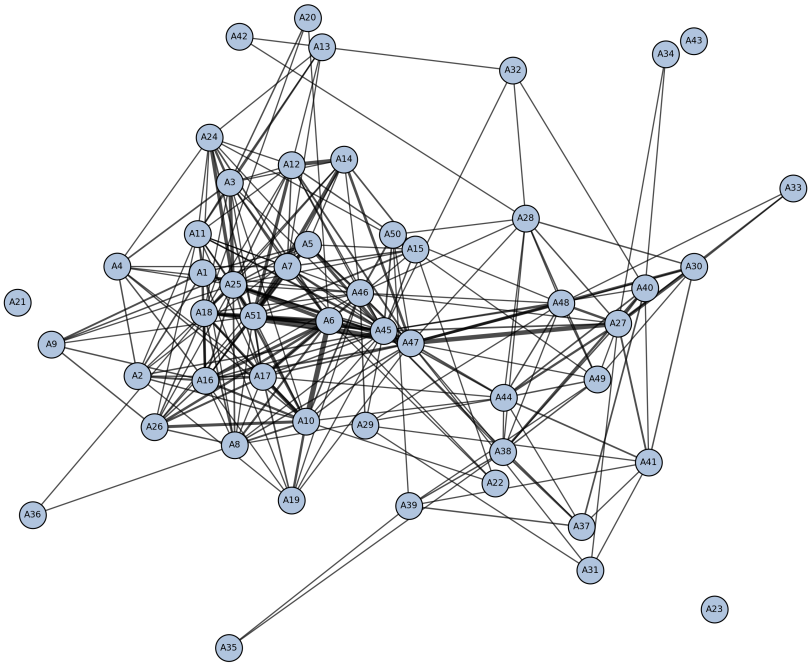

Figure 2: Co-occurrence graph induced by applicants' preference lists.

and computational characteristics of the algorithm for each instance.

Table 3: Summary of results per year.

Year	Applicants	Total accepted	First rejected	90th accepted	Total time (minutes)
2024	146	97	69	129	4
2025	206	127	47	120	390
2026	248	135	61	115	172

The column *Total accepted* reports the final size of the robustly compatible set S^*. The experiments process all applicants without limiting the cohort to the planned intake. Despite the growth in the applicant pool, the number of accepted applicants increases at a slower rate, from 97 in 2024 to 135 in 2026, reflecting tighter compatibility constraints as demand intensifies.

The column *First rejected* indicates the iteration at which the first applicant is rejected. In all instances, this occurs relatively early, well before the end of the process, signaling that incompatibilities arise even when the admitted set is still small. The column 90th accepted reports the iteration at which the 90th applicant is accepted, providing a proxy for the transi-

tion from the initial low-congestion regime to a more saturated phase. Notably, this threshold is reached earlier in the larger instances, indicating that the system enters the high-congestion regime sooner as the market grows.

Finally, *Total time* reports the overall running time of the algorithm in minutes. Computational effort increases dramatically with instance size and congestion. While the 2024 instance is solved in a few minutes, the 2025 and 2026 instances require roughly 98 and 43 times as much time, respectively. The non-monotonic relationship between instance size and total time further highlights the sensitivity of the algorithm to structural properties of the data, rather than scale alone.

6.4 Detailed results for 2026

We focus our analysis on the 2026 instance since most trends also appear in the 2024 and 2025 instances.

Figure 3 shows the cumulative number of accepted applicants as a function of the iteration. The dashed line represents the identity function, which would correspond to the hypothetical case in which every tested applicant is accepted. We observe that the acceptance curve initially follows the identity line, indicating that during the early iterations every applicant tested can be feasibly admitted. This regime corresponds to a low-congestion phase in which capacity and robustness constraints are largely nonbinding. As iterations progress, the acceptance curve gradually bends away from the identity line, reflecting an increasing frequency of rejections. This slowdown marks the onset of congestion: as the set of accepted applicants grows, additional applicants are more likely to violate robustness conditions, leading to rejections and a sublinear growth of the number of accepted applicants.

Regarding computational performance, Figure 4 reports the running time (in seconds) of the first 230 iterations, out of 248, of Algorithm 1. Each iteration corresponds to testing the inclusion of a single applicant, and bars are color-coded according to whether the applicant is ultimately accepted (blue) or rejected (orange). Somewhat expected, running times exhibit a clear increasing trend as the algorithm progresses. Early iterations are computationally inexpensive, while later iterations become significantly more costly. This pattern reflects the cumulative nature of the procedure: as more applicants are admitted, the size of (RobIP$'$) grows and verifying that an additional applicant can be included requires solving increasingly complex problems. A salient feature of the results is that accepted applicants systematically require more computational effort than rejected ones, especially in the later stages of the

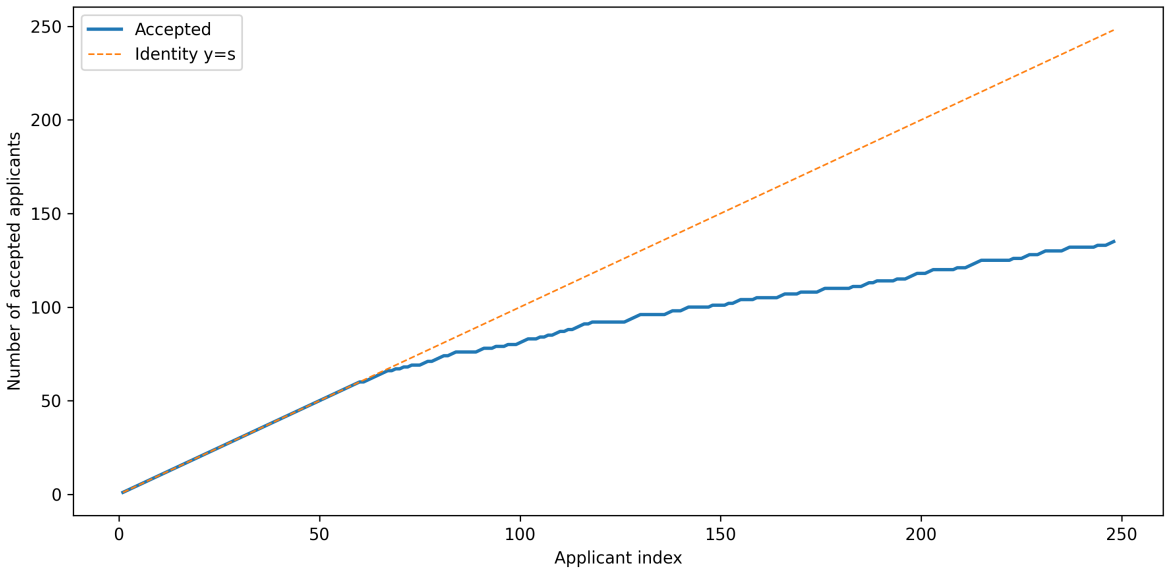

Figure 3: Cumulative number of accepted applicants.

algorithm. Intuitively, acceptance requires certifying that the applicant is compatible with all currently admitted applicants, which involves verifying that all feasible solutions to the IP match all students. In contrast, rejection can often be concluded earlier, once a feasible solution that leaves a student unmatched is identified.

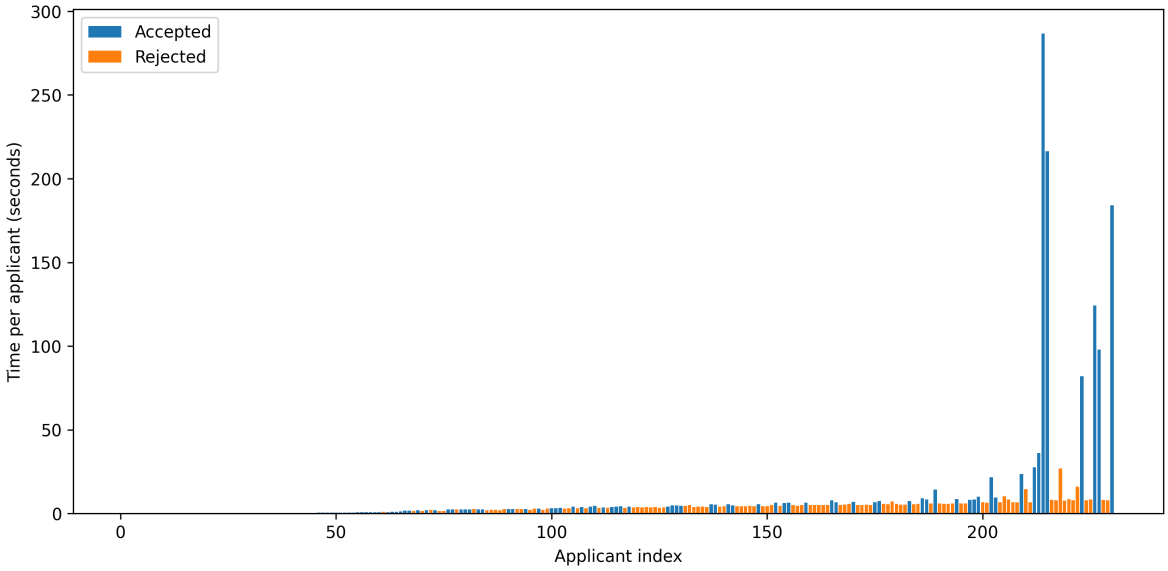

Figure 4: Time spent testing the inclusion of each applicant.

Table 4 reports detailed results for iterations 231-248, corresponding to the final applicants processed. For each applicant, the table shows the acceptance decision, the total running time of the iteration (in seconds), and the number of branch-and-bound nodes explored by

the solver. Rejected applicants require 8–18 seconds and at most 3,326 branch-and-bound nodes. Accepted applicants require 116–3,680 seconds and 74,659–2,473,435 nodes. Late acceptance decisions therefore dominate computational effort, reflecting the need to certify complete assignment under every exit ranking.

Table 4: Detailed results of iterations 231-248.

Applicant	Status	Time (seconds)	Nodes
231	Accepted	326	213,370
232–235	Rejected	8–14	$\leq 1,854$
236	Accepted	116	74,659
237	Accepted	775	474,948
238–243	Rejected	9–18	$\leq 3,326$
244	Accepted	931	591,469
245–246	Rejected	11	≤ 544
247	Accepted	3,680	2,473,435
248	Accepted	2,438	958,377

Finally, Figure 5 compares the optimal value of the LP relaxation of (RobIP$'$) without the lower-bounding constraint (19) with the optimal IP value at each iteration. The blue bars represent the LP optimum, while the orange bars show the remaining integrality gap. We observe that, without additional constraints, the LP relaxation becomes increasingly loose as the algorithm progresses, leading to a growing gap in later iterations. However, by Proposition 6, when the lower bound (19) is enforced, the LP objective differs from the IP optimum by at most one unit at every iteration. In the absence of this lower bound, the IP solver would need to generate additional cutting planes and explore substantially larger branch-and-bound trees to close the gap, significantly increasing computational effort in iterations that lead to rejection.

6.5 Augmenting preference lists and capacity

We consider two auxiliary procedures used to generate instances that reflect sensible actions that *Bachillerato Inicia* might take to increase the number and entrance ranking of admitted applicants. The first procedure (Algorithm 2) augments applicants' preference lists by adding a fourth option based on empirical correlation patterns across programs. The second procedure (Algorithm 3) constructs alternative capacity vectors through a saturation-based

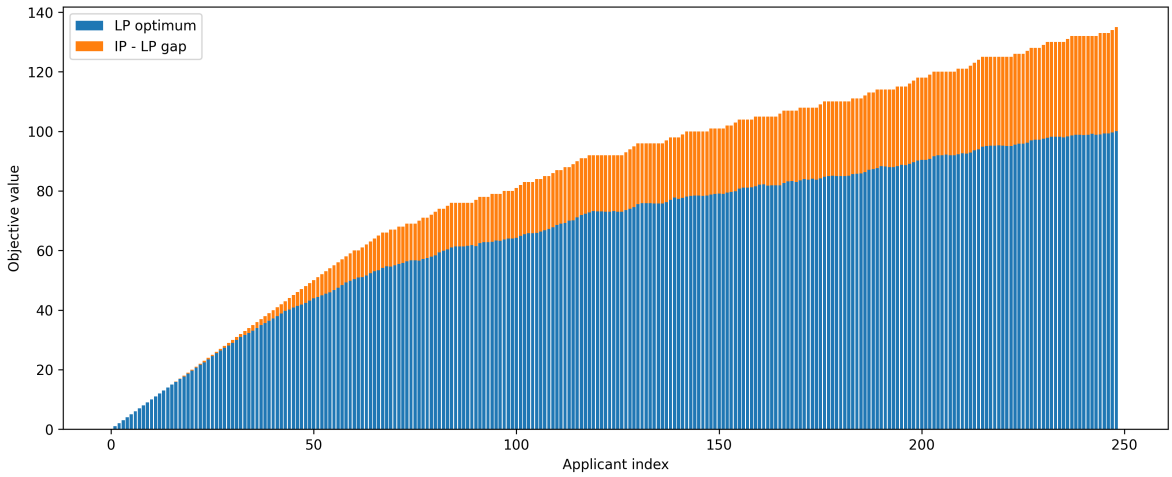

Figure 5: Comparison of LP relaxation and IP optimal value.

Algorithm 2 Correlation-based preference augmentation

Require: Preference lists $(P_s)_{s \in S}$, weighted graph $G = (P, E, w)$

Ensure: Augmented preference lists $(\tilde{P}_s)_{s \in S}$

for each applicant $s \in S$ **do**
  for each $p \in P \setminus P_s$ **do**
    compute $\text{score}_s(p) = \sum_{q \in P_s : \{p,q\} \in E} w_{pq}$
  end for
  select $p_4 \in \arg\max_p \text{score}_s(p)$
  set $\tilde{P}_s = (P_s, p_4)$
end for

allocation rule that concentrates additional slots on the most constrained programs.

Correlation-based fourth preference In order to construct augmented preference profiles, we assign a fourth program to each applicant using aggregate information on the similarity between programs. This information is summarized by a weighted, undirected graph on the set of programs, denoted by $G = (P, E, w)$, which is constructed from application data and illustrated in Figure 2. The graph captures empirical patterns of joint demand across programs and is interpreted as a measure of proximity or relatedness between academic options. Using this graph as input, the fourth program assigned to an applicant is chosen so as to be most closely related, in aggregate, to her originally declared choices, while excluding the programs already listed in her preference set.

Algorithm 3 Saturation-based capacity expansion

Require: Capacities $(c_p)_{p \in P}$, demand $(d_p)_{p \in P}$, budget k

Ensure: Capacity increments $(\Delta_p)_{p \in P}$

$c^{(0)} \leftarrow c$

for $t = 1$ to k **do**

  choose $p^\star \in \arg\max_{p \in P} \frac{d_p}{c_p^{(t-1)}}$

  $c_{p^\star}^{(t)} \leftarrow c_{p^\star}^{(t-1)} + 1$

end for

$\Delta_p \leftarrow c_p^{(k)} - c_p \ \ \forall p \in P$

Saturation-based capacity increments To construct alternative capacity scenarios, we consider a rule that allocates additional slots to programs as a function of their overall demand pressure. For each program $p \in P$, let d_p denote its total demand, defined as the number of applicants who list p anywhere in their preference profile. Let c_p be the original capacity of program p, and let $k \in \mathbb{Z}_+$ denote a fixed budget of additional slots to be distributed. Starting from the original capacity vector, additional slots are assigned sequentially to programs that exhibit the highest level of saturation, measured as a ratio between demand and current capacity. At each step, one unit of capacity is allocated to the program with the largest demand-to-capacity ratio, and the capacity vector is updated accordingly before proceeding to the next allocation.

We built the additional instances based on the 2026 dataset, restricting executions of Algorithm 1 to the first 230 applicants for computational tractability. Table 5 reports summary statistics analogous to those in Table 3, considering the baseline instance, a variant in which applicants are allowed to declare a fourth program, and scenarios in which articulation capacities are increased by $k = 10, 20, 30$ slots in saturated programs. Figure 6 visualizes the acceptance outcomes across these scenarios. Each row corresponds to one execution, and each applicant is marked as accepted or rejected, allowing a direct comparison of acceptance patterns across instances.

The results show that, in this instance, allowing a fourth declared program has no effect on admission outcomes. However, this modification substantially increases total running time, indicating that, under correlated preferences, additional declared options increase computational complexity without improving admission performance. In contrast, increasing articulation capacity has a clear and systematic effect. As k grows, the first rejection is delayed, and the algorithm reaches the 90th accepted applicant earlier, reflecting reduced congestion in the

Table 5: Summary of results per instance.

Instance	Applicants	Total accepted	First rejected	90th accepted	Total time (minutes)
Baseline	230	129	61	115	32
Fourth program	230	129	61	115	1,327
$k = 10$	230	128	72	113	21
$k = 20$	230	136	82	103	349
$k = 30$	230	142	86	100	490

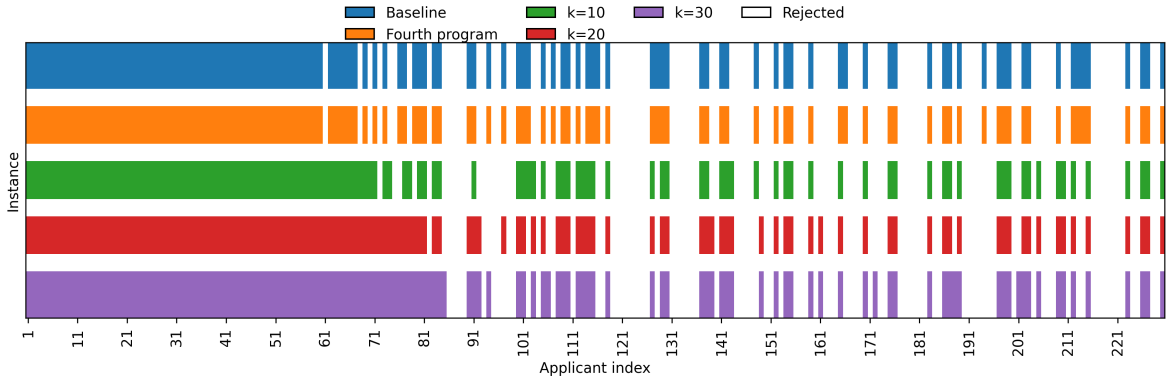

Figure 6: Accepted applicants per instance.

articulation stage. At the same time, the total number of accepted applicants increases from 129 in the baseline to 142 for $k = 30$. This improvement comes at a significant computational cost, as total running time grows rapidly with k, exceeding several hundred minutes for larger capacity increases.

A closer inspection of Figure 6 also reveals that acceptance sets are not nested across scenarios. In particular, focusing on applicants in the range 90-100, we observe that the sets of accepted applicants differ in a non-monotone way: some applicants accepted in one execution may be rejected in another, and vice versa. This behavior reflects the path-dependent nature of Algorithm 1. Once two executions diverge at a given applicant, subsequent compatibility tests are performed against different admitted sets, which may trigger further divergences downstream. Importantly, this observation is not in conflict with Propositions 4 and 5. Both results are comparative statics statements that apply to a fixed robustly compatible set S^*: appending programs to preference lists or increasing capacity cannot invalidate robustness of an already robust set. In contrast, the algorithm constructs S^* sequentially, and different parameter choices may lead the algorithm to select different robustly compatible sets, none of which need to be nested. Thus, non-containment across executions is an algorithmic

phenomenon rather than a violation of robustness monotonicity.

Taken together, Table 5 and Figure 6 suggest that while expanding preference lists offers no practical benefit, moderate increases in articulation capacity at saturated programs can meaningfully improve admission outcomes. From a practical perspective, these results indicate that targeted capacity expansions may allow *Bachillerato Inicia* to admit more and higher-ranked students, albeit at the expense of increased computational effort.

7 Implementation experience

The enrollment process for inclusion and special access programs at Chilean universities is conducted within a highly constrained time window, typically lasting between two and three days after the official publication of the data reflecting students' performance in secondary education and national examinations. *Bachillerato Inicia* extends enrollment offers concurrently with other inclusion programs at PUC and with academic programs at other universities. Consequently, students may be preselected and receive enrollment offers from multiple programs. Most academic programs in Chile do not employ algorithmic optimization in the final stage of selection. Instead, enrollment offers are issued strictly in accordance with the entrance ranking.

A nontrivial fraction of students accepted by the algorithm into *Bachillerato Inicia* decline the enrollment offer. These rejections create vacancies, allowing admission of students who were not selected in the initial round. Prior to extending offers to students lower in the entrance ranking, it is preferable to update the candidate list by removing students who have declined enrollment. This adjustment allows higher-ranked students to be reconsidered and potentially selected in subsequent rounds. In practice, the algorithm is therefore executed multiple times. The short enrollment window, combined with the need for repeated re-optimization, imposes significant constraints on the computational performance of the algorithm and emphasizes minimizing execution time.

In the 2026 admission cycle, the algorithm was executed in three rounds. In the first round, the algorithm considered the top 150 students in the entrance ranking, of whom 101 were selected. In the second round, students who declined enrollment in the first round were removed, and candidates ranked 151-248 were included. In this round, the algorithm accepted 37 additional students, including two students who had not been selected in the first round. In the third round, after additional removals due to enrollment rejections, the algorithm selected nine students who had not been previously selected. The enrollment rejection rate remained

constant across rounds, at approximately 23%. The final enrolled cohort for the program in the 2026 admission cycle consisted of 113 students. Note that by Proposition 3, admitted students who accept the offer remain robustly compatible, and they provide the initialization of S^* in Algorithm 1 for subsequent rounds.

The model developed in this study can be implemented in two different ways. In the first case, a strict implementation guarantees that all selected students are assigned to one of the three preferences declared at the time of application, based on their performance ranking at the end of the first year. Under this strict implementation, students who wish to access a program different from their initially declared preferences lose priority in the ranking and may only access their preferred program if places remain available after all other assignments have been completed. Again, by Proposition 3, the students who do not change their lists remain robustly compatible.

A second implementation of the model allows students to engage in vocational exploration during the first year and to change their program preferences without losing their position in the exit ranking. The robustness guarantee applies only to the original preference profile. If preference changes are relatively limited and there are unused places, the number of students who fail to access their preferred programs at exit may be relatively small.

In *Bachillerato Inicia*, the model was applied in an almost strict manner for the 2024 cohort and with preference changes allowed for the 2025 cohort. In the 2024 cohort, 3% of students changed their first preference, and 92% were assigned to their first preference at the end of the first year. In the 2025 cohort, 19% of students changed their first preference, and 95% were assigned to their first preference at the end of the first year. For the *Bachillerato Inicia* program, the implementation of the model that allows changes in preferences introduces greater flexibility in students' options, without a significant loss in first-preference assignment outcomes.

We also highlight that the first-year retention rate of *Bachillerato Inicia* was 93% in years 2024 and 2025. This sharply contrasts with the situation of similar programs in Chile, with retention rates of approximately 40%, and also compares favorably with the 84% for the Chilean university system as a whole (MINEDUC (2026c)). For the 2026 cohort, if the first 125 applicants were admitted, by using (RobIP$'$) without (19), we find that up to 28 students could be left unmatched in the worst case. By avoiding such situations, *Bachillerato Inicia* is likely to keep high retention rates.

Finally, we address the use of a fully robust model. For the 2025 cohort, Figure 7 compares

the rankings for the 80 students who completed the first year, with both orders restricted to these students. The median absolute displacement is 16 positions and its 90th percentile is 43 positions; only three of the ten highest-ranked students at entrance are among the top ten at exit. The right panel reports the share of students whose displacement is at most k. An uncertainty set requiring every student's displacement to be at most k would need $k \geq 71$ to contain the realized ranking. Bounds of $k = 20$ and $k = 40$ are violated by 37.5% and 12.5% of the students, respectively, and therefore exclude that realization. These observations document substantial reordering and motivate allowing broad ranking uncertainty. Other uncertainty sets may still be appropriate, but their predictive coverage would require validation using additional cohorts.

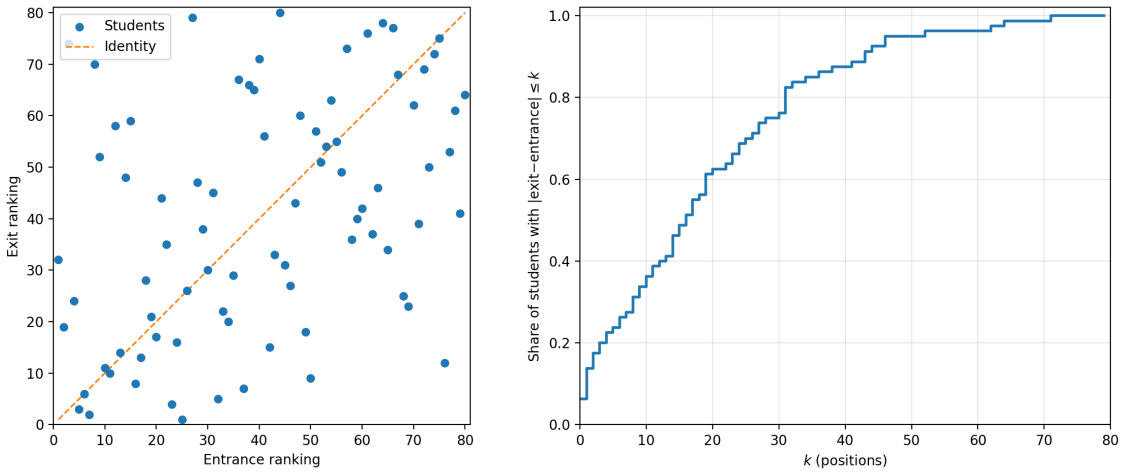

Figure 7: Entrance versus exit rankings for the 2025 cohort.

8 Conclusions

This paper studies a two-stage admission and assignment problem under uncertainty, where admission decisions must remain feasible for all possible exit orders of admitted applicants. We formalize this requirement through a notion of robust compatibility and derive an adversarial formulation, based on an integer programming model of stable matching, to test robustness under worst-case exit rankings. This formulation leads to a sequential admission algorithm that incrementally constructs a robustly compatible set of applicants. Computational results on real admission data show that robustness could come at a significant computational cost, with total running time driven by a small number of late acceptance decisions in congested regimes. The implementation of the mechanism has led to high first-year retention in

Bachillerato Inicia.

There are several avenues for future research. From the stable matching theory perspective, it would be interesting to extend the problem to the general many-to-one case by allowing programs to have different preference profiles over students, or to specialize the results to the classical stable marriage problem. In either case, properties of robustly compatible sets should be revisited. From the algorithmic point of view, the large computing times exhibited by some iterations call for improving the IP formulation, for instance, by carrying out a polyhedral study to obtain facets of the convex hull of feasible solutions of (RobIP). Future work could study restricted uncertainty sets, inspired by the d-robustness of Chen et al. (2021), and assess their predictive coverage using additional cohorts.

Acknowledgements. The authors sincerely thank the referee for their careful reading of the manuscript and constructive comments, which have helped considerably improve its quality. The authors also acknowledge the use of ChatGPT and Claude to process data and results, streamline proofs, and assist with writing. The authors take full responsibility for the content of the article.

Declarations

Conflict of Interest

All the authors declare that they have no conflict of interest.

References

Abdulkadiroğlu, A. and Sönmez, T. (2003). School choice: A mechanism design approach. *American Economic Review*, 93(3):729–747.

Afacan, M. O. (2012). Group robust stability in matching markets. *Games and Economic Behavior*, 74(1):394–398.

Aziz, H., Biró, P., Gaspers, S., de Haan, R., Mattei, N., and Rastegari, B. (2020). Stable matching with uncertain linear preferences. *Algorithmica*, 82:1410–1433.

Baïou, M. and Balinski, M. (2000). The stable admissions polytope. *Mathematical Programming*, 87:427–439.

Bikhchandani, S. (2017). Stability with one-sided incomplete information. *Journal of Economic Theory*, 168:372–399.

Bordon, P. and Fu, C. (2015). College-major choice to college-then-major choice. *The Review of Economic Studies*, 82(4):1247–1288.

Bredereck, R., Chen, J., Knop, D., Luo, J., and Niedermeier, R. (2020). Adapting stable matchings to evolving preferences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1830–1837.

Cechlárová, K., Fleiner, T., and Schlotter, I. (2017). Possible and necessary allocations under serial dictatorship with incomplete preference lists. In Rothe, J., editor, *Algorithmic Decision Theory (ADT 2017)*, volume 10576 of *Lecture Notes in Computer Science*, pages 300–314, Cham. Springer.

Chakraborty, A., Citanna, A., and Ostrovsky, M. (2010). Two-sided matching with interdependent values. *Journal of Economic Theory*, 145(1):85–105.

Chen, J., Skowron, P., and Sorge, M. (2021). Matchings under preferences: Strength of stability and tradeoffs. *ACM Transactions on Economics and Computation*, 9(4):20:1–20:55.

DEMRE (2026). Departamento de Evaluación, Medición y Registro Educacional. `https://www.demre.cl`. Accessed: January 2026.

Ehlers, L. and Massó, J. (2015). Matching markets under (in) complete information. *Journal of Economic Theory*, 157:295–314.

Faenza, Y., Foussoul, A., and He, C. (2024). Two-stage stochastic stable matching. In *Integer Programming and Combinatorial Optimization (IPCO 2024)*, pages 154–167. Springer.

Faenza, Y., Foussoul, A., and He, C. (2025). Minimum cut representability of stable matching problems. *arXiv preprint arXiv:2504.04577*.

Gajulapalli, K., Liu, J. A., Mai, T., and Vazirani, V. V. (2020). Stability-preserving, time-efficient mechanisms for school choice in two rounds. In *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Gale, D. and Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15.

Gangam, R., Mai, T., Raju, N., and Vazirani, V. V. (2026). Robust stable matchings: Dealing with changes in preferences. *arXiv preprint arXiv:2601.07959*.

Ge, D., Huangfu, Q., Wang, Z., Wu, J., and Ye, Y. (2022). Cardinal Optimizer (COPT) user guide. https://guide.coap.online/copt/en-doc.

Genc, B., Siala, M., O'Sullivan, B., and Simonin, G. (2017). Robust stable marriage. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Grötschel, M., Jünger, M., and Reinelt, G. (1985). Facets of the linear ordering polytope. *Mathematical Programming*, 33:43–60.

Irving, R. W., Manlove, D. F., and Scott, S. (2008). The stable marriage problem with master preference lists. *Discrete Applied Mathematics*, 156(15):2959–2977.

Kojima, F. (2011). Robust stability in matching markets. *Theoretical Economics*, 6(2):257–267.

Liu, Q., Mailath, G. J., Postlewaite, A., and Samuelson, L. (2014). Stable matching with incomplete information. *Econometrica*, 82(2):541–587.

Mai, T. and Vazirani, V. V. (2018). Finding stable matchings that are robust to errors in the input. In *26th Annual European Symposium on Algorithms (ESA 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329.

MINEDUC (2026a). Acceso Educación Superior. `https://www.mifuturo.cl/`. Accessed: January 2026.

MINEDUC (2026b). Datos Abiertos. `https://http://www.datosabiertos.mineduc.cl/`. Accessed: January 2026.

MINEDUC (2026c). Mi Futuro. `https://www.mifuturo.cl/`. Accessed: January 2026.

Ríos, I., Larroucau, T., Parra, G., and Cominetti, R. (2021). Improving the Chilean college admissions system. *Operations Research*, 69(4):1186–1205.

Roth, A. E. (1989). Two-sided matching with incomplete information about others' preferences. *Games and Economic Behavior*, 1(2):191–209.

Roth, A. E. and Sotomayor, M. A. O. (1990). *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*. Econometric Society Monographs. Cambridge University Press, Cambridge.

Rothblum, U. G. (1992). Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54:57–67.

Salas-Velasco, M. (2024). Vocational education and training systems in europe: A cluster analysis. *European Educational Research Journal*, 23(3):434–449.

Vate, J. H. V. (1989). Linear programming brings marital bliss. *Operations Research Letters*, 8(3):147–153.