# On the Complexities of Understanding Matching Mechanisms*

Yannai A. Gonczarowski† Clayton Thomas‡

December 1, 2022

**Abstract**

We study various novel complexity measures for two-sided matching mechanisms, applied to the popular real-world school choice mechanisms of Deferred Acceptance (DA) and Top Trading Cycles (TTC). In contrast to typical bounds in computer science, our metrics are *not* aimed to capture how hard the mechanisms are to compute. Rather, they aim to capture certain aspects of the difficulty of understanding or explaining the mechanisms and their properties.

First, we study a set of questions regarding the complexity of how one agent's report can affect other facets of the mechanism. We measure this complexity as the number of bits required to describe the functions from one agent's report to some other piece of data. Despite structural properties such as strategyproofness and nonbossiness, we show that in both DA and TTC, one agent's report can have a quite complex effect on the final matching. Considering how one agent's report can affect another agent's set of obtainable options, we show that this effect has high complexity for TTC, but low complexity for DA, showing that one agent can only affect another in DA in a quantitatively controlled way.

Second, we study a set of questions about the complexity of communicating various facets of the outcome matching, after calculating it. We work with blackboard protocols, i.e., we ask how many bits the mechanism designer would need to publicly communicate. We find that when there are many more students than schools, it is provably harder to concurrently describe to each student her match in TTC than in DA, giving lower bounds showing the tightness of existing constructions [LL21, AL16]. In contrast, we show that the outcomes of TTC and DA are equally hard to jointly *verify*, and that all agents' sets of obtainable options are equally hard to describe, showcasing ways in which the two mechanisms are comparably complex.

Our results uncover new lenses into how TTC may be more complex than DA. This stands in contrast with recent results under different models [GHT22], emphasizing the richness of the landscape of complexities of matching mechanisms. Our proofs uncover novel structural properties of TTC and DA, which may be of independent interest.

*We thank Nick Arnosti, Ori Heffetz, Jacob Leshno, Irene Lo, Kunal Mittal, and S. Matthew Weinberg for illuminating discussions and helpful feedback.

†Department of Economics and Department of Computer Science, Harvard University — *E-mail*: yannai@gonch.name.

‡Department of Computer Science, Princeton University — *E-mail*: claytont@princeton.edu.

# Contents

# 1 Introduction

School districts in many cities employ school choice mechanisms, i.e., algorithms that produce a matching of students to schools on the basis of students' reported preferences and schools' priorities.[1] In contrast to most of the algorithms people encounter throughout daily life—from search result ranking in Google, through rider-driver matching in ride-sharing, to cryptographic protocols in banking apps—school districts give detailed explanations of how these algorithms work, and expect students and parents to understand them. A vast literature studies the complexity of matching mechanisms through a variety of lenses [Knu76, IL86, Pit89, Pit92, RRVV93, Sub94, IM05, Seg07, CFL14, SS15, Gon14, AL16, AKL17, AG18, GNOR19, Tro19, LL21, KMQ21, GHT22]. Yet, very little attention has been devoted to the rigorous study of their explainability. Since simplicity, trust, and ease of participation are paramount to these mechanisms, explainability should be a first-order concern [RNS21, GHT22].

In this paper, we aim to measure the complexity of *understanding* matching rules. We consider mechanisms matching applicants to institutions, and study two canonical and widely deployed matching rules: Top Trading Cycles [SS74] (henceforth TTC, which prioritizes finding an optimal matching for applicants), and Deferred Acceptance [GS62] (henceforth DA, which prioritizes finding a fair and stable matching). While our tools are inspired by traditional computer science techniques designed to measure the hardness of computing algorithms, our questions are specifically tailored to the task of measuring explainability, and require novel models and definitions to capture the insights we seek. Broadly, we ask two types of questions:

(Question 1) Effects Complexities: Before one applicant's report is known, how complex of an effect could her choice have?

(Question 2) Output Complexities: After all the reports are known, how much communication is required to represent or verify certain information about the output matching?

We view Question 1 as a broad theoretical lens into understandability—after all, if an applicant truly understands the mechanism, she should understand the effect of changing her own report. We view Question 2 as a concrete and practically applicable lens into explainability—it provides formal measures of how hard it is to communicate and justify the outcome.

**Effect Complexities.** To give exposition into Question 1, consider two classical game-theoretic properties of mechanisms: strategyproofness and nonbossiness.

Strategyproofness is a celebrated property that says an applicant $d$ always maximizes her match according to some preference list $P_d$ by submitting list $P_d$ as her report (in standard notation: $f_d(P_d, P_{-d}) \succeq_d^{P_d} f_d(P'_d, P_{-d})$). This is closely related to applicant $d$'s **menu given $P_{-d}$**, which is defined as the set of institutions that $d$ could be matched to under some report, when holding the reports of all the other applicants fixed at $P_{-d}$ (in standard notation: $\{f_d(P'_d, P_{-d}) \mid P'_d \in \mathcal{T}_d\}$). One interpretation (and equivalent definition) of strategyproofness is that applicant $d$'s report $P_d$ can only possibly affect $d$'s own match in a very controlled way: by matching $d$ to her highest-ranked institution on her menu.

Nonbossiness is another canonical property in the mechanism design literature. Nonbossiness says that an applicant can only change the matching if her own match changes (in common notation,

[1] As is customary in many papers, we use the word "priorities" to refer to schools' preferences over the students (or more generally, institutions' preferences over the applicants). This reflects the fact that we assume the priorities are constant and fixed ahead of time (e.g., based on factors such as the distance from the school to a student's residence, or whether the student already has a sibling attending the school).

Table 1: Summary of our results addressing Question 1.

| | TTC | DA |
|---|---|---|
| Complexity of one's effect on one's/another's match | $\widetilde{\Theta}(n)$ By nonbossiness, see Observation 3.2 | $\widetilde{\Theta}(n)$ Corollary of other results, see Corollary 4.12. |
| Complexity of one's effect on the full matching | $\widetilde{\Theta}(n^2)$ By Theorem 3.8. | $\widetilde{\Theta}(n^2)$ For APDA, by [GHT22]. For IPDA, by Theorem 3.11. |
| Complexity of one's effect on another's menu | $\widetilde{\Theta}(n^2)$ By Theorem 4.2. | $\widetilde{\Theta}(n)$ By Theorem 4.4. |

**Notes:** We consider markets matching $n$ applicants and $n$ institutions. Each result bounds the number of bits required to represent the function from one (fixed) applicant's report onto some piece of data; in other words, these results count the log of the number of such functions. Note that $\widetilde{\Omega}(n^2)$ matches the number of bits recording all preferences of all applicants, and is thus as high as possible. The last cell of this table, perhaps the most surprising result in this paper, is highlighted.

if $f_d(P_d, P_{-d}) = f_d(P'_d, P_{-d})$, then $f(P_d, P_{-d}) = f(P'_d, P_{-d})$). One interpretation of nonbossiness is that changing applicant $d$'s report can only possibly affect the overall matching in a very controlled scenario: when that change also affects $d$'s own match.

Both strategyproofness and nonbossiness can be viewed as *qualitative* approaches to answering Question 1: each property restricts the type of effects that an applicant's report can have. TTC is both strategyproof and nonbossy. DA comes in two variants: applicant-proposing DA (henceforth, APDA), which is strategyproof and yet bossy, and institution-proposing DA (henceforth, IPDA), which is non-strategyproof and yet nonbossy.

In the first part of this paper, we take a *quantitative* approach to Question 1: we measure how much one applicant's report can affect various aspects of the mechanism. But how should we formally define this measure? A naïve approach could be to bound the magnitude of one applicant's effect on the matching. However, it's not hard to see that even in very simple mechanisms, changing one applicant's report can change the match of *every* applicant.[2] Thus, we look not at the magnitude, but rather the structure, of one applicant's effect on the matching. In particular, we measure the complexity of representing the function from one (fixed) applicant's report to some other piece of relevant data, i.e., the number of bits required to describe this function. This provides a direct, principled way to address Question 1 in a variety of ways.

To concretely illustrate the framework in this part of the paper, consider the effect of one's

[2]For instance, consider a matching mechanism where each of the applicants $d_1, d_2, d_3, d_4$ whose preferences are

$$d_1 : h_1 \succ h_2 \succ \emptyset \qquad d_2 : h_2 \succ h_3 \succ \emptyset \qquad d_3 : h_3 \succ h_4 \succ \emptyset \qquad d_4 : h_4 \succ h_1 \succ \emptyset$$

is (in order from $d_1$ to $d_4$) permanently matched to her favorite yet-unmatched institution. Observe that if $d_1$ changes her preferences to $h_2 \succ h_1$, then every applicant's match changes. (This mechanism is an instance of serial dictatorship, a special case of TTC, APDA, and IPDA.)

report on another's match, i.e. the mapping from the report of a (fixed) applicant $d_*$ to the match of another (fixed) applicant $d_\dagger$, for some value of $P_{-d_*}$. We measure the complexity of how one's report can affect another's match as (the logarithm of) the number of distinct functions of this form that can arise for some $P_{-d_*}$ (see Row 1 in Table 1). It is not hard to see that nonbossiness and strategyproofness together show that there cannot be many such functions for TTC: For each institution $h$ on $d_*$'s menu, there is a unique matching $\mu$ pairing $d_*$ and $h$, so writing down all pairs $(h, \mu(d_\dagger))$ for each such $h$ and $\mu$ provides a way to describe this function with $\widetilde{O}(n)$ bits. It's also clear that this complexity is at most $\widetilde{O}(n^2)$ bits, as this is the number of bits it takes to describe $P_{-d_*}$, which suffices to describe the above function (as well as all of those that we consider below). In contrast to TTC, for the bossy mechanism APDA, or the non-strategyproof mechanism IPDA, it is initially far from clear where this complexity lies between these two extremes. (Eventually, we are able to tightly bound these quantities as a corollary of our other results.)

Arguably our most interesting bounds hold for the complexity of representing the function from one (fixed) applicant's report to: (A) the matching of all applicants (Section 3; see Row 2 in Table 1), or (B) another (fixed) applicant's menu (Section 4; see Row 3 in Table 1). For TTC, IPDA, and APDA, the complexity of (A) is nearly as high as possible ($\Omega(n^2)$ bits, as explained above), and likewise for TTC regarding quantity (B). When we prove these lower bounds, our techniques are reminiscent of data-structure lower bounds and counting arguments: we carefully craft a large set of inputs such that each distinct input defines a distinct function.

For quantity (B) in DA, we uncover a novel and interesting structure behind how one applicant's report can affect another applicant's menu. This structure gives an $\widetilde{O}(n)$-bit upper bound, and we exploit it to provide additional characterizations. These results give a contrast between TTC and APDA, which is particularly surprising in view of the fact that only TTC is nonbossy: despite the fact that TTC is far more restricted regarding when an applicant's report can affect the mechanism, there is a far more quantitatively complex way that one applicant's report can affect another applicant's set of obtainable institutions.

**Output Complexities.** In the second part of our paper, we address Question 2, and consider the communication complexity of representing or verifying the outputs of matching mechanisms. These bounds are in some ways closer to traditional computer science questions that those under Question 1, however we stress that we are not literally concerned with the complexity of computing the matching. Rather, we focus on formalizing and efficiently answering questions that might commonly occur in real-world school choice settings, such as "what is the matching and why?"

To give exposition into Question 2, we recall a property of DA that is well-known from prior work (e.g., [AL16]). Denote the set of applicants by $\mathcal{A}$ and institutions by $\mathcal{I}$. Suppose applicants' priorities at institutions are represented by scores in $[0, 1]$, where a higher score indicates higher priority, and each applicant knows her priority score at each institution (either because the priority scores are communicated to everyone ahead of time, or because these scores are determined by set policies that are detailed on each individual school's website). Then, for any stable matching $\mu$ that could possibly result from DA, there exist *cutoffs* $c_h \in [0, 1]$, one for each institutions $h$, such that each applicant is matched in $\mu$ to her highest-ranked institution $h$ whose cutoff is below her priority score.[3] The mechanism designer could realistically post these cutoff scores on a blackboard communication channel after the mechanism is run, and (when combined with applicants' knowledge of their own preferences as well as their priorities at institutions) these cutoffs would inform every applicant of her own match. This gives a precise sense in which all applicants' matches in DA can

[3]This representation of the matching is just a simple reformulation of the definition of a stable matching; see Observation 5.3.

Table 2: Summary of our results addressing Question 2.

| | TTC | DA |
|---|---|---|
| Concurrently describing to every applicant her own match | $\widetilde{\Theta}\big(\min(\lvert\mathcal{A}\rvert, \lvert\mathcal{I}\rvert^2)\big)$ / $\widetilde{\Theta}(n)$<br>By [LL21] and Theorem 5.5. | $\widetilde{\Theta}\big(\lvert\mathcal{I}\rvert\big)$ / $\widetilde{\Theta}(n)$<br>For any stable matching, by [AL16] / Observation 5.4. |
| In addition to the above, jointly verifying the matching | $\widetilde{\Theta}\big(\lvert\mathcal{A}\rvert\big)$ / $\widetilde{\Theta}(n)$<br>By deterministic complexity, see Observation 5.10. | $\widetilde{\Theta}\big(\lvert\mathcal{A}\rvert\big)$ / $\widetilde{\Theta}(n)$<br>For both APDA and IPDA, by Theorem 5.11. |
| Describing simultaneously all applicants' menus | $\Omega\big(\lvert\mathcal{I}\rvert^2\big)$ / $\widetilde{\Theta}(n^2)$<br>By Theorem 6.5. | $\Omega\big(\lvert\mathcal{I}\rvert^2\big)$ / $\widetilde{\Theta}(n^2)$<br>By Theorem 6.3. |
| Describing all applicants' effects on one's match | $\Omega\big(\lvert\mathcal{I}\rvert^2\big)$ / $\widetilde{\Theta}(n^2)$<br>Even for serial dictatorship, by Theorem 6.8. | |

**Notes:** Each result bounds the number of bits required to simultaneously convey certain information to, or performing a verification task with, all applicants simultaneously. A single message must be posted on a common black blackboard The set of applicants is $\mathcal{A}$ and the set of institutions is $\mathcal{I}$. For ease of exposition, we display our results both in the general, unbalanced case (i.e., for any $|\mathcal{A}| \geq |\mathcal{I}|$) and in the balanced case where $n = |\mathcal{A}| = |\mathcal{I}|$. For bounds where one of these two cases is trivial, or simply extends a lower bound from the balanced case to the unbalanced case, we display this case in grey.

be conveyed using communication proportional to the number of institutions, i.e., using $\widetilde{O}(|\mathcal{I}|)$ bits.

There are two important things to note regarding the above representation. First, it is very natural to have $|\mathcal{A}| \gg |\mathcal{I}|$ (for example, in student-school matchings where many students will attend the same school), and thus the above representation meaningfully saves communication over the trivial solution of fully communicating all matches using $\widetilde{\Theta}(|\mathcal{A}|)$ bits. Second, while the cutoffs can be communicated with $\widetilde{O}(|\mathcal{I}|)$ bits, the protocol crucially relies on applicants knowing far more information than this. Namely, each applicant knows her priority score at all $|\mathcal{I}|$ institutions, requiring a total of $\widetilde{\Theta}(|\mathcal{A}| \cdot |\mathcal{I}|)$ bits of information to be known by the applicants ahead of time. Our model in Section 5.1 accounts for both of these factors.

The first instantiation of Question 2 that we investigate in this model is: how many bits are required to represent to each applicant her own match? (Section 5.2; see Row 1 in Table 2.) As sketched above, this quantity is $\widetilde{O}(|\mathcal{I}|)$ for DA (and it is not hard to show that this is tight). [LL21] have previously studied a variant of this question for TTC, and showed that there exists a cutoff-like representation of the matching, but one requiring a cutoff for each *pair* of institutions. This gives an $\widetilde{O}(|\mathcal{I}|^2)$-bit upper bound for TTC. Our main contribution in Section 5.2 is a nearly matching lower bound: when $|\mathcal{A}|$ is sufficiently large relative to $|\mathcal{I}|$ (say, if $|\mathcal{A}| = |\mathcal{I}|^3$) then representing the matching *requires* $\Omega(|\mathcal{I}|^2)$ bits. Note, however, that if $|\mathcal{A}| < |\mathcal{I}|^2$, then it is always more communication-efficient to simply write down the match of each applicant separately; this gets our final tight bound of $\Theta\big(\min(|\mathcal{A}|, |\mathcal{I}|^2)\big)$ on the complexity of representing the matching in TTC.

To prove this result, we use another carefully crafted lower-bound construction quite akin to our constructions under Question 1.

Our second instantiation of Question 2 builds upon the first, and asks: how many bits are required to additionally *verify* that the matching was calculated correctly? (Section 5.3; see Row 2 in Table 2.) Here, we mean verification in the traditional computer science sense: no incorrect matching can be described without some applicant being able to detect the fact that the matching is incorrect.[4] Despite the fact that TTC is harder to represent than DA, we prove that these two mechanisms are equally difficult to verify, requiring $\widetilde{\Theta}(|\mathcal{A}|)$ bits. In fact, this result is perversely more difficult to show for the easier-to-represent mechanism DA, requiring a somewhat intricate protocol exploiting classical properties of the extremal elements of the set of stable matchings [GI89]. This protocol crucially uses the fact that the priorities are prior knowledge in order to circumvent known lower bounds [GNOR19].

We additionally address a complexity measure that arise as a follow-up to Question 2. We consider the complexity of simultaneously representing all applicant's menus (Section 6.1; see Row 3 in Table 2), and show that it is $\Omega(|\mathcal{I}|^2)$, even for the simple-to-represent DA mechanism.[5] Finally, we study a topic squarely in the intersection of Question 1 and Question 2. We consider the complexity of representing, for all applicants simultaneously, the effect that unilaterally changing the applicant's type can have on a single (fixed) applicant's match (Section 6.2; see Row 4 in Table 2). Interestingly, we show that this complexity is high, even for the serial dictatorship (SD) mechanism, a canonical but extremely simple special case of each of APDA, IPDA, and TTC in which all institutions have the same priority list.

Holistically, our results under both Question 1 and Question 2 formalize ways in which TTC is more complex than DA: one agent can have a more complex effect on another's menu, and (when priorities are common knowledge) the matching is harder to describe. However, we also find that such distinctions are somewhat limited: in both mechanisms, one applicant can have an equally complex effect on the matching overall; the two mechanisms are equally complex to verify; and it is equally hard to describe all applicants' menus together. Out results could be seen as complementing and contrasting recent results under different models [GHT22], which point out ways in which DA is more complex than TTC, uncovering a rich and multi-faceted nature of the landscape of complexities of matching mechanisms.

### 1.1 Related work

Many of our questions on effect complexities (Question 1) are inspired by the recent mechanism design paper [GHT22]. Briefly and informally, [GHT22] are interested in describing mechanisms to participants in terms of their menus, as a way to better convey strategyproofness. In this direction, our Section 3 provides another lens into the relationship between one applicant's menu and the full matching (see Section 3.3 for details), and our Section 4 provides another lens into the relationship between different applicants' menus (see Section 4.3 for details).

Our study of outcome complexities (Question 2) is inspired by the literature on the cutoff structures of DA [AL16] and TTC [LL21]. We formalize some of their insights and prove that they

[4] To quickly see why the above representation of DA in terms of cutoffs *does not* suffice to verify the matching, suppose that the mechanism designer posts cutoffs that are so high that no applicant is matched to any institution. The outcome is certainly not stable. However, no applicant will even be able to tell that some institution is under-subscribed (let alone that the matching is unstable) from only the cutoffs, the priorities, and her own preferences.

[5] We emphasize that an applicant's menu is *not* the same as the set of all institutions where this applicant's priority score (as defined above) is above the cutoff. For example, an applicant's own report can affect the cutoffs (and thus at which institutions she is above the cutoff), but not the menu. For more discussion, see Section 6.1.

are tight.[6] However, we also show that verification of these mechanisms is more delicate than may have been previously believed.

There has also been prior work towards Question 2 in more traditional models where the priorities of the institutions also need to communicated. For example, [Seg07, GNOR19] prove lower bounds of $\Omega(n^2)$ for computing or even verifying a stable matching in this model. Since our protocol in Section 5.3 for DA constructs an $\widetilde{O}(n)$ upper bound for verification (see Row 2 in Table 2), our models are provably quite different. In terms of motivation, the model of [GNOR19] is very specifically focused on the two-sided nature of the question, asking how much communication must occur between a party who knows the preferences and a party who knows the priorities; we focus on a one-sided-uncertainty variant of the question, where priorities are common knowledge and we consider protocols with $n$ separate parties, each knowing a single applicant's preferences.

Menus of selling mechanisms such as auctions are a well-studied object in algorithmic mechanism design, traditionally for single-buyer mechanisms [HN13, DDT17, BGN17, SSW18, Gon18], but also in the pioneering work of [Dob16, DR21] for multi-buyer auctions. In contrast, the study of the complexity of menus for matching mechanisms is still very new [GHT22], and requires asking different questions.[7] Our work also fits into a recent push in the algorithmic mechanism design literature to understand incentive-constrained computation beyond traditional questions regarding computationally efficient truthful mechanisms. This push includes papers studying restricted solutions concepts like dominant-strategy implementations [RST+21, DRV22] or the power of different types of simple mechanisms [BK96, HR09, HN13, HN17, EFF+17a, EFF+17b, BILW20, BGG20, BGN22, and many others].

In various contexts, different works have studied the prospect of changing one applicant's preferences at a time in stable matching markets. [MV18, GMRV18] study matchings that are stable both before and after one applicant changes their list; our upper bound characterization in Section 4.2 may have some conceptual relation to these works (though little technical resemblance). [Kup20] studies the effect of one applicant changing her reported preference to her strategically optimal one under IPDA.

Much work has gone into understanding matching mechanisms under more traditional computer-science questions (such as computational efficiency) or economic questions (such as incentives). [IL86, SS15] show that several problems related to stable matching are #P-hard. [Sub94, CFL14] show that stable matchings are connected to comparator circuits and certain novel complexity classes between NL and P. [BG16, AG18, Tro19, Tho21, MR21] study the strategic simplicity of matching mechanisms through the lens of obvious strategyproofness [Li17, PT21, GL21]. [Pit89, IM05, AKL17, KMQ21, CT22] study the running time of DA under random preferences.

## 2 Preliminaries

This paper studies rules for matching a set of *applicants* $\mathcal{A}$ and a set of *institutions* $\mathcal{I}$. A matching rule is a function $f : \mathcal{T}_1 \times \ldots \times \mathcal{T}_{|\mathcal{A}|} \to \mathcal{M}$, where each set $\mathcal{T}_d$ (mnemonic: the possible *types* of applicant $d$) is the set of all rank-order preference lists over $\mathcal{I}$, and $\mathcal{M}$ is the set of matchings $\mu : \mathcal{A} \to \mathcal{I} \cup \{\emptyset\}$ (where we write $\mu(d) = \emptyset$ if $d \in \mathcal{A}$ is unmatched). Preference lists may be partial: when an applicant does not rank an institution, this indicates that the applicant finds the

[6] [LL21] give a brief argument as to how TTC is formally more complex than DA, but do not prove any type of $\Omega(n^2)$ lower bounds; for a full discussion, see Section C.2.

[7] [Dob16] formulates the *taxation complexity* of mechanisms, which measures the number of distinct menus a bidder can have. This is analogous to the all-type-to-one's-own-match complexity considered in Definition 3.1, which is trivial to bound for strategyproof matching rules. The remainder of our complexity measures, particularly type-to-matching complexity, can be seen as significant generalizations of taxation complexity.

institution unacceptable. We typically write $P_d \in \mathcal{T}_d$, and let $\succ_d^{P_d}$ denote the relation over $\mathcal{I}$ such that $h \succ_d^{P_d} h'$ if and only if $h$ is ranked above $h'$ according to $P_d$. We also write $h \succ_d h'$ where no confusion can arise. For each $d \in \mathcal{A}$, we let $f_d : \mathcal{T}_1 \times \ldots \times \mathcal{T}_n \to \mathcal{I}$ denote the function giving the match of applicant $d$ according to $f$.[8] Throughout Sections 3 and 4, we consider markets with $n$ applicants and $n$ institutions.

In our model, applicant preferences are initially known only to the applicants. Institutions have fixed, commonly known priorities, which we denote by $Q = (Q_h)_{h \in \mathcal{I}}$. Similarly to the applicants, we write $d \succ_h^{Q_h} d'$ or simply $d \succ_h d'$ when $d$ has higher priority at $h$ according to $Q$. Since these priorities are fixed, each of our lower bounds hold for some fixed, worst-case profile of priorities; this fact only makes our lower bound stronger. The upper bounds in Section 5 crucially rely on the fact that each applicant knows the priorities ahead of time, but the upper bound we prove in Section 4.2 does not.

We study three canonical priority-based matching mechanisms. The first is TTC.

**Definition 2.1.** The Top Trading Cycles ($\mathsf{TTC} = \mathsf{TTC}_Q(\cdot)$) mechanism is defined with respect to a profile of priority orders $Q = \{Q_h\}_h$. The matching is produced by repeating the following until every applicant is matched (or has exhausted her preference list): each remaining (i.e., not-yet-matched) applicant points to her favorite remaining institution, and each remaining institution points to its highest-priority remaining applicant. There must be some cycle in this directed graph (as the graph is finite). Pick any such cycle and permanently match each applicant in this cycle to the institution to which she is pointing. These applicants and institutions do not participate in future iterations.

TTC produces a Pareto-optimal matching under the applicants' preferences, i.e. a matching $\mu$ such that no $\mu' \neq \mu$ exists such that $\mu'(d) \succeq_d \mu(d)$ for each $d \in \mathcal{A}$. Despite the fact that the TTC procedure does not specify the order in which cycles are matched, the matching produced by TTC is unique (Lemma A.1).[9]

The second and third mechanisms that we study are the two canonical variants of DA, namely, APDA and IPDA:

**Definition 2.2.** Applicant-Proposing Deferred Acceptance ($\mathsf{APDA} = \mathsf{APDA}_Q(\cdot)$) is defined with respect to a profile of priority orders $Q = \{Q_h\}_h$, one for each institution $h$, over applicants. The matching is produced by repeating the following until every applicant is matched (or has exhausted her preference list): A currently unmatched applicant is chosen to *propose* to her favorite institution that has not yet *rejected* her. The institution then rejects every proposal except for the *top-priority applicant* who has proposed to it thus far. Rejected applicants become (currently) unmatched, while that top-priority applicant is tentatively matched to the institution. This process continues until no more proposals can be made, at which time the tentative allocations become final.

The mechanism Institution-Proposing Deferred Acceptance $\mathsf{IPDA} = \mathsf{IPDA}_Q(\cdot)$ is defined in one-to-one markets identically to APDA, except interchanging the roles of the applicants and the institutions. In other words, the matching of $\mathsf{IPDA}_Q(P)$ coincides with $\mathsf{APDA}_P(Q)$, treating the

[8] We also follow standard notations such as writing $h \succeq_d^{P_d} h'$ when $h \succ_d^{P_d} h'$ or $h = h'$, writing $P \in \mathcal{T}$ to denote $(P_1, \ldots, P_n) \in \mathcal{T}_1 \times \ldots \times \mathcal{T}_n$, writing $P_{-i} \in \mathcal{T}_{-i} = \mathcal{T}_1 \times \ldots \times \mathcal{T}_{i-1} \times \mathcal{T}_{i+1} \times \ldots \times \mathcal{T}_n$, and writing $(P_i', P_{-i})$ to denote $(P_1, \ldots, P_{i-1}, P_i', P_{i+1}, \ldots, P_n)$. For a set of applicants $S$, we also write $P_{-S}$ for a profile of preferences of applicants not in set $S$. For a matching $\mu$ and institution $h$, we abuse notation and let $\mu(h) \in \mathcal{A} \cup \{\emptyset\}$ denote the applicant $d$ (if there is any) with $\mu(d) = h$. To avoid common variables $a$ and $i$, we typically use $d$ (mnemonic: doctor) to denote an element of $\mathcal{A}$ and $h$ (mnemonic: hospital) to denote an element of $\mathcal{I}$.

[9] TTC is also commonly studied under a model where each applicant starts out "owning" a single institution (a so-called housing market, which is equivalent to each institution having a distinct top-priority applicant in our model). Our lower bounds outside of Section 5 hold for this model as well.

preferences $P$ as priorities and priorities $Q$ as preferences.

We use DA to denote either APDA or IPDA when the distinction is not important (i.e., when the result holds for both mechanisms by the same proof). Both APDA and IPDA produce stable matchings. A matching $\mu$ is stable if there is no unmatched pair $d, h$ such that $d \succ_h \mu(h)$ and $h \succ_d \mu(d)$. Despite the fact that these procedures do not specify the order in which proposals are made, the matching produced by each of them is unique (Corollary A.4).

We also consider the very simple mechanism of *serial dictatorship* ($\mathsf{SD} = \mathsf{SD}_{\succ}(\cdot)$), which corresponds to both TTC and to DA in the case where all institutions' priority lists are $\succ$. If $\succ$ ranks applicants in order $d_1 \succ \ldots \succ d_n$, then the matching is determined by the preference lists $P_{d_1}, \ldots, P_{d_n}$ (in that order) picking their favorite remaining institution; we denote this as $\mathsf{SD}_{d_1,\ldots,d_n}(\cdot)$.

**Strategyproofness, menus, and nonbossiness.** The matching rules of TTC and APDA (though not IPDA) are *strategyproof* for the applicants, i.e., for all $d \in \mathcal{A}$, all $P_d, P'_d \in \mathcal{T}_d$ and all $P_{-d} \in \mathcal{T}_{-d}$, we have $f_d(P_d, P_{-d}) \succeq_d^{P_d} f_d(P'_d, P_{-d})$. This property is tightly connected to the classical notion of the *menu* of a player in a mechanisms, which is a central notion throughout our paper. The menu is the natural definition of the set of obtainable institutions that an applicant has in the mechanism:

**Definition 2.3.** For any matching rule $f$ and applicant $d$, the *menu* $\mathsf{Menu}_d^f(P_{-d})$ *of* $d$ *given* $P_{-d} \in \mathcal{T}_{-d}$ is the subset of all institutions $h \in \mathcal{I}$ such that there exists some $P_d \in \mathcal{T}_d$ such that $f_d(P_d, P_{-d}) = h$. That is,

$$\mathsf{Menu}_d(P_{-d}) = \mathsf{Menu}_d^f(P_{-d}) = \left\{ f_d(P_d, P_{-d}) \;\middle|\; P_d \in \mathcal{T}_d \right\} \subseteq \mathcal{I}.$$

The menu is a lens through which one can view or understand strategyproofness, as captured by the following equivalence:

**Theorem 2.4** ([Ham79])**.** *A matching rule $f$ is strategyproof if and only if each applicant $d$ is always matched to her favorite institution from her menu (that is, for each $P_{-d} \in \mathcal{T}_{-d}$ and $P_d \in \mathcal{T}_d$, we have $f_d(P_d, P_{-d}) \succeq_d^{P_d} h$ for any $h \in \mathsf{Menu}_d^f(P_{-d})$).*

We also frequently make use of the classically studied property of nonbossiness:

**Definition 2.5.** A mechanism $f$ is *nonbossy* if, for all $d \in \mathcal{A}$, all $P_d, P'_d \in \mathcal{T}_d$, and all $P_{-d} \in \mathcal{T}_{-d}$, we have the following implication:

$$f(P_d, P_{-d}) \neq f(P'_d, P_{-d}) \quad \Longrightarrow \quad f_d(P_d, P_{-d}) \neq f_d(P'_d, P_{-d}).$$

That is, if changing $d$'s report changes some applicant's match, then it in particular changes $d$'s own match.

TTC and IPDA are nonbossy, but APDA is bossy.

# 3 Type-to-Matching Effect Complexity

## 3.1 Discussion and Definitions

As per Question 1 in Section 1, the results in Section 3 and Section 4 are designed to answer the question: how complex can the effect of one applicant's report be? We measure this complexity via the number of bits it takes to represent the function from one applicant's preference $P_d$ to some

other piece of data.

For a first discussion, we consider the complexity of one applicant's effect on a single applicant's match. We separately consider the effect of one's report on another's match, and on one's own match, as follows:

**Definition 3.1.** The *type-to-another's-match complexity* of a matching mechanism $f$ is

$$\log_2 \max_{d_*, d_\dagger \in \mathcal{A}} \left| \left\{ f_{d_\dagger}(\cdot, P_{-d_*}) \;\middle|\; P_{-d_*} \in \mathcal{T}_{-d_*} \right\} \right|,$$

where $f_{d_\dagger}(\cdot, P_{-d_*}) : \mathcal{T}_{d_*} \to \mathcal{M}$ is the function mapping each $P_{d_*} \in \mathcal{T}_{d_*}$ to the match of applicant $d_\dagger \neq d_*$ in $f(P_{d_*}, P_{-d_*})$.

The *type-to-one's-own-match* complexity is as in the above definition, except taking $d_\dagger = d_*$.

For any strategyproof mechanism, an applicant is always matched to her top-ranked institution on her menu, by Theorem 2.4. Thus, writing down $d_*$'s menu suffices to describe $d_*$'s match under any possible report, showing that the type-to-one's-own-match complexity is $\widetilde{O}(n)$. If the mechanism is additionally nonbossy, there are at most $n$ matchings $\mu$ that can result from $d_*$ submitting any preference list $P_{d_*}$; by writing down both the menu and the value of $\mu(d_\dagger)$ for each resulting matching $\mu$, one additionally knows the function $f_{d_\dagger}(\cdot, P_{d_*})$ for $d_\dagger \neq d_*$. This shows that the type-to-another's-match complexity of a strategyproof and nonbossy mechanism is $\widetilde{O}(n)$. All told:

**Observation 3.2.** *The type-to-one's-own-match complexity of* TTC *and* APDA *is* $\widetilde{O}(n)$. *The type-to-another's-match complexity of* TTC *is* $\widetilde{O}(n)$.

In contrast, note that the type-to-another's-match complexity of APDA is not immediately clear. Note also that despite IPDA being nonbossy, and thus there are at most $n$ distinct matchings that can result from $d_*$ unilaterally changing her report $P_{d_*}$, it is not immediately clear how $d_*$'s report $P_{d_*}$ determines which of the $n$ matchings are the result of IPDA. Thus, it is not clear how to bound either the type-to-one's-own-match complexity or the type-to-another's-match complexity for IPDA. In Section 4.2 we obtain $\widetilde{O}(n)$ upper bounds for DA as well, as a corollary of how one applicant can effect another's menu.

We now proceed to consider a different generalization of Definition 3.1: one applicant's effect on the matching overall, the main complexity measure of this section.

**Definition 3.3.** The *type-to-matching complexity* of a matching mechanism $f$ is

$$\log_2 \max_{d \in \mathcal{A}} \left| \left\{ f(\cdot, P_{-d}) \;\middle|\; P_{-d} \in \mathcal{T}_{-d} \right\} \right|,$$

where $f(\cdot, P_{-d}) : \mathcal{T}_d \to \mathcal{M}$ is the function mapping each $P_d \in \mathcal{T}_d$ to the matching $f(P_d, P_{-d})$.

**Observation 3.4.** *For any matching mechanism $f$, the type-to-one's-own-match complexity and the type-to-another's-match complexity are at most the type-to-matching complexity.*

To begin the discussion of type-to-matching complexity, consider the very simple mechanism SD. In this mechanism, applicants are chosen in order $1, 2, \ldots, n$, and each chosen applicant is permanently matched to her top-ranked remaining institution. Despite the extreme simplicity of this mechanism, and despite the fact that it is strategyproof and nonbossy, it is not immediately clear whether there is any way to represent the function $\mathsf{SD}(\cdot, P_{-1}) : \mathcal{T}_1 \to \mathcal{M}$ more efficiently than to write down a separate matching $\big(\mathsf{SD}(\{h_1\}, P_{-1}), \ldots, \mathsf{SD}(\{h_n\}, P_{-1})\big)$ for each possible institution

that applicant 1 might pick. This representation takes $\widetilde{\Omega}(n^2)$ bits (matching the $\widetilde{\Omega}(n^2)$ solution that simply records the entirety of $P_{-1}$). In Appendix D we show that this complexity is $\widetilde{O}(n)$ for SD, using a novel data structure representing all possible matchings as a function of $P_1$. For our main mechanisms of interest, this complexity measure will in contrast turn out to be high, as we see next.

### 3.2 TTC

We now bound the type-to-matching complexity of TTC. Informally, this complexity measure is high because an applicant in TTC can dramatically affect the order in which future cycles will be matched in TTC, and thus dramatically affect the entirety of the matching.

To present our formal proof, we first define an intermediate mechanism that we call $\mathsf{SD}^{\mathsf{rot}}$, a variant of SD where the first applicant can affect the order in which other applicants choose institutions. We show that, unlike SD, the mechanism $\mathsf{SD}^{\mathsf{rot}}$ has high type-to-matching complexity. Then, we show that TTC can "simulate" $\mathsf{SD}^{\mathsf{rot}}$, and thus TTC has type-to-matching complexity at least as high as $\mathsf{SD}^{\mathsf{rot}}$.

**Definition 3.5.** Consider a matching market with $n+1$ applicants $\{d_*, d_1, \ldots, d_n\}$ and $2n$ institutions $\{h_1, \ldots, h_n, h_1^{\mathsf{rot}}, \ldots, h_n^{\mathsf{rot}}\}$. Define a mechanism $\mathsf{SD}^{\mathsf{rot}}$ as follows: first, $d_*$ is permanently matched to her top-ranked institution $h_j^{\mathsf{rot}}$ from $\{h_1^{\mathsf{rot}}, \ldots, h_n^{\mathsf{rot}}\}$. Then, in order, each of the applicants $d_j, d_{j+1}, \ldots, d_{n-1}, d_n$ is permanently matched to her top-ranked remaining institution from $\{h_1, \ldots, h_n\}$ (and all other applicants go unmatched). In other words, applicants are allocated to $\{h_1, \ldots, h_n\}$ according to $\mathsf{SD}_{d_j, d_{j+1}, \ldots, d_n}(\cdot)$.

Informally, this mechanism has high type-to-menu complexity because (under different reports of applicant $d_*$) each applicant $d_i$ with $i > 0$ may be matched *anywhere* in the ordering of the remaining agents, and thus *any* part of $d_i$'s preference list might matter for determining $d_i$'s matching.

**Lemma 3.6.** *The type-to-matching complexity of* $\mathsf{SD}^{\mathsf{rot}}$ *is* $\Omega(n^2)$.

*Proof.* Fix $k$, where we will take $n = \Theta(k)$. For notational convenience, we relabel the applicants $d_1, d_2, \ldots, d_n$ as $d_1^L, d_1^R, d_2^L, d_2^R, \ldots, d_k^L, d_k^R$ in order, and relabel the institutions $h_1, h_2, \ldots, h_n$ as $h_1^0, h_1^1, h_2^0, h_2^1, \ldots, h_k^0, h_k^1$. We define a collection of preference profiles of all applicants other than $d_*$. This collection is defined with respect to a set of $k(k+1)/2 = \Omega(n^2)$ bits $b_{i,j} \in \{0,1\}$, one bit for each $i, j \in [k]$ with $j \leq i$. For such a bit vector, consider preferences such that for each $i \in [k]$, we have:

$$d_i^L: \quad h_1^{b_{i,1}} \succ h_1^{1-b_{i,1}} \succ h_2^{b_{i,2}} \succ h_2^{1-b_{i,2}} \succ \ldots \succ h_i^{b_{i,i}} \succ h_i^{1-b_{i,i}}$$
$$d_i^R: \quad h_1^{1-b_{i,1}} \succ h_1^{b_{i,1}} \succ h_2^{1-b_{i,2}} \succ h_2^{b_{i,2}} \succ \ldots \succ h_i^{1-b_{i,i}} \succ h_i^{b_{i,i}}$$

In words, each such list agrees that $h_1^0$ and $h_1^1$ are most preferred, followed by $h_2^0$ and $h_2^1$, etc., and the lists of $d_i^L$ and $d_R^i$ rank all such institutions up to $h_i^0$ and $h_i^1$. But $d_i^L$ and $d_R^i$ may flip their ordering over each $h_j^0$ and $h_j^1$ for $j \leq i$, as determined by the bit $b_{i,j}$. The key lemma is the following:

**Lemma 3.7.** *Consider any* $i, j \in [k]$ *with* $j \leq i$. *If* $d_*$ *submits a preference list containing only* $\{h_{i-j+1}^{\mathsf{rot}}\}$, *then applicant* $d_{i,j}^L$ *matches to* $h_j^{b_{i,j}}$.

To prove this lemma, consider the execution of $\mathsf{SD}(d_{i-j+1}^L, d_{i-j+1}^R, \ldots, d_k^L, d_k^R)$. Initially, applicants $d_{i-j+1}^L, d_{i-j+1}^L$ pick $h_1^0$ and $h_1^1$, then applicants $d_{i-j+2}^L, d_{i-j+2}^L$ pick $h_2^0, h_2^1$, and so on, until

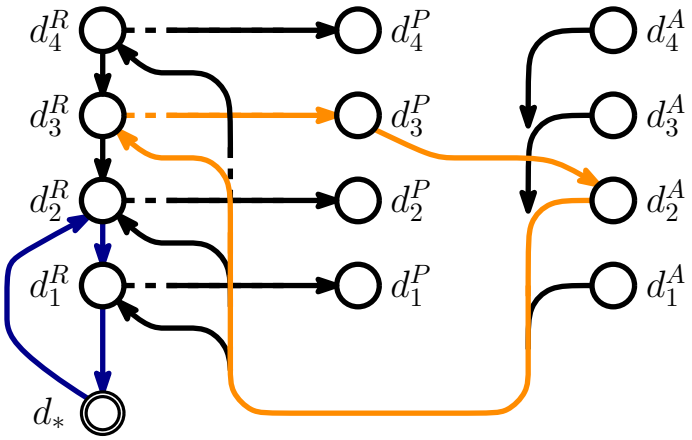


Figure 1: Illustration of the set of preferences showing that the type-to-matching complexity of $TTC$ is $\Omega(n^2)$ (Theorem 3.8).

**Notes:** Two example cycles are highlighted, the first with $d_*$ matching to $h_2^R$, and the second with $d_3^P$ matching to $d_2^A$, her top choice from $\{d_1^A, \dots, d_4^A\}$.

applicants $d_i^L, d_i^R$ pick among $h_j^0, h_j^1$. Thus, $d_i^L$ pick $h_j^{b_{i,j}}$, proving Lemma 3.7.

This shows that for every pair of distinct preference profiles $P$ and $P'$ of the above form, there exists a preference list $P_{d_*}$ of $d_*$ such that $\mathsf{SD}^{\mathsf{rot}}(P_{d_*}, P) \neq \mathsf{SD}^{\mathsf{rot}}(P_{d_*}, P')$. Thus, there are at least $2^{\Omega(k^2)}$ distinct possible functions $\mathsf{SD}^{\mathsf{rot}}(\cdot, P_{-d_*})$, and the type-to-matching complexity of $\mathsf{SD}^{\mathsf{rot}}$ is at least $\Omega(k^2) = \Omega(n^2)$, as claimed. □

We now proceed to our first main mechanism of interest, $\mathsf{TTC}$.[10]

**Theorem 3.8.** *The type-to-matching complexity of* $\mathsf{TTC}$ *is* $\Omega(n^2)$.

*Proof.* For some $k = \Theta(n)$, we consider applicants $\mathcal{A} = \{d_*, d_1^R, \dots, d_k^R, d_1^P, \dots, d_k^P, d_1^A, \dots, d_k^A\}$. For this construction, we consider markets in which there are exactly $|\mathcal{A}|$ institutions, and each institution ranks a distinct, unique applicant with highest priority. In this case, it is easy to see that only the top-priority ranking of each institution can matter for determining the outcome, so this defines the priorities of the institutions. For notational convenience, we identify each institution with the applicant that they rank highest, e.g., we use $d_1^R$ to denote both an applicant and the institution which ranks $d_1^R$ highest. (This is equivalent to considering this construction in a housing market, i.e., where each applicants starts by "owning" the institution where they have highest priority.)

We describe collection of preference profiles which each induce a distinct function $\mathsf{TTC}(\cdot, P_{-d_*}) : \mathcal{T}_{d_*} \to \mathcal{M}$. For applicants outside of $\{d_*, d_1^P, \dots, d_k^P\}$, the preferences are fixed, and defined as follows:

$$\begin{aligned} d_1^R: &\quad d_* \succ d_1^P \\ d_i^R: &\quad d_{i-1}^R \succ d_i^P && \forall i \in \{2, \dots, k\} \\ d_i^A: &\quad d_1^R \succ d_2^R \succ \dots \succ d_k^R && \forall i \in \{1, \dots, k\} \end{aligned}$$

This construction allows us to embed run of $\mathsf{SD}^{\mathsf{rot}}$ into $\mathsf{TTC}$ as follows:

**Lemma 3.9.** *Suppose applicants in* $\{d_1^P, \dots, d_k^P\}$ *only rank institutions in* $\{d_1^A, \dots, d_k^A\}$. *Furthermore, suppose that applicant* $d_*$ *submits list* $\{d_j^R\}$ *for some* $j \in \{0, 1, \dots, k-1\}$, *where we*

[10] Our construction uses a "$\mathsf{TTC}$-pointing graph" with one long path and several isolated vertices in order to simulated $\mathsf{SD}^{\mathsf{rot}}$. One can show that if this initial $\mathsf{TTC}$-pointing graph consists of *only* a long path, then the type-to-matching complexity is low. Moreover, if this initial graph consists only of isolated vertices, then matching the cycle that $d_*$ completes can only affect the pointing graph in a quite small way. Thus, the way our construction works seems like a good fit for bounding the type-to-matching complexity of $\mathsf{TTC}$.

*denote institution $d_*$ as $d_0^R$. Then, applicants in $\{d_1^P, \ldots, d_k^P\}$ will be matched to institutions in $\{d_1^A, \ldots, d_k^A\}$ according to $\mathsf{SD}_{d_{j+1}^P, d_{j+2}^P, \ldots, d_k^P}(\cdot)$.*

To prove this lemma, consider the run of *TTC* after $d_*$ points to institution $d_j^R$. First, $d_*$ and each $d_1^R, \ldots, d_j^R$ is matched to their top-ranked institution in one cycle. Now, each applicant in $\{d_1^A, \ldots, d_k^A\}$ has the same preference list, and her top-ranked remaining institution is $d_{j+1}^R$. Additionally, $d_{j+1}^R$ now points to $d_{j+1}^P$, so $d_{j+1}^P$ will be matched to her top-ranked institution in $\{d_1^A, \ldots, d_k^A\}$. Next, all of the remaining institutions in $\{d_1^A, \ldots, d_k^A\}$ point (transitively through $d_{j+2}^R$) to $d_{j+2}^P$, who can match to her top-ranked remaining institution. This continues for all additional applicants in $\{d_{j+1}^P, \ldots, d_k^P\}$. This proves Lemma 3.9; see Figure 1 for an illustration.

Thus, applicants in $\{d_1^P, \ldots, d_k^P\}$ are matched to $\{d_1^A, \ldots, d_k^A\}$ according to $\mathsf{SD}^{\mathsf{rot}}$, and the type-to-matching complexity of TTC is at least as high as $\mathsf{SD}^{\mathsf{rot}}$, which by Lemma 3.9, is $\Omega(k^2) = \Omega(n^2)$. □

### 3.3 APDA and Relation to [GHT22]

We now study the type-to-matching complexity of the stable matching mechanisms APDA and IPDA. For APDA, this result is actually an immediate corollary of results from [GHT22]. Indeed, we initially formulated Theorem 3.8 for TTC in order to provide a deeper investigation of results proved in [GHT22], which paint a contrast between TTC and APDA. We now explain this result and how it implies a bound on the type-to-matching complexity of APDA.

[GHT22] look for algorithms for computing matching rules while making the strategyproofness of these matching rules clear. For a strategyproof $f$ and some applicant $d$, they look for algorithms with the following three steps:

(1) Using only $P_{-d}$, the preferences of applicants other than $d$, calculate $d$'s menu $\mathsf{Menu}_d^f(P_{-d})$.

(2) Using $d$'s preferences $P_d$, match $d$ to her favorite institution from her menu.

(3) Using $P_d$ and $P_{-d}$, calculate the rest of the matching $f(P_d, P_{-d})$.

Furthermore, [GHT22] look for algorithms with the above outline that are also similar to the tradition descriptions of the algorithms. For TTC and DA, they model this through algorithms that (A) read the preferences of each applicant one time only, in favorite-to-least-favorite order, and (B) only store roughly the amount of memory required to keep track of a single matching, namely, $\widetilde{O}(n)$ bits.

[GHT22] prove that for TTC and each applicant $d$, there is a description meeting the above three-step outline that is *very* similar to the traditional description. In fact, this description follows directly from the fact that TTC is independent of the order in which cycles are matched (Lemma A.1). Namely, one run of TTC goes as follows:

(1) Iteratively match all possible cycles not involving applicant $d$, i.e., run TTC without letting applicant $d$ point to any institutions until no more cycles are possible.

(2) Since there must be some cycle when every applicant points, now match $d$ to her favorite institution according to $P_d$.

(3) Continue calculating the TTC matching in any order.

In particular, this description reads preferences and stores memory in a very similar way to the traditional description of TTC. In stark contrast to TTC, however, [GHT22] prove that for any

description of APDA meeting the above three-step outline, if that description reads each applicant's preference in favorite-to-least-favorite order, then the description *requires* $\Omega(n^2)$ *memory*. In particular, such a description cannot possibly just work by iteratively modifying a single tentative matching using $\widetilde{O}(n)$ memory, the way the traditional descriptions of APDA work. For additional details, and discussion in context, see [GHT22].

One reason we find Theorem 3.8 fascinating in this context is that it shows that from a "global" perspective, TTC is just as complex as APDA. In particular, the type-to-matching complexity of $f$ exactly equals the memory requirements of an algorithm with the following *two* steps:

(1) Perform any calculation whatsoever using only $P_{-i}$.

(2) Calculate the matching $f(P_d, P_{-d})$ using only $P_i$.

Thus, for TTC, despite the fact that [GHT22] provide an algorithm for the three-step outline with $\widetilde{O}(n)$ memory, our result Theorem 3.8 shows that any two-step outline requires vastly more memory, namely, $\Omega(n^2)$. In particular, the "pointing graph" at the end of Step (1) above does not contain *nearly* enough information to calculate the rest of the matching. Thus, the three-step outline very precisely captures the sense in which TTC is "strategically simpler" than APDA under the framework of [GHT22].

For our direct purposes, we observe the following corollary of the construction in [GHT22].

**Corollary 3.10** (Follows from [GHT22, Section 5.3])**.** *The type-to-matching complexity of* APDA *is* $\Omega(n^2)$.

*Proof.* The impossibility theorem of [GHT22] directly constructs a set of $2^{\Omega(n^2)}$ preferences for applicants other than $d_*$ such that the function $\mathsf{APDA}(\cdot, P_{-d_*}) : \mathcal{T}_{d_*} \to \mathcal{M}$ is distinct for each $P_{-d_*}$ in this class. □

In the next section, we lower bound the type-to-matching complexity for IPDA.[11]

### 3.4 IPDA

We next turn our attention to the type-to-matching complexity of IPDA. While APDA and IPDA are both stable mechanisms, they operate in a different way and can produce very different matchings. Since IPDA is nonbossy, one might even hope for low complexity, however, we now prove another $\Omega(n^2)$ lower bound. The analysis is very different from that of APDA, though.[12]

**Theorem 3.11.** *The type-to-matching complexity of* IPDA *is* $\Omega(n^2)$.

*Proof.* Fix $k$, where we will have $n = |\mathcal{I}| = |\mathcal{A}| = \Theta(k)$. The institutions are $h_i^0, h_i^1$ for $i = 1, \ldots, k$ and $h_i^R$ for $i = 0, \ldots, k$, and the applicants are $d_i, d_i'$ for $i = 1, \ldots, k$ and $d_i^R$ for $i = 1, \ldots, k$, as

[11] While IPDA is not strategyproof, and hence the above three-step outline does not apply to IPDA, there is a different sense in which a type-to-matching lower bound for IPDA shows that an algorithm from [GHT22] is tight. Namely, [GHT22, Appendix D.3] constructs a delicate algorithm using the outcome of IPDA as a building block, and if there were an algorithm $A$ that were able to calculate and store the function $\mathsf{IPDA}(\cdot, P_{-d_*})$ in $\widetilde{O}(n)$ bits, then the delicate algorithm of [GHT22] could have been easily implemented using calls to $A$.

[12] In fact, one can show that there is a single stable matching that is the result of IPDA under *any* preference profile among the set of inputs constructed by [GHT22, Section 5.3]. More broadly, bounding the type-to-matching complexity in APDA seems informally easier than in IPDA, because in APDA, an applicant $d_*$ can trigger a "single effect" by making a proposal at will, but in IPDA, each applicant (including $d_*$) rejects all proposals except for at most one, so she must trigger all effects except for one. Our construction overcomes this difficulty by making the proposals to $d_*$ all sequential, and giving each such proposal the ability to change the match of all applicants.

well as $d_*$. First we define the fixed priorities $Q$ of the institutions (where, for the entirety of this construction, we take indices mod $k$):

$$\begin{aligned}
&h_i^b : d_i \succ d'_i \succ d_{i+1} \succ d'_{i+1} \succ \ldots \succ d_{i-1} \succ d'_{i-1} && \text{For each } i = 1, \ldots, k \text{ and } b \in \{0, 1\}\\
&h_0^R : d_* \succ d_1^R\\
&h_1^R : d_* \succ d_1 \succ d'_1 \succ d_2^R\\
&h_i^R : d_i^R \succ d_* \succ d_1 \succ d'_1 \succ d_{i+1}^R && \text{For each } i = 2, \ldots, k-1\\
&h_k^R : d_k^R \succ d_*
\end{aligned}$$

The preferences of the applicants $\{d'_1, \ldots, d'_n, d_0^R, d_1^R, \ldots, d_k^R\}$ are fixed. The preferences of applicants $\{d_1, \ldots, d_k\}$ depend on bits $(b_{i,j})_{i,j \in \{1,\ldots,k\}}$ where each $b_{i,j} \in \{0, 1\}$. Since the run of IPDA will involve applicants receiving proposals in (loosely) their reverse order of preference, we display the preferences of applicants in worst-to-best order for readability. The preferences are as follows: First, for applicants in $\{d_1^R, \ldots, d_k^R\}$:

$$d_1^R : h_0^R \qquad\qquad d_i^R : h_i^R \prec h_{i-1}^R \qquad\qquad \text{For each } i = 2, \ldots, k$$

Next, for $i = 2, 3, \ldots, k$, we have:

$$\begin{array}{lccccccccc}
d_i : & h_i^{1-b_{1,1}} & \prec & h_i^{b_{1,1}} & \prec & h_{i-1}^{1-b_{1,2}} & \prec & h_{i-1}^{b_{1,2}} & \prec & h_{i-2}^{1-b_{1,3}} \prec h_{i-2}^{b_{1,3}} \prec \ldots \prec h_{i+1}^{1-b_{1,k}} \prec h_{i+1}^{b_{1,k}}\\
d'_i : & h_i^0 & \prec & h_i^1 & \prec & h_{i-1}^0 & \prec & h_{i-1}^1 & \prec & h_{i-2}^0 \prec h_{i-2}^1 \prec \ldots \prec h_{i+1}^0 \prec h_{i+1}^1
\end{array}$$

In words, applicants of the form $d'_i$ always prefer institutions in the cyclic order, starting with institutions of the form $h_i^b$ as their least-favorites. Applicants of the form $d_i$ also rank institutions like this, but they flip adjacent places in this preference based on the bits $b_{i,j}$.

Finally, we define the preferences of $d_1$ and $d'_1$, which are like the other $d_i$ and $d'_i$, except that these applicants will also accept proposals from institutions of the form $h_j^R$. Specifically:

$$\begin{array}{lccccccccc}
d_1 : & h_1^{1-b_{1,1}} & \prec & h_1^{b_{1,1}} & \prec & h_1^R \prec h_k^{1-b_{1,2}} & \prec & h_k^{b_{1,2}} & \prec & h_2^R \prec h_{k-1}^{1-b_{1,3}} \prec h_{k-1}^{b_{1,3}} \prec \ldots \prec h_{k-1}^R \prec h_2^{1-b_{1,k}} \prec h_2^{b_{1,k}}\\
d'_1 : & h_1^0 & \prec & h_1^1 & \prec & h_1^R \prec h_k^0 & \prec & h_k^1 & \prec & h_2^R \prec h_{k-1}^0 \prec h_{k-1}^1 \prec \ldots \prec h_{k-1}^R \prec h_2^0 \prec h_2^1
\end{array}$$

These preferences are illustrated in Figure 2, along with an informal description of how the preferences operate. Formally, our key claim is the following:

**Lemma 3.12.** *Consider any $j \in \{1, \ldots, k\}$. If $d_*$ submits a preference list containing only $\{h_j^R\}$, then for each $i \in \{1, \ldots, k\}$, it holds that $d_i$ is matched to $h_{i-j+1}^{b_{i,j}}$ in* IPDA.[13]

To prove this lemma, start by considering $j = 1$. When $j = 1$, each $h_i^0$ and $h_i^1$ for $i = 1, \ldots, k$ simply proposes to $d_i$, and $d_i$ prefers and tentatively accept $h_i^{b_{i,1}}$. Moreover, $d_*$ rejects $h_0^R$, who then proposes to $d_1^R$, and no further proposals are made.

Now consider $j \geq 2$, and suppose we choose an order of proposals such that $d_*$ has already rejected $h_{j-2}^R$, but has not yet rejected $h_{j-1}^R$. This is equivalent to tentatively considering the matching where $d_*$ submits only list $h_{j-1}^R$, so by induction, we know that we have each $d_i$ and $h'_i$ matched to $h_{i-j+2}^0$ and $h_{i-j+2}^1$. In particular, $d_1$ and $d'_1$ are matched to $h_{3-j}^0$ and $h_{3-j}^1$ (equivalently, $h_{k-j+3}^0$ and $h_{k-j+3}^1$). Now consider what happens when $d_*$ rejects $h_{j-1}^R$. First, $h_{j-1}^R$ proposes to $d_1$, and the proposal is tentatively accepted. This causes rejections among $d_1$ and $d'_1$ which lead

[13] This construction would also work with the full-length list $h_k^R \succ h_{j+1}^R \succ h_j^R \succ h_0^R \succ h_{j-1}^R \succ \ldots \succ h_1^R \succ \ldots$.

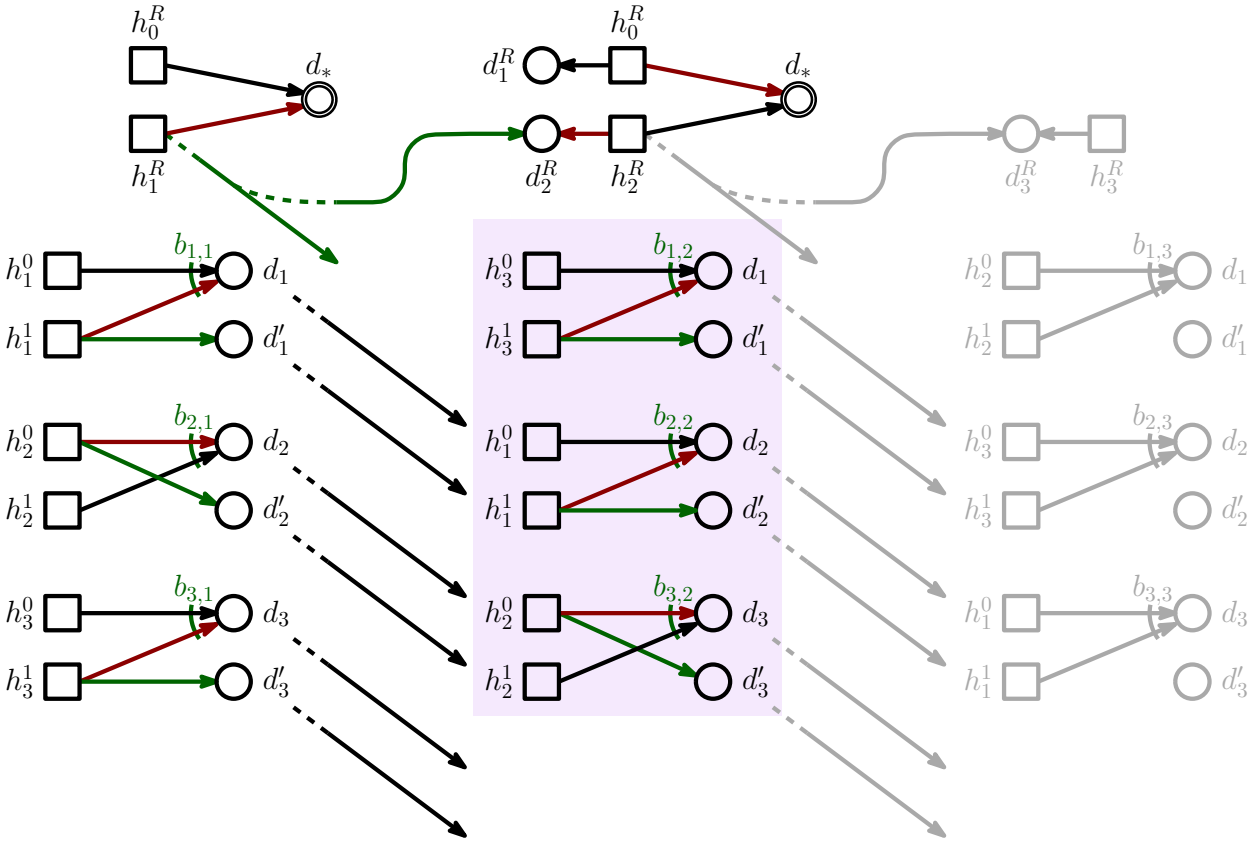


Figure 2: Illustration of the set of preferences showing that the type-to-matching complexity of IPDA is $\Omega(n^2)$ (Theorem 3.11).

**Notes:** When $d_*$ is matched to $h_j^R$, all the institutions in $h_1^R, h_2^R, \ldots, h_{j-1}^R$ propose to $d_1$ and $d'_1$. This successively "rotates" all of the institutions and applicants once for each $h_\ell^R$ that is rejected by $d_*$, so the institutions and applicants are rotated $j-1$ times. When this happens, the bits $b_{i,j}$ determine the match of each applicant $d_i$, which shows that all of the bits $b_{i,j}$ can affect the matching for some possible report of $d_*$. In the illustration, $d_*$ matches to $h_2^R$, so the applicants are rotated once; the selected matching is in the highlighted box, and proposals made in hypothetical future rotations are shown in grey.

$h^0_{3-j}$ to propose to $d_2$. This leads analogously to $h^0_{4-j}$ proposing to $d_3$. This continues similarly, with each $h^0_{i-j+1}$ proposing to $d_i$ for each $i$, until $h^0_{2-j}$ proposes to $d_1$. This proposal is accepted, causing $h^R_{j-1}$ to propose to $d'_1$, and $h^1_{3-j}$ proposes to $d_2$. Regardless of which proposal among $h^0_{3-j}$ and $h^1_{3-j}$ is accepted by $d_2$, we next have $h^1_{4-j}$ proposing to $d_3$. This continues similarly, with each $h^1_{i-j+1}$ proposing to $d_i$ for each $i$, until $h^1_{2-j}$ proposes to $d_1$, leading to $h^R_{j-1}$ being rejected by $d'_1$. Finally, $h^R_{j-1}$ proposes to $d^R_j$, and $d_*$ next receives a proposal from $h^R_j$, which she accepted, ending the run of IPDA.

All told, when $h_*$ submits list $\{h^R_j\}$, each $d_i$ receives a proposal from $h^0_{i-j+1}$ and $h^1_{i-j+1}$, and picks and is finally matched to whichever of the two she prefers according to $b_{i,j}$. This proves Lemma 3.12.

Thus, for each possible profile of bits $b = (b_{i,j})_{i,j\in\{1,\dots k\}}$, there is a distinct function $\mathsf{IPDA}_Q(\cdot, P_{-d_*})$. This proves that the type-to-menu complexity of IPDA is at least $k^2 = \Omega(n^2)$. □

# 4 Type-to-Menu Effect Complexity

In Section 3, we studied the complexity of the effect that one applicant's report has on the entire matching. One somewhat atypical attribute of this framework is that it measures the complexity of a function from $\widetilde{\Theta}(n)$ bits (one applicant's preference list) to $\widetilde{\Theta}(n)$ bits (the matching overall). There is another very natural $O(n)$-bit piece of data in the context of matching mechanisms—another applicant's menu (Definition 2.3)—which we now use to get a new lens into how one applicant can effect another.

In this section, we establish bounds on the complexity of the function from one applicant's report to another applicant's menu, quantifying the complexity of the effect that one applicant can have on another applicant's set of available options. We will show that this quantity is high for TTC, providing another novel way in which TTC is complex. In contrast, we give a new structural result for stable matching mechanisms which shows that this quantity is low for DA. We hope this to be of independent interest, and which in Section 4.3 we also use to give additional characterizations and connections to [GHT22]. Our main definition is:

**Definition 4.1.** The *type-to-menu complexity* of a matching mechanism $f$ is

$$\log_2 \max_{d_*,d_\dagger\in\mathcal{A}} \left|\left\{\mathsf{Menu}^f_{d_\dagger}(\cdot, P_{-\{d_*,d_\dagger\}}) \;\middle|\; P_{-\{d_*,d_\dagger\}} \in \mathcal{T}_{-\{d_*,d_\dagger\}}\right\}\right|,$$

where $\mathsf{Menu}^f_{d_\dagger}(\cdot, P_{-\{d_*,d_\dagger\}}) : \mathcal{T}_{d_*} \to 2^{\mathcal{I}}$ is the function mapping each $P_{d_*} \in \mathcal{T}_{d_*}$ to the menu $\mathsf{Menu}^f_{d_\dagger}(P_{d_*}, P_{-\{d_*,d_\dagger\}}) \subseteq \mathcal{I}$ of $d_\dagger$.

## 4.1 TTC

We now prove an $\Omega(n^2)$ lower bound on the type-to-menu complexity of TTC. One might hope that the ideas behind Theorem 3.8, which reduces the question of the type-to-matching complexity of TTC to the complexity of $\mathsf{SD}^{\mathsf{rot}}$, might gain traction towards bounding the type-to-menu complexity of TTC as well. However, it turns out that $\mathsf{SD}^{\mathsf{rot}}$ has *low* type-to-menu complexity ($\widetilde{O}(n)$, as we prove in Theorem D.4 for completeness) so new ideas are needed. We will now establish a $\Omega(n^2)$ bound for TTC via quite a different construction, which allows applicant $d_*$ to "select" a single applicant $d^X_j$ to point to $d_\dagger$; $d_\dagger$'s menu then contains all institutions that point to $d^X_j$.

**Theorem 4.2.** *The type-to-menu complexity of* TTC *is* $\Omega(n^2)$.

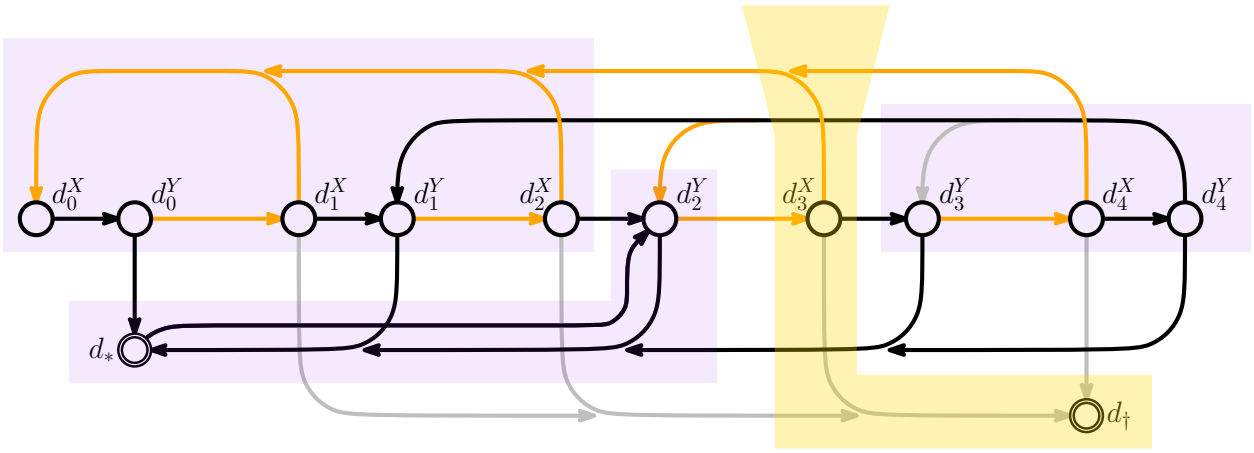


Figure 3: Illustration of the selection gadget used to bound the type-to-menu complexity of TTC (Theorem 4.2).

**Notes:** Applicants' first choices are denoted by black colored arrows, and their second and third choices are colored orange and gray respectively. The illustration shows an example where $d_*$ matches to $h_2^Y$. After this, $\{d_0^X, d_0^Y, d_1^X, d_1^Y, d_2^X\}$ complete a cycle, then $\{d_3^Y, d_4^X, d_4^Y\}$ complete a cycle. Finally, $d_3^X$ points to $d^\dagger$, and $d^\dagger$'s menu is determined by precisely which applicants in $\{d_1^T, d_2^T, d_3^T, d_4^T\}$ point to $d_3^X$.

*Proof.* As in the proof of Theorem 3.8, we prove this bound using a construction with $n = |\mathcal{A}| = |\mathcal{I}|$, and each institution ranking a distinct, unique applicant with highest priority (equivalent to the case of a housing market), and for convenience, we identify each institution with the applicant that it prioritizes highest, e.g., we use $d_1^X$ to denote both an applicant and the institution that prioritizes $d_1^X$ highest.

For some $k = \Theta(n)$, we consider applicants $\mathcal{A} = \{d_*, d_\dagger, d_0^X, d_0^Y, d_1^X, d_1^Y, \ldots, d_k^X, d_k^Y, d_1^T, d_2^T, \ldots, d_k^T\}$. The key to our construction is a "selection gadget" created from applicants of the form $d_j^X$ and $d_j^Y$. Their preferences are fixed, as follows:

$$
\begin{aligned}
&d_0^X : d_0^Y \succ d_0^X && \\
&d_j^X : d_j^Y \succ d_0^X \succ d_\dagger \succ d_j^X && \text{For each } j \in \{1, \ldots, k\} \\
&d_j^Y : d_* \succ d_{j+1}^X && \text{For each } j \in \{0, 1, \ldots, k-1\} \\
&d_k^Y : d_1^Y \succ d_2^Y \succ \ldots \succ d_{k-1}^Y \succ d_k^T &&
\end{aligned}
$$

Our key claim shows that when $d_*$ points to $d_{j-1}^Y$, the selection gadget causes only $d_j^X$ to remain unmatched.

**Lemma 4.3.** *Suppose $d_*$ submits preference list $\{d_{j-1}^Y\}$, for $j \in \{1, \ldots, k\}$. Then every applicant in the selection gadget except for $d_j^X$ is matched to other agents in the selection gadget (but $d_j^X$ is not).*

To prove this lemma, observe that when $d_*$ points to $d_{j-1}^Y$, she matches to $d_{j-1}^Y$. Next, cycle $d_0^X, d_0^Y, d_1^X, \ldots, d_{j-1}^X$ matches. Finally, cycle $d_j^Y, d_{j+1}^Y, \ldots, d_k^Y$ matches. This proves Lemma 4.3; see Figure 3 for an illustration.

Now, for each $i = 1, \ldots, k$, we consider applicant $d_i^T$ whose preferences are defined by an arbitrary subset $T_i \subseteq \{d_1^X, d_2^X, \ldots, d_k^X\}$. Namely,

$$d_i^T : T_i \succ d_i^T,$$

where the element of $T_i$ may be placed on this list in an arbitrary fixed order.

Finally, recall that $d_\dagger$'s menu consists precisely of those institutions that remain unmatched after all possible cycles not involving $d_\dagger$ are eliminated.[14] Thus, Lemma 4.3 shows that for each $j$, when $d_*$ submits list $\{d^Y_{j-1}\}$, exactly those $d^T_i$ such that $j \in B_i$ transitively point to $d_\dagger$ through applicant $d^X_j$. Thus, there is a distinct function $\mathsf{Menu}^{\mathsf{TTC}}_{d_\dagger}(\cdot, P_{-\{d_*,d_\dagger\}})$ for each distinct profile of subsets $T_1, \ldots, T_n$. There are $2^{k^2}$ such sets, showing that the type-to-menu complexity of TTC is $\Omega(k^2) = \Omega(n^2)$, as claimed. □

## 4.2 DA

Thus far, our study of quantifying the effect complexities of matching mechanisms (Question 1) has only yielded negative results and as-high-as-possible lower bounds. In contrast, for the type-to-menu complexity of APDA and IPDA, we will next achieve a positive result, showing that one applicant can only affect another applicant's menu in a combinatorially simple way. First, we remark that by Lemma A.7, the menu in APDA is identical to the menu in IPDA, so proving this bound for the two mechanisms is identical. For the remainder of this section, we thus simply refer to the menu in DA.

**Theorem 4.4.** *The type-to-menu complexity of* DA *is* $\widetilde{\Theta}(n)$.

The remainder of this subsection is dedicated to proving this theorem. Consider any pair of applicant $S = \{d_*, d_\dagger\}$, priorities $Q$ and preferences $P_{-\{d_*,d_\dagger\}}$. Our goal is to represent the function $\mathsf{Menu}^{\mathsf{APDA}_Q}_{d_\dagger}(\cdot, P_{-S}) : \mathcal{T}_{d_*} \to 2^{\mathcal{I}}$ using an $\widetilde{O}(n)$-bit data structure. The starting point of our representation will be the following fact from [GHT22]:

**Lemma 4.5** (Follows from [GHT22, Section 3])**.** *The menu of $d_\dagger$ in* DA *under priorities $Q$ and preferences $P_{-d_\dagger}$ is exactly the set of proposals $d_\dagger$ receives in* $\mathsf{IPDA}_Q(d_\dagger : \emptyset, P_{-d_\dagger})$*, i.e., the set of proposals $d_\dagger$ receives if* IPDA *is run with $d_\dagger$ rejecting all proposals.*

Note that, while this lemma characterizes the menu in both APDA and IPDA, the mechanism IPDA specifically must be used to achieve this characterization (see [GHT22] for a discussion). In every run of IPDA for the remainder of this subsection, the priorities $Q$ and preferences $P_{-S}$ will be fixed. Thus, we suppress this part of the notation, and write $\mathsf{IPDA}(d_S : P_S)$ in place of $\mathsf{IPDA}_Q(d_S : P_S, d_{-S} : P_{-S})$.

Remarkably, beyond Lemma 4.5, the *only* property of DA that we will use in this proof (beyond the definition of how DA is calculated) is the fact that IPDA is independent of the order in which proposals are chosen (Corollary A.4). We start by defining a graph representing collective data about multiple different runs of $\mathsf{IPDA}_Q$ under related preference lists in a cohesive way. This data structure is parametrized by a general set $S$ in order to avoid placing unnecessary assumptions and in hope that it will be of independent interest, but we will always instantiate it with $S = \{d_*, d_\dagger\}$.

**Definition 4.6.** Fix a set $S \subseteq \mathcal{A}$ of applicants, and a profile of priorities $Q$ and preferences $P_{-S}$ of all applicants other than those in $S$. For a $d \in S$ and $h \in \mathcal{I}$, define an ordered list of pairs in $S \times \mathcal{I}$ called $\mathsf{chain}(d, h)$ as follows: First, calculate $\mu = \mathsf{IPDA}(d : \{h\}, d_{S\setminus\{d\}} : \emptyset)$. If $h$ never proposes to $d$, set $\mathsf{chain}(d, h) = \emptyset$. Otherwise, starting from tentative matching $\mu$, let $d$ reject $h$ and have $h$ continue proposing, following the rest of the execution of IPDA with $d$ rejecting all proposals. Note

[14] This follows by the fact that TTC is independent of the order in which cycles are eliminated (Lemma A.1), and the fact that after all cycles not including $d_\dagger$ have been matched, $d_\dagger$ must complete a cycle regardless of which institution they point to. These arguments are from [GHT22, Section 5.2], and are also detailed in our Section 3.3.

that this constitutes a valid run of $\mathsf{IPDA}(d_S : \emptyset)$, and that during this "continuation" there is a unique proposal order because only a single element of $\mathcal{I}$ is proposing at any point in time. Now, define $\mathsf{chain}(d, h)$ as the ordered list of pairs

$$(d = d_0, h = h_0) \longrightarrow (d_1, h_1) \longrightarrow (d_2, h_2) \longrightarrow \ldots \longrightarrow (d_k, h_k),$$

where $d_i \in S$ is each applicant in $S$ receiving a proposal from $h_i \in \mathcal{I}$ during the continued run of $\mathsf{IPDA}$ after $d = d_0$ rejects $h = h_0$, written in the order in which the proposals occur.

Now, define the *un-rejection graph* $\mathsf{UnrejGr} = \mathsf{UnrejGr}(Q, P_{-S})$ as the union of all possible consecutive pairs in $\mathsf{chain}(d, h)$ for all $d \in S$ and $h \in \mathcal{I}$. In other words, $\mathsf{UnrejGr}$ is a directed graph defined on the subset of pairs $S \times \mathcal{I}$ which occur in some $\mathsf{chain}(d, h)$, where the edges are all pairs $(d_i, h_i) \longrightarrow (d_{i+1}, h_{i+1})$ which are consecutive elements of some $\mathsf{chain}(d, h)$.

Note that $\mathsf{UnrejGr} \setminus \mathsf{chain}(d, h)$ is exactly the set of rejections which applicants in $S$ make in $\mathsf{IPDA}(d : \{h\}, d_{S \setminus d} : \emptyset)$, i.e., the set of pairs $(d', h') \in S \times \mathcal{I}$ for which $d$ rejects $h$ in this run of $\mathsf{IPDA}$. We start by establishing general properties of $\mathsf{UnrejGr}$.

**Lemma 4.7.** *Consider any* $\mathsf{chain}(d, h) \neq \emptyset$*, and node* $(d', h')$ *contained in* $\mathsf{chain}(d, h)$*. It must be the case that* $\mathsf{chain}(d', h')$ *equals the tail of the* $V$*, including and after* $(d', h')$*.*

*Proof.* Consider any such $(d, h)$ and $(d', h')$. To start, note that by the definition of $\mathsf{chain}(d, h)$, we cannot have $h'$ propose to $d'$ during $\mathsf{IPDA}(d : \{h\}, d_{S \setminus \{d\}} : \emptyset)$. Thus, this run of $\mathsf{IPDA}$ will produce exactly the same matching as $\mathsf{IPDA}(d : \{h\}, d' : \{h'\}, d_{S \setminus \{d, d'\}} : \emptyset)$, so letting $d$ reject $h$ on top of this must produce a valid run of $\mathsf{IPDA}(d' : \{h'\}, d_{S \setminus \{d'\}})$ by the fact that $\mathsf{DA}$ is independent of execution order (Corollary A.4). After this run, we can now let $d'$ reject $h'$ in order to calculate the $\mathsf{chain}(d', h')$. But this will also correspond exactly to the part of the $\mathsf{chain}(d, h)$ after $h'$ proposes to $d'$, i.e., the tail of the $\mathsf{chain}(d, h)$ including and after $(d', h')$. This finishes the proof. □

**Lemma 4.8.** $\mathsf{UnrejGr}$ *is a DAG with out-degree at most* 1 *at each node.*

*Proof.* Consider any vertex $(d, h)$ in $\mathsf{UnrejGr}$, and consider $\mathsf{chain}(d, h)$. There is at most one edge outgoing from $(d, h)$ in the $\mathsf{chain}(d, h)$ by definition. By Lemma 4.7, if $(d, h)$ appears in any other possible $\mathsf{chain}(d', h')$, then the edge outgoing from $(d, h)$ in this chain must be the same as in $\mathsf{chain}(d, h)$. Thus, the vertices in $\mathsf{UnrejGr}$ have out-degree at most one.

Since $\mathsf{UnrejGr}$ has out-degree at most 1, every possible max-length path in $\mathsf{UnrejGr}$ must equal $\mathsf{chain}(d, h)$ for some value of $(d, h)$ (namely, the first $(d, h)$ along the path). Now, this implies that $\mathsf{UnrejGr}$ must be acyclic, because each possible $\mathsf{chain}(d, h)$ is acyclic by definition. □

Thus, $\mathsf{UnrejGr}$ is a forest. The following notation will be highly convenient:

**Definition 4.9.** For two nodes $v, w$ in $\mathsf{UnrejGr}$, we write $v \trianglelefteq w$ if there exists a path in $\mathsf{UnrejGr}$ from $v$ to $w$. For a subset $T$ of vertices in $\mathsf{UnrejGr}$, we write $\mathsf{nodes}_d(T) \subseteq \mathcal{I}$ to denote the set of all $h \in \mathcal{I}$ such that $(d, h) \in T$; we also refer to nodes of the form $(d, h)$ as $d$-nodes.

Note that $\trianglelefteq$ this defines a partial order on the nodes of $\mathsf{UnrejGr}$, and that $\mathsf{chain}(d, h)$ equals the set of all $(d', h') \in \mathsf{UnrejGr}$ such that $(d, h) \trianglelefteq (d', h')$.

Since $\mathsf{UnrejGr}$ is defined in terms of runs of $\mathsf{IPDA}$ when some $d \in S$ submits a list of the form $\{h\}$, it is not clear how this relates to what will happen when $d$ submits an arbitrary list $P_d$. The following definition will end up providing the connection we need between $\mathsf{UnrejGr}$ and $\mathsf{IPDA}(d : P_d, P_{S \setminus \{d\}} : \emptyset)$ via a combinatorial characterization of the match of $d$ under list $P_d$.

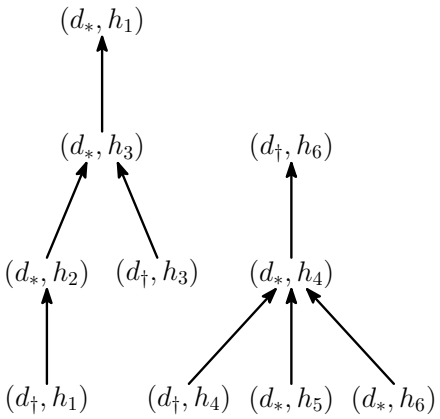


Figure 4: Illustration of UnrejGr with the following priorities and preferences:

$h_1 : d_\dagger \succ d_2 \succ d_*$
$h_2 : d_2 \succ d_* \succ d_3$
$h_3 : d_\dagger \succ d_3 \succ d_* \succ d_2$
$h_4 : d_4 \succ d_\dagger \succ d_* \succ d_5$
$h_5 : d_* \succ d_4 \succ d_5$
$h_6 : d_* \succ d_4 \succ d_5 \succ d_\dagger$

$d_2 : h_3 \succ h_1 \succ h_2$
$d_3 : h_2 \succ h_3$
$d_4 : h_5 \succ h_6 \succ h_4$
$d_5 : h_4 \succ h_6$

**Definition 4.10.** Fix $S$, priorities $Q$, and preferences $P_{-S}$. For any $d \in S$ and $P_d \in \mathcal{T}_d$, define $\mathsf{stab}_d(P_d)$ as the set of $d$-nodes $v = (d, h) \in \mathsf{UnrejGr}$ such that $h \succ_d^{P_d} h'$ for all $h' \in \mathsf{nodes}_d(\mathsf{UnrejGr} \setminus \mathsf{chain}(v))$. In words, $\mathsf{stab}_d(P_d)$ is the set of all $d$-vertices $v$ in UnrejGr such that $P_d$ prefers $v$ to all vertices $v'$ which do not come after $v$ according to $\trianglelefteq$ (in particular, $P_d$ must rank each $h$ in $\mathsf{nodes}_d(\mathsf{stab}_d(P_d))$ as acceptable).[15]

Define $\mathsf{unrej}_d(P_d)$ as the $\trianglelefteq$-minimal element of $\mathsf{stab}_d(P_d)$ (or, if $\mathsf{stab}_d(P_d) = \emptyset$, then set $\mathsf{unrej}_d(P_d) = \emptyset$).

Note that $\mathsf{stab}_d(P_d)$ is defined solely in terms of UnrejGr and $P_d$, and does not depend in any other way on the input priorities $Q$ or preferences $P_{-S}$. Note also that $\mathsf{stab}_d(P_d)$ must be contained in some path in UnrejGr, as otherwise we would have some $(d, h), (d, h') \in \mathsf{stab}_d(P_d)$ which are incomparable under $\trianglelefteq$ with both $h \succ_d^{P_d} h'$ and $h' \succ_d^{P_d} h$. This means that $\mathsf{unrej}_d(P_d)$ is uniquely defined.

**Lemma 4.11.** *Let $\mathsf{unrej}_d(P_d) = (d, h)$ for some $h \in \mathcal{I}$. For any $d \in S$ and $P_d \in \mathcal{T}_d$, the set of proposals which an applicant $d' \in S \setminus \{d\}$ receives in $\mathsf{IPDA}(d : P_d, d_{S\setminus\{d\}} : \emptyset)$ is exactly $\mathsf{nodes}_d\big(\mathsf{UnrejGr} \setminus \mathsf{chain}(\mathsf{unrej}_d(P_d))\big)$.*

*Proof.* Let $m_d(P_d)$ denote $d$'s match in $\mathsf{IPDA}(d : P_d, d_{S\setminus\{d\}} : \emptyset)$ (and note that this definition depends on $Q$ and $P_{-S}$, not just on UnrejGr). First, observe that this run of IPDA also constitutes one valid run of $\mathsf{IPDA}(d : \{m_d(P_d)\}, d_{S\setminus\{d\}} : \emptyset)$, by the fact that IPDA is independent of the order in which proposals are chosen (Corollary A.4), and by the fact that every proposal to $d$ except for $m_d(P_d)$ is eventually rejected in $\mathsf{IPDA}(d : P_d, d_{S\setminus\{d\}} : \emptyset)$. (In other words, whatever ordering of proposals you like under $(d : P_d, d_{S\setminus\{d\}} : \emptyset)$ also corresponds to some ordering of proposals under $(d : \emptyset, d_{S\setminus\{d\}} : \emptyset)$, possibly delaying future proposals from certain $h$ when they propose to $d$. The same argument is used to prove IPDA is nonbossy, Proposition A.9.) Thus, it suffices to show that $m_d(P_d) = \mathsf{unrej}_d(P_d)$, and going forward we can consider $\mathsf{IPDA}(d : \{m_d(P_d)\}, d_{S\setminus\{d\}} : \emptyset)$.

By definition of UnrejGr, the set of institutions which $d$ rejects in $\mathsf{IPDA}(d : \{m_d(P_d)\}, d_{S\setminus\{d\}} : \emptyset)$ is exactly $\mathsf{nodes}_d(\mathsf{UnrejGr} \setminus \mathsf{chain}(d, m_d(P_d)))$. By the definition of IPDA, we know $P_d$ must prefer $m_d(P_d)$ to every institution which they reject. Thus, by the definition of $\mathsf{stab}_d(P_d)$, we must have $m_d(P_d) \in \mathsf{nodes}_d(\mathsf{stab}_d(P_d))$.

To finish the proof, it suffices to show that no $v = (d, h') \lhd (d, m_d(P_d))$ satisfies $v \in \mathsf{stab}_d(P_d)$. Suppose for contradiction that this were the case. Then, by the definition of $\mathsf{chain}(v)$, we know

[15] One can show that $\mathsf{stab}_d(P_d)$ is exactly the set of stable partners of $d$ under priorities $Q$ and preferences $(d : P_d, d_{S\setminus\{d\}} : \emptyset, d_{-S} : P_{-S})$ (for instance, using the techniques of [CT19]); this fact is not needed for our arguments, but it is what inspired the name $\mathsf{stab}_d(P_d)$.

that $m_d(P_d)$ will only ever propose to $d$ after $d$ rejects $h'$ in some run of $\mathsf{IPDA}(d : P', d_{S\setminus\{d\}} : \emptyset)$. But, since $P_d$ prefers $h'$ to every institution in $\mathsf{nodes}_d(\mathsf{UnrejGr} \setminus \mathsf{chain}(d, h'))$, and institutions in $\mathsf{chain}(d, h')$ can only propose to $d$ after $d$ rejects $h'$, we know that $d$ will never reject $h'$ in $\mathsf{IPDA}$. Thus, $m_d(P_d)$ cannot possibly propose to $d$ in $\mathsf{IPDA}(d : P_d, d_{S\setminus\{d\}})$, a contradiction. Thus, we know that we must have $(d, m_d(P_d)) = \mathsf{unrej}_d(P_d)$, as desired. □

We are now ready to prove the theorem.

*Proof (of Theorem 4.4).* Take $S = \{d_*, d_\dagger\}$ in the definition of $\mathsf{UnrejGr}$. Since $\mathsf{UnrejGr}$ has out-degree at most one and at most one vertex for every pair in $S \times \mathcal{I}$, it takes $\widetilde{O}(n)$ bits to represent. Moreover, by Lemma 4.11 and Lemma 4.5, we know that for all $P_{d_*} \in \mathcal{T}_{d_*}$, we have

$$\mathsf{Menu}_{d_\dagger}^{\mathsf{APDA}_Q}(P_{d_*}, P_{-S}) = \mathsf{nodes}_{d_\dagger}(\mathsf{UnrejGr} \setminus \mathsf{chain}(\mathsf{unrej}_{d_*}(P_{d_*}))).$$

Thus, the map from $P_{d_*}$ to the menu of $d_\dagger$ can be represented using only $\mathsf{UnrejGr}$, which takes at most $\widetilde{O}(n)$ bits, as desired.

Note also that $\Omega(n)$ bits are certainly required, since any possible subset of $\mathcal{I}$ could be in $d_\dagger$'s menu (even without taking into account $d_*$'s list). □

### 4.3 Applications and Relation to [GHT22]

We now explore applications of the combinatorial structure we uncovered in Section 4.2, namely, the un-rejection graph $\mathsf{UnrejGr}$. First, we give some easy corollaries of Theorem 4.4 for the complexity measures we first discussed in Section 3.1 (completing the discussion surrounding Observation 3.2).

**Corollary 4.12.** *The type-to-one's-own-match complexity of* $\mathsf{IPDA}$ *is* $\widetilde{\Theta}(|\mathcal{I}|)$. *The type-to-another's-match complexity of both* $\mathsf{APDA}$ *and* $\mathsf{IPDA}$ *is* $\widetilde{\Theta}(|\mathcal{I}|)$.

*Proof.* First, consider the type-to-one's-own-match complexity of $\mathsf{IPDA}$. Lemma 4.11 shows that if we define $\mathsf{UnrejGr}$ with $S = \{d_*\}$, then under preference $P_{d_*}$, applicant $d_*$ will match to $\mathsf{unrej}_{d_*}(P_{d_*})$ in $\mathsf{IPDA}(d_* : P_{d_*})$. Thus, $\mathsf{UnrejGr}$ with $S = \{d_*\}$ suffices to represent $f_{d_*}(\cdot, P_{-d_*})$.

Now, the type-to-another's-match complexity of $\mathsf{APDA}$ follows directly from Theorem 4.4 and the fact that $\mathsf{APDA}$ is strategyproof. For $\mathsf{IPDA}$, consider $\mathsf{UnrejGr}$ with $S = \{d_*, d_\dagger\}$, and observe that the $d_\dagger$-nodes in $\mathsf{UnrejGr} \setminus \mathsf{chain}(\mathsf{unrej}_{d_*}(P_{d_*}))$ will also correspond to $\mathsf{UnrejGr}$ defined with $S = \{d_\dagger\}$ (and $d_*$ submitting list $P_{d_*}$), with the same ordering according to $\trianglelefteq$. Thus, Lemma 4.11 shows that $\mathsf{unrej}_{d_\dagger}(P_{d_\dagger})$ in this restricted instance of $\mathsf{UnrejGr}$ suffices to give the match of $d_\dagger$ in $\mathsf{IPDA}(d_* : P_{d_*}, d_\dagger : P_{d_\dagger})$, as desired. □

Next, we explore a more involved application. We consider natural extension of the notion of the menu to a *pair* of applicants: the set of institutions to which that pair might match (before the preference list of either of the two applicants is known). A simple corollary of Theorem 4.2 shows that this set of pairs requires $\Omega(n^2)$ bits to represent for $\mathsf{TTC}$. However, the construction of $\mathsf{UnrejGr}$ in Section 4.2 (and the fact that this construction already treats $d_*$ and $d_\dagger$ symmetrically) already tells us that this set of pairs can be represented in $\widetilde{O}(n)$ bits. This next theorem shows something even stronger: that this set of pairs has a simple and natural characterization in terms of $\mathsf{UnrejGr}$. Interestingly, the proof will use nothing about $\mathsf{APDA}$ except for its strategyproofness; all other reasoning concerns combinatorial arguments regarding the representation $\mathsf{UnrejGr}$.

**Theorem 4.13.** *Fix any priorities* $Q$, *applicants* $S = \{d_*, d_\dagger\}$, *and preferences and* $P_{-S} = P_{-\{d_*, d_\dagger\}}$. *Let* $\mathsf{UnrejGr}$ *be defined with respect to* $S$ *and* $P_{-S}$ *as in Section 4. Then, for any pair* $(h_*, h_\dagger) \in \mathcal{I} \times \mathcal{I}$, *the following are equivalent:*

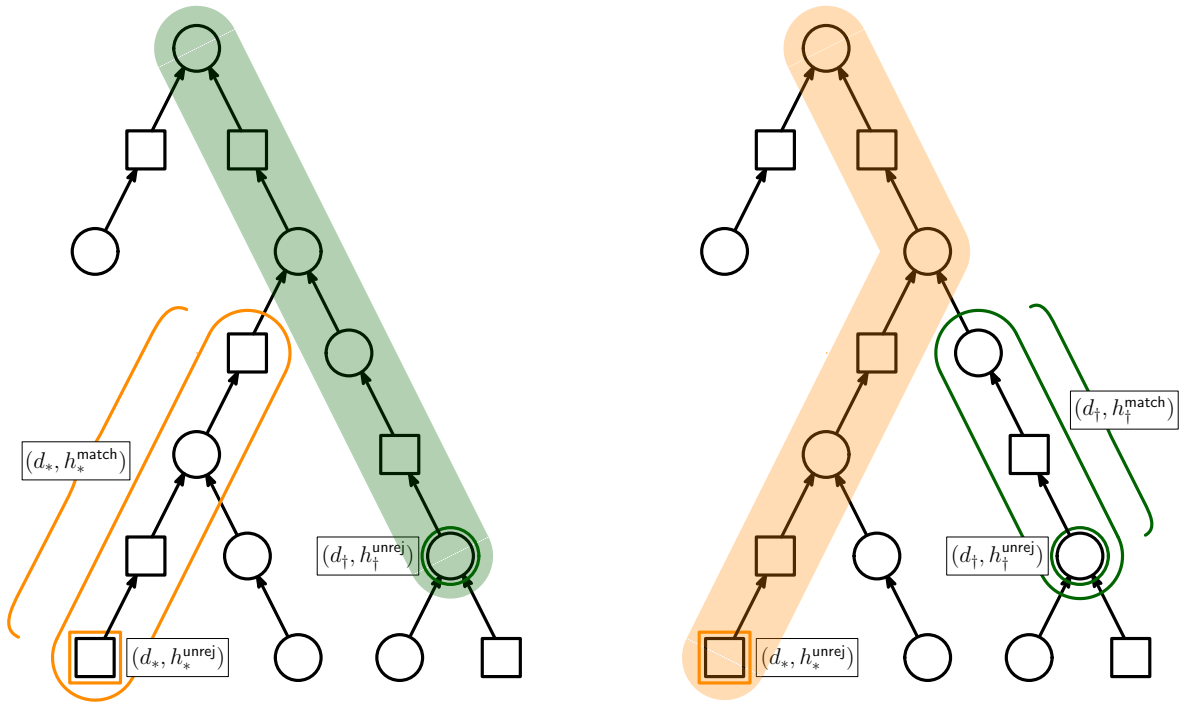


Figure 5: Case 1 of the proof of Theorem 4.13.

*(i) There exists $P_*, P_\dagger$ such that $\mu(d_*) = h_*$ and $\mu(d_\dagger) = h_\dagger$, where $\mu = \mathsf{APDA}_Q(P_*, P_\dagger, P_{-S})$.*

*(ii) $(d_*, h_*)$ and $(d_\dagger, h_\dagger)$ are nodes in* UnrejGr *and are not comparable under $\trianglelefteq$.*

*Proof.* $((ii) \implies (i))$ First, suppose that $(d_*, h_*), (d_\dagger, h_\dagger) \in \mathsf{UnrejGr}$, but are not comparable under $\trianglelefteq$. Consider $P_{d_*} = \{h_*\}$ and $P_{d_\dagger} = \{h_\dagger\}$. The results of Section 4.2 directly show that for each $i \in \{*, \dagger\}$, we have $\mathsf{unrej}_{d_i}(P_i) = h_i$, hence $h_i \in \mathsf{Menu}_{d_i}(P_{S-\{d_i\}}, P_{-S})$ by the fact that $(d_*, h_*)$ and $(d_\dagger, h_\dagger)$ are incomparable, hence $\mu(d_i) = h_i$.

$((i) \implies (ii))$ For the second and harder direction, we consider an arbitrary pair $P_{d_*}, P_{d_\dagger}$, and show that whatever $d_*$ and $d_\dagger$ match to in APDA, the corresponding vertices in UnrejGr cannot be comparable under $\trianglelefteq$. To this end, consider any $P_{d_*}, P_{d_\dagger}$ and let $\mu = \mathsf{APDA}(d_* : P_{d_*}, d_\dagger : P_{d_\dagger})$. We make the following definitions (where the two rightmost equalities will follow from the strategyproofness of APDA, and the results of Section 4.2):

$$h_*^{\mathsf{unrej}} \stackrel{\text{def}}{=} \mathsf{unrej}_{d_*}(P_{d_*}) \qquad h_*^{\mathsf{match}} \stackrel{\text{def}}{=} \mu(d_*) = \max_{P_{d_*}} \left(\mathsf{nodes}_{d_*}\left(\mathsf{UnrejGr} \setminus \mathsf{chain}(d_\dagger, h_\dagger^{\mathsf{unrej}})\right)\right)$$

$$h_\dagger^{\mathsf{unrej}} \stackrel{\text{def}}{=} \mathsf{unrej}_{d_\dagger}(P_{d_\dagger}) \qquad h_\dagger^{\mathsf{match}} \stackrel{\text{def}}{=} \mu(d_\dagger) = \max_{P_{d_\dagger}} \left(\mathsf{nodes}_{d_\dagger}\left(\mathsf{UnrejGr} \setminus \mathsf{chain}(d_*, h_*^{\mathsf{unrej}})\right)\right)$$

For nodes $v, w \in \mathsf{UnrejGr}$, we say that $w$ is *above* $v$ if $v \trianglelefteq w$ (and $w$ is *below* $v$ if $w \trianglelefteq v$). In words, $h_*^{\mathsf{match}}$ is determined by first removing every node from UnrejGr which is above $(d_\dagger, h_\dagger^{\mathsf{unrej}})$, then taking the maximum-ranked $d_*$ node according to $P_{d_*}$; vice-versa holds for $h_\dagger^{\mathsf{match}}$. The remainder of the proof proceeds in two cases based on $h_*^{\mathsf{unrej}}$ and $h_\dagger^{\mathsf{unrej}}$.

**(Case 1: $h_*^{\mathsf{unrej}}$ and $h_\dagger^{\mathsf{unrej}}$ are incomparable.)** Suppose that neither $h_*^{\mathsf{unrej}} \trianglelefteq h_\dagger^{\mathsf{unrej}}$ nor $h_\dagger^{\mathsf{unrej}} \trianglelefteq h_*^{\mathsf{unrej}}$. By the definition of $\mathsf{unrej}_d$, we know that $h_*^{\mathsf{unrej}} \succeq_{d_*}^{P_{d_*}} h$ for all $h \in \mathsf{nodes}_{d_*}(\mathsf{UnrejGr})$ where we do not have $(d_*, h_*^{\mathsf{unrej}}) \trianglelefteq (d_*, h)$. On the other hand, compared to $\mathsf{nodes}_{d_*}(\mathsf{UnrejGr})$,

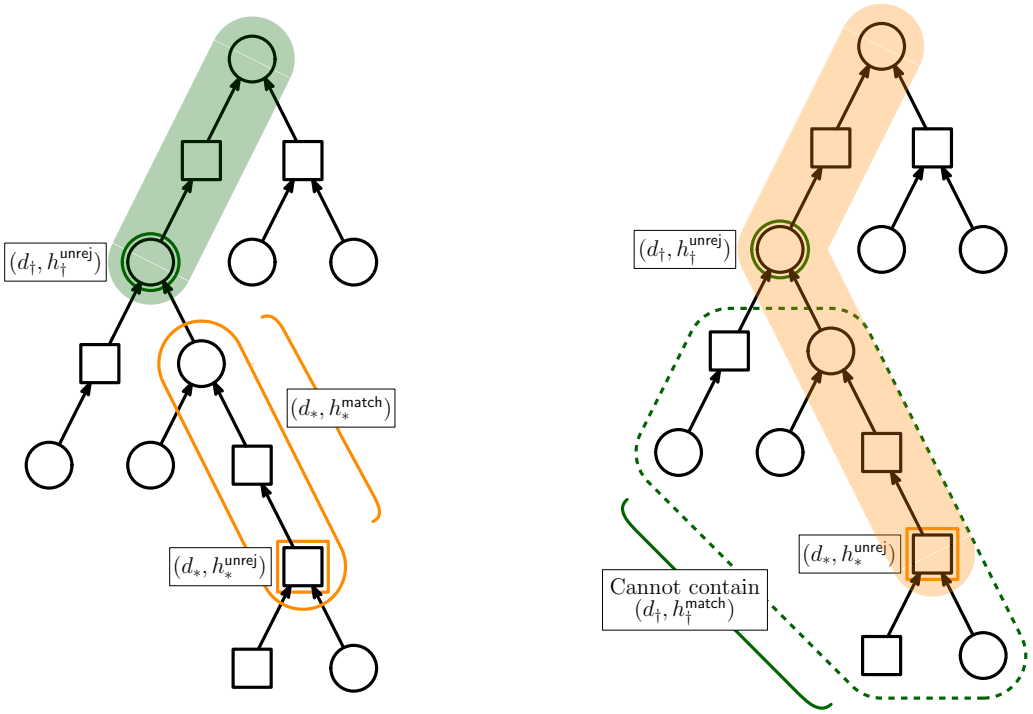


Figure 6: Case 2 of the proof of Theorem 4.13.

the set $\mathsf{Menu}_{d_*}(P_\dagger, P_{-S})$ only removes some subset $\mathsf{nodes}_{d_*}(S)$, where $S$ is some (possibly empty) subset of nodes strictly above $(d_*, h)$ according to $\trianglelefteq$. So $(d_*, h_*^{\mathsf{match}})$ will be weakly above $(d_*, h_*^{\mathsf{unrej}})$ and strictly below any common upper bound of $(d_*, h_*^{\mathsf{unrej}})$ and $(d_\dagger, h_\dagger^{\mathsf{unrej}})$, if such an upper bound exists. Dually, $(d_\dagger, h_\dagger^{\mathsf{match}})$ will be weakly above $(d_*, h_*^{\mathsf{unrej}})$ and strictly below any common upper bound of $(d_*, h_*^{\mathsf{unrej}})$ and $(d_\dagger, h_\dagger^{\mathsf{unrej}})$. Thus, $(d_*, h_*^{\mathsf{match}})$ and $(d_\dagger, h_\dagger^{\mathsf{match}})$ cannot be comparable in $\mathsf{UnrejGr}$, as desired. This case is illustrated in Figure 5.

**(Case 2: $h_*^{\mathsf{unrej}}$ and $h_\dagger^{\mathsf{unrej}}$ are comparable.)** Suppose without loss of generality that $h_*^{\mathsf{unrej}} \trianglelefteq h_\dagger^{\mathsf{unrej}}$. By the same logic as in the previous case, we know that $(d_*, h_*^{\mathsf{match}})$ will be weakly above $(d_*, h_*^{\mathsf{unrej}})$ and strictly below $(d_\dagger, h_\dagger^{\mathsf{unrej}})$. However, in this case, neither $h_\dagger^{\mathsf{unrej}}$ nor any institution in $\mathsf{nodes}_{d_\dagger}(\mathsf{chain}(d_\dagger, h_\dagger^{\mathsf{unrej}}))$ will be in $\mathsf{Menu}_{d_\dagger}(P_*, P_{-S})$. Thus, consider the favorite $d_\dagger$ node according to $P_\dagger$ which is outside of $\mathsf{chain}(d_\dagger, h_\dagger^{\mathsf{unrej}})$, and call the corresponding institution $h^{\mathsf{second}}$. That is:

$$h^{\mathsf{second}} = \max_{P_{d_\dagger}} \left( \mathsf{nodes}_{d_\dagger} \left( \mathsf{UnrejGr} \setminus \mathsf{chain}(d_\dagger, h_\dagger^{\mathsf{unrej}}) \right) \right).$$

Observe that we cannot have $(d_\dagger, h^{\mathsf{second}}) \trianglelefteq (d_\dagger, h_\dagger^{\mathsf{unrej}})$, as then we would have $h_\dagger^{\mathsf{unrej}} \in \mathsf{stab}_{d_\dagger}(P_\dagger)$, contradicting the definition of $\mathsf{unrej}_{d_\dagger}(P_\dagger) = h_\dagger^{\mathsf{unrej}}$. Since every node in $\mathsf{chain}(d_*, h_*^{\mathsf{unrej}})$ is comparable to $(d_\dagger, h_\dagger^{\mathsf{unrej}})$ in this case, we know that $(d_\dagger, h^{\mathsf{second}}) \in \mathsf{UnrejGr} \setminus \mathsf{chain}(d_*, h_*^{\mathsf{unrej}})$, and thus $h^{\mathsf{second}} \in \mathsf{Menu}_{d_\dagger}(P_*, P_{-S})$. Thus, in fact we have $h_\dagger^{\mathsf{match}} = h^{\mathsf{second}}$. All told, we know that $(d_*, h_*^{\mathsf{match}})$ will be below $(d_\dagger, h_\dagger^{\mathsf{unrej}})$, but that $(d_\dagger, h_\dagger^{\mathsf{match}})$ can neither be below nor above $(d_\dagger, h_\dagger^{\mathsf{unrej}})$. Thus, $(d_*, h_*^{\mathsf{match}})$ and $(d_\dagger, h_\dagger^{\mathsf{match}})$ cannot be comparable in $\mathsf{UnrejGr}$, as desired. This case is illustrated in Figure 6. □

Now, having proven this characterization, we discuss a connection between the results in Section 4.2 and the work of [GHT22]. We first briefly recall the relevant results of [GHT22] and their motivation. First, we rephrase Lemma 4.5 in terms of a way to describe (to each applicant separately) their match in APDA.

**Definition 4.14** (Equivalent to [GHT22, Description 1])**.** For any $d \in \mathcal{A}$, define $D_d : \mathcal{T} \to \mathcal{I} \cup \{\emptyset\}$ as follows:[16]

$$D_d^Q(P) = \max_{P_d} \left\{ h \in \mathcal{A} \;\middle|\; d \text{ receives a proposal from } h \text{ in } \mathsf{IPDA}_Q(d : \emptyset, P_{-d}) \right\}.$$

Then, by Lemma 4.5 and the fact that APDA is strategyproof, we know that for each $d \in \mathcal{A}$, priorities $Q$, and preferences $P$, we have $\mathsf{APDA}_d^Q(P) = D_d^Q(P)$.

[GHT22] are interested in $D_d^Q(\cdot)$ as an alternative way to describe APDA that might make the strategyproofness of APDA more clear. Indeed, to see that $D_d^Q(\cdot)$ is strategyproof, all $d$ has to observe is that her own report cannot effect the set of proposals she receives in $\mathsf{IPDA}_Q(d : \emptyset, P_{-d})$, and that submitting her true preference ranking $P_d$ always matches her to her highest-ranked obtainable institution. As discussed in Section 3.3, [GHT22] also prove that (in a perhaps surprising contrast to TTC) traditional descriptions of APDA cannot suffice to obtain a description whose strategyproofness is easy to see in this way.

However, note that in contrast to traditional descriptions of mechanisms, $D_d^Q$ does not describe the matching of agents other than $d$. In particular, the matching $\mathsf{IPDA}^Q(d : \emptyset, P_{-d})$ does not give the match of agents other than $d$, and the match of another applicant $d'$ is described by $D_{d'}^Q$. This may raise concerns with the description $D_d^Q$: while an agent can easily observe strategyproofness, she cannot easily see that, for example, the matching that results from this description is a feasible (one-to-one) matching, something that is clear under the traditional description. In fact, [GHT22] go on to formalize a tradeoff between conveying the menu (and hence strategyproofness) or conveying the matching (and hence one-to-one); we defer to [GHT22, Section 6] for details.

Interestingly, our Theorems 4.4 and 4.13 can serve to address this tradeoff to some theoretical degree. Namely, we show next that our theorems provide a direct way for applicants to observe that, if $D_d^Q(\cdot)$ is run separately for each applicant $d$, then no pair of applicants will match to the same institution. In particular, observe that every aspect of the proof of Theorems 4.4 and 4.13 could have been directly phrased in terms of $D_{d_*}^Q, D_{d_\dagger}^Q$, since these results exclusively argue about runs of IPDA of the form given by $D_{d_*}^Q, D_{d_\dagger}^Q$. Thus, the characterization of the menu (and "pairwise menu") provided by Theorems 4.4 and 4.13 also characterizes the set of institutions appearing in Definition 4.14. This allows us to prove the following:

**Corollary 4.15.** *For any priorities $Q$ and preferences $P$, and any two applicants $d_*$ and $d_\dagger$, we have $D_{d_*}^Q(P) \neq D_{d_\dagger}^Q(P)$.*

*Proof.* Consider UnrejGr defined with respect to $S = \{d_*, d_\dagger\}$ and $P_{-\{d_*, d_\dagger\}}$. Observe that for any fixed $h \in \mathsf{nodes}_{d_*}\mathsf{UnrejGr} \bigcap \mathsf{nodes}_{d_\dagger}\mathsf{UnrejGr}$, if we have $d_* \succ_h d_\dagger$, then we must have $(d_*, h) \trianglelefteq (d_\dagger, h)$ by the definition of $\trianglelefteq$ and $\mathsf{chain}(\cdot)$. Theorem 4.13 then shows that $d_*$ and $d_\dagger$ will not simultaneously match to $h$ under any possible $P_{d_*}$ and $P_{d_\dagger}$. Thus, $d_*$ and $d_\dagger$ cannot simultaneously match to the same institution according to $D_{d_*}^Q, D_\dagger^Q$, as desired. □

[16] [GHT22] phrase their "Description 1" in terms of running IPDA in the market not including $d$ at all. However, it is immediate to see that their definition is equivalent to this one.

While these results are complicated, when phrased in terms of $D^Q_{d_*}$ and $D^Q_{d_\dagger}$, they *only* rely on the fact that IPDA is independent of the order in which proposals are chosen (Corollary A.4), since the usage of the strategyproofness of APDA in the proof of Theorem 4.13 is replaced with the use of $\max_{P_d}$ directly in the definition of $D^Q_d$. Thus, the fact that $\{D^Q_d\}_{d\in\mathcal{A}}$ gives a one-to-one matching can be directly proven from the way the description $D^Q_d$ itself operates. That being said, we cannot imagine how such a proof could be relayed to laypeople in its current form, so this does not diminish the message of [GHT22] in terms of the tradeoff in explainability to real-world participants.

# 5 Outcome Complexities

We now switch gears and enter into the second part of our paper, designed to measure complexities inherent in the outcomes of TTC and DA. More explicitly, recall Question 2 from Section 1: After all the reports to the mechanism are known, how much communication is required to represent or verify certain information about the output matching?

## 5.1 Model

We now discuss the model needed to formulate our protocols and state our results for Section 5. Recall that the set of applicants is $\mathcal{A}$ and the institutions is $\mathcal{I}$. This model has two major components, which we phrase as two assumptions:

(Assumption 1) We assume there are many more applicants than institutions, specifically, $|\mathcal{A}| \geq |\mathcal{I}|^C$ for some fixed, arbitrary $C > 1$.

(Assumption 2) We assume the priority lists of the institutions are a fixed piece of data that defines the mechanism being used, and are thus common knowledge to all applicants.

See Figure 7 for an illustration. In Section 1, we discussed why these assumptions are plausible as stylized models of real matching mechanism environments (e.g. in school choice settings). Here, we discuss why these two assumptions are *necessary* in order to make our questions nontrivial and distinct from prior work [Seg07, GNOR19]. Our two goals will be to measure the complexity of concurrently representing to each applicant her own match (Section 5.2), and the complexity of additionally jointly verifying that the matching is correct (Section 5.3).

**(Assumption 1)** First, we consider a many-to-one matching market with $|\mathcal{A}| \gg |\mathcal{I}|$. This means that each $h \in \mathcal{I}$ can match with multiple distinct applicants, up to some **capacity** $q_h \in \mathbb{Z}_{\geq 1}$. See Appendix A for the standard extension of the definitions of all the matching mechanisms we consider to many-to-one markets (which simply consider $q_h$ identical copies of $h$ for each $h \in \mathcal{I}$).

While having $|\mathcal{A}| \gg |\mathcal{I}|$ is very natural, it's somewhat unlikely that real markets like those in school choice will literally have $|\mathcal{A}|$ scaling like $|\mathcal{I}|^C$ asymptotically; for instance, this implicitly assumes that as the number of students increases, more and more students will attend each school. However, this stylized assumption serves to unambiguously nail down whether a given complexity measure depends on $|\mathcal{A}|$ or $|\mathcal{I}|$. For example, if we consider balanced markets with $n = |\mathcal{A}| = |\mathcal{I}|$, then the questions regarding concurrent representation (Section 5.2) become trivial: an $\widetilde{O}(n)$-bit representation is achievable simply by writing down the matching, and an $\Omega(n)$ lower bound is not hard to come up with, missing much of the structural "action."

**(Assumption 2)** Second, we assume that the priorities of the institutions are fixed and part of the matching rule $f$ (e.g., $f = \mathsf{APDA}_Q$ with priorities $Q$). This means that each applicant knows ahead of time a $\widetilde{\Theta}\big(|\mathcal{A}| \cdot |\mathcal{I}|\big)$ data structure that contains every institutions' priority rankings over every student. It is not hard to show that, if the applicants know nothing about the priorities,

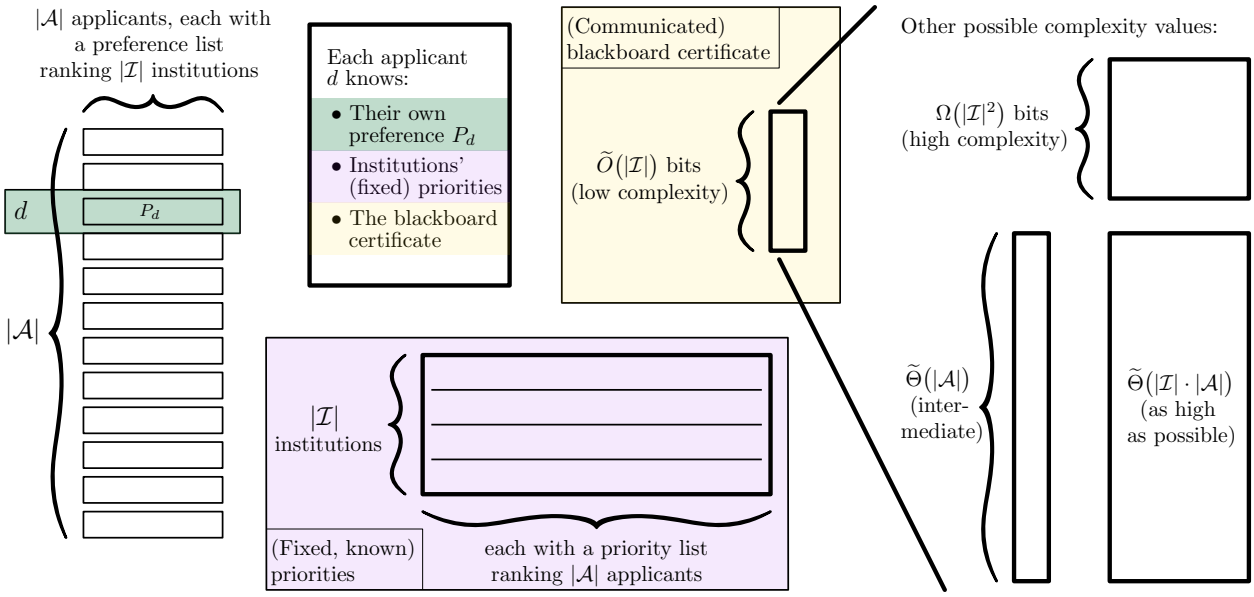


Figure 7: Illustration of the different components in our model.

**Notes:** There is a set of applicants $\mathcal{A}$ and a set of institutions $\mathcal{I}$, where $|\mathcal{A}| \gg |\mathcal{I}|$. The complexity of a protocol is the maximum number of bits that need to be written on the blackboard, i.e., the logarithm of the number of certificates the protocol might require. We think of an $\widetilde{O}(|\mathcal{I}|)$-bit protocol as potentially practical / low complexity, and of an $\Omega(|\mathcal{I}|^2)$-bit protocol as impractical / high complexity. A complexity of $\widetilde{\Theta}(|\mathcal{A}|)$ can be thought of as intermediate: it may be much less or much more than $|\mathcal{I}|^2$ depending on the relative values of $|\mathcal{A}|$ and $|\mathcal{I}|$.

then the questions of representation (Section 5.2) become trivial, and the optimal protocols for both TTC and DA require $\widetilde{\Theta}(|\mathcal{A}|)$ bits.[17] Moreover, regarding verification (Section 5.3), [Seg07, GNOR19] prove a $\Omega(|\mathcal{I}|^2)$ lower bound if the priorities must be communicated.

Assuming that the priorities are common knowledge is a natural stylized version of real markets such as school choice mechanisms, where the priorities are determined by pre-committed policies which are typically publicly available information for each school. Moreover, assuming complete common knowledge of the priorities only makes our lower bounds stronger. But the number one reason we make this modeling assumption is that it is the most *generic* way to handle the fact that some knowledge of the priorities must be present before the mechanism is run: we do not consider the institutions to be agents actively participating in the protocol or reporting their priorities, so we treat these priorities as fixed.[18]

Despite this, we note that all of our protocols are also possible under a different modeling assumption. This is the model where priorities are described by "scores" $e_h^d \in [0,1]$ and $d \succ_h d'$ if and only if $e_h^d > e_h^{d'}$, as mentioned in Section 1. All of the protocols we construct can be implemented in the same complexity if each applicant $d$ starts off knowing (and trusting) their scores $\{e_h^d\}_{h \in \mathcal{I}}$ at each institutions. Thus, each applicant only needs to start off knowing a $\widetilde{O}(|\mathcal{I}|)$

[17] For example, consider two institutions, $h^T$ and $h^B$, each with capacity $k$. Suppose there are $2k$ applicants, and every applicant prefers $h^T$ to $h^B$. Then in both mechanisms, the top half of the applicants will be matched to $h^T$. But then, to describe every applicant's match, at least one bit per applicant is required to tell applicants whether they are in the top half or the bottom half of $h^T$'s priority list.

[18] We also remark that, to keep our model and paper theoretically cohesive, all of the lower bounds in Section 3 and 4 construct a fixed set of priorities where the lower bound holds, so these results also hold for the model where priorities are common knowledge. In contrast, our positive results for the type-to-menu complexity of DA in Section 4.2 hold even when the priorities are unknown.

data structure, not an $\Omega(|\mathcal{I}| \cdot |\mathcal{A}|)$ data structure. We give the details in Remark B.8.

## 5.2 Concurrent Representation Complexity

In this section, we study the complexity of the mechanism designer simply describing or representing the matching to all of the applicants. In particular, we study the task of posting some black-board certificate $C$ such that each applicant $d$ can figure out their own match using $C$ (as well as their privately known preferences $P_d$, and for the priority-based mechanisms TTC and DA, also the publicly known priorities $Q$).

**Definition 5.1.** A *(concurrent) representation protocol* of a matching mechanism $f$ is a profile of functions $(\mathsf{match}_d)_{d\in\mathcal{A}}$, where for each $d$, $\mathsf{match}_d : \mathcal{T}_d \times \mathcal{C} \to \mathcal{I}$ for some set $\mathcal{C}$, with the following property: for every $t = (P_d)_{d\in\mathcal{A}} \in \mathcal{T}$, there exists $C \in \mathcal{C}$ such that if $\mu = f(t)$, then $\mu(d) = \mathsf{match}_d(P_d, C)$ for each $d \in \mathcal{A}$. We call an element $C \in \mathcal{C}$ a *certificate*, and if $\mu(d) = \mathsf{match}_d(P_d, C)$ for all $d \in \mathcal{A}$, we say that $\mu$ is the matching *induced by* certificate $C$ in the protocol.

The *cost* of a representation protocol is $\log_2 |\mathcal{C}|$. The *(concurrent) representation complexity* of a mechanism $f$ is the minimum of the costs of all concurrent representation protocols for $f$.

Observe that there is always a trivial approach to constructing a concurrent representation protocol, by simply unambiguously describing the full matching in the certificate $C \in \mathcal{C}$. This amounts to writing the institution each applicant is matched to, using $|\mathcal{A}| \log |\mathcal{I}| = \widetilde{O}\big(|\mathcal{A}|\big)$ bits.

For stable matching mechanisms, known results imply that a large savings over this trivial solution is possible in any situation where $|\mathcal{A}| \gg |\mathcal{I}|$. In fact, the representation used is simply a restatement of the definition of stability, and thus suffices to represent any stable matching (regardless of the mechanism $f$), and uses only $\widetilde{O}\big(|\mathcal{I}|\big)$ bits. This is somewhat remarkable: the match of each applicant can be simultaneously conveyed to the applicants in less than a single bit per applicant, assuming applicants have complete access to the priorities. To conveniently state and discuss this result, we state the following definition and lemma:

**Definition 5.2.** Given priorities $Q$ and a matching $\mu$, define an applicant $d$'s *stable budget set* as:

$$\mathsf{StabB}_d^Q(\mu) = \big\{h \in \mathcal{I} \;\big|\; \text{either } |\mu(h)| < q_h \text{ and } d \succ_h \emptyset, \\ \text{or } |\mu(h)| = q_h \text{ and there is some } d' \in \mu(h) \text{ such that } d \succ_h d'\big\}.$$

In particular, for one-to-one matching markets, $\mathsf{StabB}_d^Q(\mu)$ is the set of all $h$ such that $d \succ_h \mu(h)$, and Definition 5.2 is the natural extension of this definition to many-to-one markets.

**Observation 5.3.** *If $\mu$ is any stable matching, then every applicant $d$ is matched to the institution they rank highest in the set $\mathsf{StabB}_d^Q(\mu)$.*[19]

*Proof.* This follows directly from the definition of a stable matching: if $d$ preferred some $h \in \mathsf{StabB}_d^Q(\mu)$ to her match in $\mu$, then $(d, h)$ would block $\mu$ and $\mu$ could not possibly be stable. □

**Observation 5.4.** *The concurrent representation complexity of any stable matching mechanism (including* APDA *and* IPDA*) is $\widetilde{\Theta}\big(|\mathcal{I}|\big)$.*

[19] This observation is not novel. Besides simply restating the definition of stability, this observation, as well as the concurrent representation protocol constructed in Observation 5.4, is essentially equivalent to the "market-clearing cutoffs" characterization of stable matchings in [AL16].

*Proof.* First, we prove the upper bound. Consider any matching $\mu$ that is stable in some market with priorities $Q$ and preferences $P$. For each institution $h$ such that $|\mu(h)| = q_h$, let $d_h^{\mathsf{min}}$ denote the applicant matched to $h$ in $\mu$ with lowest priority at $h$. If $|\mu(h)| < q_h$, then define $d_h^{\mathsf{min}} = \emptyset$. Now, note that we have $\mathsf{StabB}_d^Q(\mu) = \{h | d \succ_h d_h^{\mathsf{min}}\}$. Thus, a certificate $C \in \mathcal{C}$ can simply record, for each institution $h$, the identity of $d_h^{\mathsf{min}} \in \mathcal{A} \cup \{\emptyset\}$. This requires $|\mathcal{I}| \log |\mathcal{A}| = \widetilde{\Theta}\big(|\mathcal{I}|\big)$ bits. Since each applicant $d$ knows her priority at each institution, she knows the set $\{h \mid d \succ d_h^{\mathsf{min}}\} = \mathsf{StabB}_d^Q(\mu)$. By Observation 5.3, her match in $\mu$ is then her highest-ranked institution in this set, so each $d$ can figure out her match $\mu(d)$.

A matching lower bound is not hard to construct. Consider a market with $n+1$ institutions $h^B, h_1, \ldots, h_n$ such that the capacity of $h^B$ is $n$, and the capacity of each $h_i$ is 1. Consider applicants $d_1, \ldots, d_n, d'_1, \ldots, d'_n$, and for each $i$, let priorities be such that $d_i \succ_{h_i} d'_i$. Consider preference lists such that each $d'_i$ always prefers $h_i \succ_{d'_i} h^B$, and each $d_i$ may independently prefer $h^B \succ h_i$ or $h_i \succ h^B$. In any stable matching, $d'_i$ is matched to $h^B$ if and only if $d_i$ ranks $h_i \succ h^B$. Thus, the certificate $C$ must contain at least $n$ bits of information, so $|\mathcal{C}| \geq 2^n$ and the concurrent representation complexity of $\Omega(n) = \Omega\big(|\mathcal{I}|\big)$. □

For TTC, the situation is more nuanced. Prior work [LL21] has shown that it is possible to represent the matching with $O\big(|\mathcal{I}|^2\big)$ "cutoffs", one for each pair of institutions. This can potentially save many bits off of the trivial solution that requires $\widetilde{\Theta}\big(|\mathcal{A}|\big)$ bits, but only when $|\mathcal{A}| \gg |\mathcal{I}|^2$. In other words, [LL21] prove an upper bound of $\widetilde{O}\big(\min(|\mathcal{A}|, |\mathcal{I}|^2)\big)$. However, until our work, no lower bounds were known.[20] The first technical contribution of this part of our paper is a matching lower bound.

**Theorem 5.5.** *The concurrent representation complexity of* TTC *is* $\widetilde{\Theta}\big(\min(|\mathcal{A}|, |\mathcal{I}|^2)\big)$.

*Proof.* An upper bound of $\widetilde{O}\big(|\mathcal{A}|\big)$ is trivially, by allowing the certificate posted by the protocol to uniquely define the entire matching. The $\widetilde{O}\big(|\mathcal{I}|^2\big)$ upper bound is constructed by [LL21, Theorem 1]. Very briefly, [LL21] use the trading and matching process calculating the TTC outcome to prove that for any $\mu = \mathsf{TTC}_Q(P)$, there exist applicants $\{d_{h_1}^{h_2}\}_{h_1,h_2 \in \mathcal{I}}$ such that for all applicants $d$, we have $\mu(d) = \max_{P_d}\{h_2 \in \mathcal{I} \mid d \succeq_{h_1} d_{h_1}^{h_2} \text{ for some } h_1 \in \mathcal{I}\}$. (Informally, this means that each applicant $d$ can use her priority at institution $h_1$ to gain admission to institution $h_2$ whenever $d \succ_{h_1} d_{h_1}^{h_2}$; see [LL21] for details.) Since the priorities are common knowledge, a concurrent representation protocol can thus communicate to all applicants their match by only communicating $\{d_{h_1}^{h_2}\}_{h_1,h_2 \in \mathcal{I}}$, which requires $\widetilde{O}\big(|\mathcal{I}|^2\big)$ bits.

To finish this proof, it suffices to consider the case where $|\mathcal{I}| = n$ and $|\mathcal{A}| = n^2$, and prove a lower bound of $\Omega(n^2)$. To see why this case suffices, observe that if $|\mathcal{I}|^2 > |\mathcal{A}|$, then one can apply this construction with $\sqrt{|\mathcal{A}|}$ of the elements of $\mathcal{I}$ to get a lower bound of $\Omega\big(|\mathcal{A}|\big)$, and if $|\mathcal{I}|^2 < |\mathcal{A}|$, one can ignore the surplus elements of $\mathcal{A}$ and apply this construction to get a lower bound of $\Omega\big(|\mathcal{I}|^2\big)$.

Fix $k$, where we will take $n = \Theta(k)$. Consider a matching market with institutions $h_1^T, \ldots, h_k^T$, $h_1^L, \ldots, h_k^L$, and $h_1^R, \ldots, h_k^R$, and applicants $\{d_{i,j}^T\}_{i,j \in \{1,\ldots,k\}}$, $\{d_{i,j}^L\}_{i,j \in \{1,\ldots,k\}}$, and $\{d_{i,j}^R\}_{i,j \in \{1,\ldots,k\}}$. Mnemonically, these are the top, left, and right participants. The capacity of each of these institutions is $k$. The preferences and priorities are fixed for every participant except the top applicants

[20] [LL21] mention some impossibility results for TTC, but do not formulate or prove any sort of $\Omega\big(|\mathcal{I}|^2\big)$ lower bound. For details of their arguments, see Section C.2.

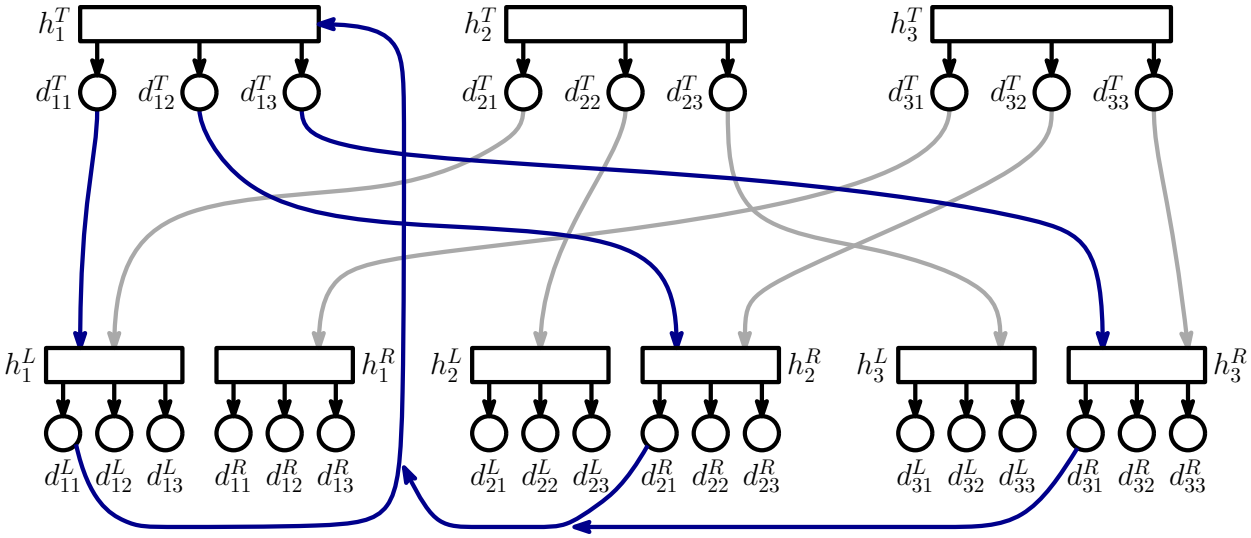


Figure 8: Illustration of the matching market used in the proof of Theorem 5.5. The three cycles matching applicants to $h_1^T$ are colored in dark blue.

$\{d_{i,j}^T\}_{i,j\in\{1,\dots,k\}}$. These fixed preferences and priorities are:

$$h_i^{\mathcal{S}} : d_{i,1}^{\mathcal{S}} \succ d_{i,2}^{\mathcal{S}} \succ \ldots \succ d_{i,k-1}^{\mathcal{S}} \succ d_{i,k}^{\mathcal{S}} \qquad \text{For each } \mathcal{S} \in \{T, L, R\} \text{ and } i \in \{1, \dots, k\}$$
$$d_{i,j}^{\mathcal{S}} : h_j^T \succ h_i^{\mathcal{S}} \qquad \text{For each } \mathcal{S} \in \{L, R\} \text{ and } i, j \in \{1, \dots, k\}$$

Now, the preferences of the $k^2$ applicants $d_{i,j}^T$ depend on $k^2$ bits $b_{i,j} \in \{L, R\}$, one for each applicant, as follows:

$$d_{i,j}^T : h_j^{b_{i,j}} \qquad \text{For each } i, j \in \{1, \dots, k\}$$

This family of matching markets is illustrated in Figure 8.

We will show next that for each $i, j$, the matches of $d_{j,i}^L$ and $d_{j,i}^R$ are determined by the bit $b_{i,j}$. Since the preferences of these applicants are fixed (and thus cannot provide any information about $b_{i,j}$ to the function $\mathsf{match}_d$ in the concurrent representation protocol), this means that the certificate $C$ must contain the information about all bits $\vec{b}$.

**Lemma 5.6.** *For each $i, j \in \{1, \dots, k\}$, if $B = b_{i,j}$, then applicant $d_{j,i}^B$ is matched to $h_i^T$ (and if $\{C\} = \{L, R\} \setminus \{B\}$, then $d_{j,i}^C$ is matched to $h_j^C$).*

First consider each $h_1^T$. First, they will point to $d_{1,1}^T$, who will point to $h_1^{b_{1,1}}$, resulting in a cycle matching $d_{1,1}^{b_{1,1}}$ to $h_1^T$. Next, a cycle with $d_{1,2}^T$ and $d_{2,1}^{b_{1,2}}$ forms, etc., until a cycle with $d_{1,k}^T$ and $d_{k,1}^{b_{1,k}}$ forms; each of these cycles match the involved (non $d_{i,j}^T$) applicant to $h_1^T$. Institution $h_1^T$'s capacity is now filled and the other applicants of the form $d_{i,1}^{\mathcal{S}}$ for $\mathcal{S} \in \{L, R\}$ who were not matched in this process then match to the corresponding $h_i^{\mathcal{S}}$. Next, $h_2^T$ can match in a similar way to each $d_{i,2}^{b_{2,i}}$ for $i = 1, \dots, k$. This continues for each top hospital until $h_k^T$ in a similar way. This proves Lemma 5.6.

Thus, for each distinct profile of bits $\{b_{i,j}\}_{i,j\in\{1,\dots,k\}}$, a concurrent representation protocol must present a distinct certificate $C$ (or otherwise, the applicants $d_{i,j}^L$ could not possibly all know their matches). This proves that the concurrent representation complexity of $\mathsf{TTC}$ is $\Omega(n^2)$, concluding the proof. □

## 5.3 Joint Verification Complexity

The concurrent representation protocols in Section 5.2 are in some sense reminiscent of nondeterministic protocols: the mechanism designer posts a certificate $C$ that all applicants can see, and we are not concerned with how the mechanism designer found the certificate $C$. However, the goal of these protocols is different than traditional nondeterministic protocols in computer science: the protocols of Section 5.2 only aim to describe a matching to the applicants, not to verify that the matching is correct. We now enhance Definition 5.1 to investigate the prospect of additionally verifying the matching.

**Definition 5.7.** A *verification protocol* of a matching mechanism $f$ is a profile of pairs of functions $\big((\mathsf{match}_d, \mathsf{check}_d)\big)_{d\in\mathcal{A}}$, where $\mathsf{match}_d : \mathcal{T}_d \times \mathcal{C} \to \mathcal{I}$ and $\mathsf{check}_d : \mathcal{T}_d \times \mathcal{C} \to \{0,1\}$ for some set $\mathcal{C}$, such that:

1. (Representation) $R = (\mathsf{match}_d)_{d\in\mathcal{A}}$ is a representation protocol for $f$.
2. (Completeness) For every $P = (P_d)_{d\in\mathcal{A}} \in \mathcal{T}$, there exists a $C \in \mathcal{C}$ such that $f(P)$ is the matching induced by $C$ over $R$, and $\mathsf{check}_d(P_d, C) = 1$ for each $d \in \mathcal{A}$.
3. (Soundness) For every $P = (P_d)_{d\in\mathcal{A}} \in \mathcal{T}$ and $\mu \neq f(P)$, if $C \in \mathcal{C}$ is such that $\mu$ is the matching induced by $C$ over $R$, then $\mathsf{check}_d(P_d, C) = 0$ for at least one $d \in \mathcal{A}$.

The *cost* of a verification protocol is $\log_2 |\mathcal{C}|$. The *verification complexity* of a mechanism $f$ is the minimum of the costs of all verification protocols for $f$.

Note that each applicant only needs to know her own match at the end of this protocol, but the applicants collectively verify the certificate in the protocol together. As in standard verification protocols, the trivial solution for a verification problem is to let the certificate $C$ record the full preference lists of all applicants, and let each $\mathsf{check}_d$ verify that the true preference lists $P_d$ is correctly recorded in $C$. This trivial solution requires $\widetilde{O}\big(|\mathcal{A}|\cdot|\mathcal{I}|\big)$ bits (in contrast to the concurrent representation protocols of Section 5.2, where the trivial solution requires $\widetilde{O}\big(|\mathcal{A}|\big)$ bits).

While Section 5.2 shows that the matching in DA (and, when $|\mathcal{A}| \gg |\mathcal{I}|^2$, in TTC) can represented with less than one bit per applicant, it turns out this is not possible for verification. Informally, most nontrivial verification protocols need to tell applicants information about other applicants, yielding an $\Omega(|\mathcal{A}|)$ lower bound (in contrast to the $\widetilde{\Theta}(|\mathcal{I}|)$- or $\widetilde{\Theta}(|\mathcal{I}|^2)$-bit representation protocols of Section 5.2, which only conveyed information about the institutions). Intuitively, the reason for this is that you need to tell applicants *why* they receive some match (not just which match they receive). We now formalize this intuition for both TTC and DA, and even for SD, by reducing the verification problem to a counting problem where each agent holds one bit.

**Proposition 5.8.** *The verification complexity of* SD *is* $\Omega(|\mathcal{A}|)$. *Thus, the verification complexity of* TTC *and any stable matching mechanism is* $\Omega(|\mathcal{A}|)$. *These results hold even for* $|\mathcal{I}| = O(1)$.

*Proof.* Fix $k$. We will construct a market with three institutions $\mathcal{I} = \{h^T, h^C, h^B\}$ and $2k$ applicants $\mathcal{A} = \{d_1^T, \dots, d_k^T, d_1^B, \dots, d_k^B\}$. The capacity of institution $h^C$ is $k$, and the capacities of each of $h^T$ and $h^B$ are $k+1$. The preferences of the applicants $d_1^T, \dots, d_k^T$ will be parametrized by bits $\vec{b} = (b_1, \dots, b_k) \in \{0,1\}^k$ for some $k$. Later, we will show that the verification complexity will be

$\Omega(k) = \Omega(|\mathcal{A}|)$. The preferences and priorities are:

$$h^{\mathcal{S}}: \quad d_1^T \succ \ldots \succ d_k^T \succ d_1^B \succ \ldots \succ d_k^B \qquad \text{For each } \mathcal{S} \in \{T, C, B\}.$$

$$d_j^T: \begin{cases} h^T & \text{if } b_j = 1 \\ h^C & \text{if } b_j = 0 \end{cases}$$

$$d_j^B: h^C \succ h^B \qquad \text{For each } j = 1, \ldots, k.$$

Since the priorities are each institution are the same, the outcome of both TTC and all stable matching mechanisms are the same under these preferences and priorities. We now characterize this outcome.

**Lemma 5.9.** *Suppose $f : \{0,1\}^k \to \mathcal{M}$ be defined such that $f(b)$ denotes the result of running* SD*,* TTC*, or any stable matching mechanism, with the above preferences and priorities with bits $b$. Then for any $b$, if we set $S = \sum_{i=1}^k b_i$, then $f(b)$ matches $d_1^B, \ldots, d_S^B$ to $h^C$ and matches $d_{S+1}^B, \ldots, d_k^B$ to $h^B$.*

We give the argument for SD. Initially, applicants $d_1^T, \ldots, d_k^T$ will match in order, with those $d_j^T$ where $b_j = 0$ matching to $h^C$. Since $h^C$ has capacity exactly $k$, this leaves $S$ seats open at $h^C$, which will go to the $S$ highest-priority applicants among $d_1^B, \ldots, d_k^B$. This proves Lemma 5.9.

Now, consider a verification protocol $P$ for any relevant mechanism $f$ (i.e., TTC, or any stable matching mechanism) with the above preferences and priorities. Since the match of applicants $d_1^B, \ldots, d_k^B$ are determined by $S = \sum_{i=1}^k b_i$, yet the preference lists of applicants $d_1^B, \ldots, d_k^B$ are fixed and do not depend on $b_1, \ldots, b_n$, we know that the certificate $C \in \mathcal{C}$ used by the verification protocol must unambiguously determine $S$; denote the value of $S$ corresponding to a certificate $C$ by $S(C)$. Now, the verification protocol must in particular verify that $S(C)$ is the value $\sum_{i=1}^k b_i$, so the verification protocol in particular suffices to define a $k$-player verification protocol for $\sum_{i=1}^k b_i$, where each player $i$ holds the single bit $b_i$. Informally, no protocol can perform this verification without (up to lower-order terms) specifying in $C$ precisely which players $i$ have $b_i = 1$; specifying this takes $\Omega(k)$ bits. With a fairly straightforward argument, we formalize this reduction and claim in Section B.1, showing that verifying $\sum_{i=1}^k b_i$ requires $\Omega(k) = \Omega(|\mathcal{A}|)$ bits. □

We now consider upper bounds; unlike the above lower bound, constructing a protocol is a very different task in TTC versus in DA. First, we show that Proposition 5.8 is tight for TTC, because TTC has an easy to construct *deterministic* protocol with matching communication complexity.

**Observation 5.10.** *The verification complexity of* TTC *is* $\widetilde{\Theta}(|\mathcal{A}|)$.

*Proof.* The lower bound is given in Proposition 5.8. The upper bound follows from the fact that TTC has a deterministic blackboard communication protocol. We describe this (fairly simple) protocol in Section B.2 for completeness. □

For stable matching mechanisms, giving an upper bound is less straightforward. To begin, note that verifying that a matching $\mu$ is the outcome of APDA (or IPDA) is a different and more complicated task than verifying that $\mu$ is stable. Indeed, to verify stability using $\widetilde{\Theta}(|\mathcal{A}|)$ bits, the protocol simply needs to use the certificate $C$ to write down the matching itself: since the priorities of the institutions are known to all applicants, each applicant $d$ can simply check that there is no institution $h$ such that $h \succ_d \mu(h)$ and such that $h$ prefers $d$ to one of their matches. However, this verification protocol will not suffice to verify that the outcome is the result of APDA, because if we consider any market with more than one stable matching, the above protocol will incorrectly verify a stable matching which is not the result of APDA.

Nevertheless, we prove that with only $\widetilde{O}(|\mathcal{A}|)$ bits, a protocol can not only describe the outcome of APDA (or IPDA), but also verify that the outcome is correct. Our protocol relies on classically known properties of the set of stable matchings (namely, the rotation poset / lattice properties, see [GI89]) to verify that a proposed matching is the extremal (i.e., either applicant or institution-optimal) matching within the set of stable matchings, all while using only a few bits per applicant. Since the techniques of this proof are largely outside the central themes of this paper, we defer the proof to Section B.3.

**Theorem 5.11.** *The verification complexity of both* APDA *and* IPDA *are* $\widetilde{\Theta}(|\mathcal{A}|)$.

Note that unlike TTC, we do not know the *deterministic* blackboard communication complexity of DA in our model (where priorities are common knowledge). While [Seg07, GNOR19] prove a lower bound of $\Omega(|\mathcal{I}|^2)$ when the priorities must be communicated, this model is provably different than our model, because the lower bounds of [Seg07, GNOR19] hold for the verification problem as well.

# 6 Additional Outcome Complexities

## 6.1 All-Menus Complexity

In this section, we discuss a different approach to representing the process and results of strategyproof matching mechanisms. This approach is inspired by the representation protocols for DA and TTC in Section 5.2. Both of these protocols (implicitly) communicate some set $B_d$ simultaneously to each $d \in \mathcal{A}$, and each $d$ determines their match as their top-ranked institution in $B_d$. This may seem to imply that $B_d$ should be $d$'s set of available options, i.e., $d$'s menu. However, this is not the case, as we will see below. However, it inspires one reasonable approach to representing the matching: communicate every applicants' menu, and tell applicants they are matched to their highest-ranked institution in the menu.

To capture the number of bits required to communicate all applicants' menus, we make the following definition:

**Definition 6.1.** The *all-menus complexity* of a matching mechanism $f$ over a set of applicants $\mathcal{A} = \{1, \dots, n\}$ is:

$$\log_2 \left| \left\{ \left( \mathsf{Menu}_1^f(P_{-1}), \mathsf{Menu}_2^f(P_{-2}), \dots, \mathsf{Menu}_n^f(P_{-n}) \right) \;\middle|\; P \in \mathcal{T} \right\} \right|.$$

Note that, in contrast to the models in Definition 5.1 and Definition 5.7, this definition does not assume that an applicant $d$ uses information about their own report $P_d$ to figure out $\mathsf{Menu}_d(P_{-d})$. However, this could not possibly help applicant $d$ in this task, since $\mathsf{Menu}_d(P_{-d})$ is by definition independent of the value of $P_d$. Thus, Definition 6.1 captures the complexity of representing to each applicant her own menu under the model of Section 5 as well. Also, recall that in our model, if $f$ is $\mathsf{TTC}_Q$ or $\mathsf{DA}_Q$ for some priorities $Q$, then we consider the priorities fixed and part of the mechanism. So, when we bound the all-menus complexity of TTC or DA, we mean the maximum over all possible $Q$ of the all-menus complexity of $\mathsf{TTC}_Q$ or $\mathsf{DA}_Q$.

We now consider APDA, and discuss in detail how the sets $\mathsf{StabB}_d(\mathsf{APDA}_Q(P))$ (the stable budget sets, which are implicitly communicated by the representation protocol in Observation 5.4) and $\mathsf{Menu}_d^{\mathsf{APDA}_Q}(P_{-d})$ differ. For perhaps the most crucial high-level difference, note that $d$'s own preference $P_d$ can influence $\mathsf{StabB}_d(\mathsf{APDA}_Q(P))$. For a concrete example of how these sets can differ, consider the following example (taken from the related work section of [GHT22]):

**Example 6.2** ([GHT22])**.** Let institutions $h_1, h_2, h_3, h_4$ all have capacity 1, and consider applicants $d_1, d_2, d_3, d_4$. Let the priorities and preferences be:

$$\begin{aligned} &h_1 : d_1 \succ d_2 && d_1 : h_1 \succ \dots \\ &h_2 : d_4 \succ d_3 \succ d_2 \succ d_1 && d_2 : h_1 \succ h_2 \succ h_4 \succ \dots \\ &h_3 : d_3 && d_3 : h_3 \succ \dots \\ &h_4 : d_2 \succ d_4 && d_4 : h_4 \succ h_2 \succ \dots \end{aligned}$$

Then $\mathsf{APDA}_Q(P)$ pairs $h_i$ to $d_i$ for each $i = 1, \dots, 4$. Now, for institution $h_2$, consider which applicants $d$ have $h_2 \in \mathsf{StabB}_d(\mathsf{APDA}_Q(P))$, and which applicants have $h_2 \in \mathsf{Menu}_d^{\mathsf{APDA}_Q}(P_{-d})$. First, $h_2$ is in the stable budget set of applicants $d_2, d_3$ and $d_4$, so despite $d_3$ being higher priority than $d_2$ at $h_2$, $h_2$ *is not* on $d_3$'s menu. Second, $h_2$ is in the menu of applicants $d_1, d_2$, and $d_4$, so despite $d_1$ being lower priority than $d_2$ at $h_2$, $h_2$ *is* on $d_1$'s menu.

More generally, for $\mathsf{APDA}$ the menu differs from the stable budget set in two ways. First, if $h$ is an institution who would accept a proposal from $d$, but a rejection cycle would lead to $d$ being kicked back out, then $h \in \mathsf{StabB}_d(\mathsf{APDA}) \setminus \mathsf{Menu}_d^{\mathsf{APDA}}$. Second, if $h$ is an institution such that $h$ received a proposal from $\mu(h)$ only as a result of $d$ proposing to $\mu(d)$ (and moreover, we have $\mu(h) \succ_h d \succ_h d'$, where $d'$ is the match of $h$ if $d$ submits an empty list), then it is possible that $h \in \mathsf{Menu}_d^{\mathsf{APDA}} \setminus \mathsf{StabB}_d(\mathsf{APDA})$. In other words, calculating the menu of an applicant $d$ must take into account both the fact that $d$ might hypothetically propose to some $h \neq \mu(d)$, and the fact that $d$ might no longer proposes to $\mu(d)$. Our main result in this section harnesses this intuition to show that the all-menus complexity of $\mathsf{DA}$ is high:

**Theorem 6.3.** *In a one-to-one market with $n$ applicants and $n$ institutions, the all-menus complexity of any stable matching mechanism is $\widetilde{\Theta}(n^2)$. In a many-to-one market, the all-menus complexity of any stable matching mechanism is $\Omega(|\mathcal{I}|^2)$.*

*Proof.* Let $n = 3k$. First we define the priorities $Q$. For each $i = 1, \dots, k$, there are three institutions $h_i^T, h_i^B, h_i^R$, and applicants $d_i^T, d_i^B, d_i^R$. The priorities of the institutions are, for each $i = 1, \dots, k$:

$$h_i^T : d_i^T \succ d_1^B \succ \dots \succ d_k^B \qquad h_i^B : d_i^R \succ d_1^T \succ \dots \succ d_k^T \succ d_i^B \qquad h_i^R : d_i^B \succ d_i^R$$

The preferences of the applicants are defined with respect to a family of subsets $B_i \subseteq \{h_1^T, \dots, h_k^T\}$, with one subset for each $i = 1, \dots, k$. We define $P = P(B_1, \dots, B_k)$ as follows:

$$d_i^T : h_i^T \qquad d_i^B : h_i^B \succ B_i \succ h_i^R \qquad d_i^R : h_i^R \succ h_i^B$$

(Where the elements of $B_i$ can appear in $d_i^B$'s list in any order.) These preferences and priorities are illustrated in Figure 9.

The key claim is the following:

**Lemma 6.4.** *Let $P = P(B_1, \dots, B_k)$. Then $h_j^B \in \mathsf{Menu}_{d_i^T}^{DA_Q}(P_{-d_i^T})$ if and only if $h_i^T \in B_j$.*

To prove this lemma, consider changing $d_i^T$'s list to $\{h_j^B\}$ to get a preference profile $P'$. This will cause $d_j^B$ to be rejected from $h_j^B$ and start proposing to each institution in $B_j \subseteq \{h_1^T, \dots, h_k^T\}$. Such an institution $h_u^T$ will accept the proposal from $d_j^B$ if and only if they have not received a proposal from $d_u^T$. But among $\{d_1^T, \dots, d_k^T\}$, only $d_i^T$ does not propose to $h_i^T$ in $P'$. So $d_j^B$ will propose in the end to $h_j^R$ if and only if $h_i^T \notin B_j$. If $d_j^B$ proposes to $h_j^R$, then $d_j^R$ will next propose to $h_j^B$, so $h_j^B \notin \mathsf{Menu}_{d_i^T}(P_{-d_i^T})$. If $d_j^B$ is instead accepted by $h_i^T$, then we have $h_j^B \in \mathsf{Menu}_{d_i^T}(P_{-d_i^T})$,

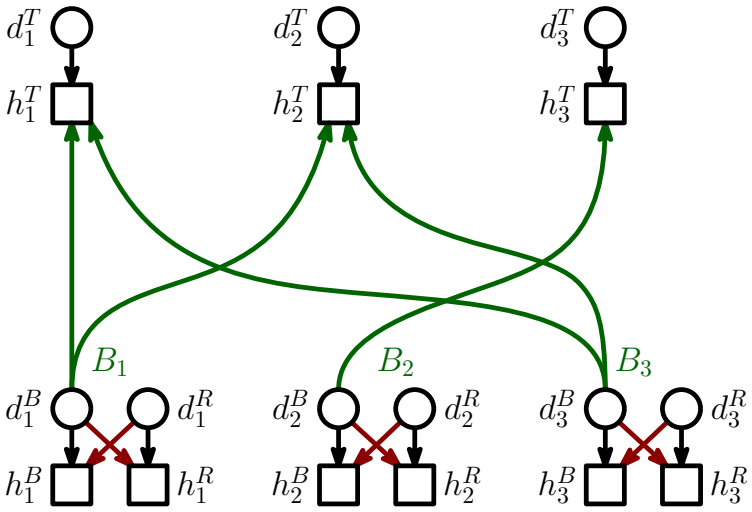


Figure 9: Illustration of the set of preferences and priorities used to show that the all-menus complexity of DA is $\Omega(n^2)$ (Theorem 6.3).

**Notes:** An applicant $d_i^T$ has institution $h_j^B$ on their menu if and only if $h_i^T \in B_j$, i.e. if and only if $d_j^B$ would propose to $h_i^T$ if they are rejected from $h_j^B$.

as desired. This proves Lemma 6.4.

Thus, there is a distinct profile of menus $\left(\mathsf{Menu}_{d_1^T}, \ldots, \mathsf{Menu}_{d_k^T}\right)$ for each distinct profile of $(B_1, \ldots, B_k)$, of which there are $2^{k^2}$ , and the all-menus complexity of APDA is $\Omega(k^2) = \Omega(|\mathcal{I}|^2)$. Moreover, since each applicants' menu is the same in all stable mechanisms Lemma A.7, the same bound holds for any stable matching mechanism. □

We remark that, if the priorities $Q$ were not considered common knowledge and thus $Q$ was not required to be fixed, then the above construction could've been much simpler. For example, we could simply let $h_i^B$ accept or not accept a proposal from each $d_j^T$ independently, such that $h_i^B$ is in the menu of $d_j^T$ if and only if $h_i^B$ will accept their proposal.[21]

It turns out that the same complexity result holds for TTC, namely, the all-menu complexity of TTC is $\Omega(|\mathcal{I}|^2)$. This is perhaps less surprising than our result for DA. For instance, consider the case where $|\mathcal{A}| \gg |\mathcal{I}|^2$, and recall that in this case, our result Theorem 5.5 shows that the representation complexity of TTC is $\Omega(|\mathcal{I}|^2)$. Moreover, for any strategyproof mechanism $f$, the all-menus complexity must be at least as high as the representation complexity (since for a strategyproof mechanism, any representation of the menus of each applicant suffices to describe to each applicant their own match). Thus, when $|\mathcal{A}| \gg |\mathcal{I}|^2$, we already know that the all-menus complexity of TTC is $\Omega(|\mathcal{I}|^2)$.

Our next result shows that the all-menus complexity of TTC is $\Omega(|\mathcal{I}|^2)$ (even in the balanced case where $|\mathcal{I}| = |\mathcal{A}|$). The construction is similar to that of Theorem 6.3, except the way that TTC works allows us to make the construction quite a bit simpler.

**Theorem 6.5.** *In a one-to-one market with $n$ applicants and $n$ institutions, the all-menus complexity of* TTC *is $\widetilde{\Theta}(n^2)$. In a many-to-one market, the all-menus complexity of any stable matching mechanism is $\Omega(|\mathcal{I}|^2)$.*

*Proof.* Let $n = 2k$. First we define the priorities $Q$. For each $i = 1, \ldots, k$, there are institutions $h_i^T, h_i^B$, and applicants $d_i^T, d_i^B$. The priorities of the institutions are, for each $i = 1, \ldots, k$:

$$h_i^T : d_i^T \qquad\qquad h_i^B : d_i^B$$

[21] In contrast, for our lower bounds in Section 3 and 4, we do not know of any significantly simpler constructions which get the same bounds by allowing the priorities $Q$ to vary.

The preferences are defined with respect to a family of subsets of $B_i \subseteq \{h_1^T, \ldots, h_k^T\}$ for each $i = 1, \ldots, k$. We define $P(B_1, \ldots, B_k)$ as follows: for each $i = 1, \ldots, k$:

$$d_i^T : h_i^T \qquad\qquad d_i^B : B_i \succ h_i^B$$

(Where the elements of $B_i$ can appear in $d_i^B$'s list in any order.)

The key claim is the following:

**Lemma 6.6.** *Let $P = P(B_1, \ldots, B_k)$. Then $h_j^B \in \mathsf{Menu}_{d_i^T}^{TTC_Q}(P)$ if and only if $h_i^T \in B_j$.*

To prove this lemma, consider changing $d_i^T$'s list to $\{h_j^B\}$ to get a preference profile $P'$. Consider now the run of TTC. After all $d_j^T$ for $j \neq i$ have been matched to the corresponding $h_j^T$, observe that $d_j^B$ will point transitively to $d_i^T$ if and only if $h_i^T \in B_j$. This is equivalent to $h_j^B$ being on $d_i^T$'s menu. This proves Lemma 6.6.

Thus, there is a distinct profile of menus $\left(\mathsf{Menu}_{d_1^T}, \ldots, \mathsf{Menu}_{d_k^T}\right)$ for each distinct profile of $(B_1, \ldots, B_k)$, of which there are $2^{k^2}$, so the all-menus complexity of TTC is $\Omega(k^2) = \Omega(|\mathcal{I}|^2)$. □

## 6.2 All-Type-To-One-Match Complexity

Finally, we investigate one additional complexity measure that lies squarely at the intersection of Question 1 and Question 2 from Section 1. Specifically, we take the "easiest / lowest" type of complexity considered under Question 1 (namely, the type-to-another's-match complexity), and investigate it under the agenda of Question 2, where information must be conveyed about all applicants simultaneously. Our definition is:

**Definition 6.7.** The *all-type-to-one-match* complexity of a matching mechanism $f$ is

$$\log_2 \max_{d_\dagger} \left| \left\{ \left( f_{d_\dagger}^1(\cdot, P_{-1}), f_{d_\dagger}^2(\cdot, P_{-2}), \ldots, f_{d_\dagger}^n(\cdot, P_{-n}) \right) \;\middle|\; P \in \mathcal{T} \right\} \right|,$$

where for each $d \in \mathcal{A} = \{1, \ldots, n\}$, the function $f_{d_\dagger}^d(\cdot, P_{-d}) : \mathcal{T}_d \to \mathcal{I}$ is such that $f_{d_\dagger}^d(P'_d, P_{-d})$ is the match of applicant $d_\dagger$ in $f(P'_d, P_d)$.

While we mostly consider this complexity measure for completeness, one interesting thing is that this measure is high, even for serial dictatorship. In particular, a direct corollary of Theorem 6.8 is that the all-type-to-one-match complexity of TTC and DA are $\Omega(n^2)$, and furthermore, that more complicated variants of Definition 6.7 (such as the complexity of representing every applicant's map from their type to the matching overall, i.e. combining Definition 6.7 with Definition 3.3) will also be $\Omega(n^2)$.

**Theorem 6.8.** *In a one-to-one market with $n$ applicants and $n$ institutions, the* all-type-to-one-match complexity *of serial dictatorship is $\widetilde{\Theta}(n^2)$. In a many-to-one market, the all-type-to-one-match complexity of serial dictatorship is $\Omega(|\mathcal{I}|^2)$.*

*Proof.* Fix $k$, where we will take $n = \Theta(k)$. Consider applicants $d_1^T, \ldots, d_k^T, d_1^B, \ldots, d_k^B, d_\dagger$, and let this order over these applicants be the (single) priority order in the serial dictatorship mechanism. Consider also institutions $h_1^T, \ldots, h_k^T, h_1^B, \ldots, h_k^B, h_\dagger$. Now, for any profiles of sets $B_1, \ldots, B_k \subseteq \{h_1^T, \ldots, h_k^T\}$, define a set of preferences $P$ such that:

$$\begin{aligned} &d_i^T : h_i^T && \text{For each } i = 1, \ldots, k \\ &d_i^B : h_i^B \succ B_i \succ h_\dagger && \text{For each } i = 1, \ldots, k \\ &d_\dagger : h_\dagger \end{aligned}$$

The key claim is the following:

**Lemma 6.9.** *If the preference list of $d_i^T$ is changed to $\{h_j^B\}$, then applicant $d_\dagger$ will match to $h_\dagger$ if and only if $h_i^T \in B_j$.*

To prove this lemma, observe that if applicant $d_i^T$ switches their preference to only rank $h_j^B$, then these two participants will permanently match. After all applicants in $\{d_1^T, \dots, d_k^T\}$ choose, only $h_i^T$ will be still unmatched. Then, when $d_j^B$ is called to pick an institution, they will pick $h_\dagger$ if and only if $h_i^T \notin B_j$. This proves Lemma 6.9.

Thus, there is a distinct profile of functions $\left(f_{d_\dagger}^{d_1^T}(\cdot, P_{-d_1^T}), \dots, f_{d_\dagger}^{d_k^T}(\cdot, P_{-d_k^T})\right)$ for each distinct profile $(B_1, \dots, B_k)$, of which there are $2^{k^2}$, so the all-type-to-one-match complexity of Serial Dictatorship is $\Omega(k^2) = \Omega(n^2) = \Omega(|\mathcal{I}|^2)$. □

In closing, we remark that in the case of many-to-one markets with $|\mathcal{A}| \gg |\mathcal{I}|$, each of the bounds in Section 6 leave a gap between a lower bound $\Omega\big(|\mathcal{I}|^2\big)$ and the trivial upper bound of $\widetilde{O}\big(|\mathcal{I}| \cdot |\mathcal{A}|\big)$.[22] Still, our bounds give qualitative negative results, since we think of $\Omega\big(|\mathcal{I}|^2\big)$ as large/complex. We also remark that none of our results in this paper are tight in terms of the exact log factors. While these log factors are small (corresponding to the need to index a single applicant/institution), getting bounds that are exactly asymptotically tight may require very different constructions that would allow the order of applicants' preference lists to vary much more dramatically than our constructions do, possibly leading to interesting technical challenges. Nevertheless, we believe that our results here carry the main economic and complexity insights for each of the questions we ask.

## References


[AG18] Itai Ashlagi and Yannai A Gonczarowski. Stable matching mechanisms are not obviously strategy-proof. *Journal of Economic Theory*, 177:405–425, 2018. Originally appeared in 2015. 1, 6

[AKL17] Itai Ashlagi, Yash Kanoria, and Jacob D. Leshno. Unbalanced random matching markets: The stark effect of competition. *Journal of Political Economy*, 125(1):69 – 98, 2017. 1, 6

[AL16] Eduardo M Azevedo and Jacob D Leshno. A supply and demand framework for two-sided matching markets. *Journal of Political Economy*, 124(5):1235–1268, 2016. 1, 3, 4, 5, 27, 46, 47, 48

[BG16] Sophie Bade and Yannai A Gonczarowski. Gibbard-satterthwaite success stories and obvious strategyproofness. In *EC*, 2016. 6

[BGG20] Moshe Babaioff, Kira Goldner, and Yannai A. Gonczarowski. Bulow-klemperer-style results for welfare maximization in two-sided markets. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2452–2471, 2020. 6

[BGN17] Moshe Babaioff, Yannai A. Gonczarowski, and Noam Nisan. The menu-size complexity of revenue approximation. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 869–877, 2017. 6


[22] The same remark applies if we consider our type-to-matching lower bounds in Section 3 in many-to-one markets. For our type-to-menu lower bounds in Section 4, one can show that our results are already tight: $\widetilde{O}(|\mathcal{I}|^2)$ suffices for TTC by nonbossiness, and $\widetilde{O}\big(|\mathcal{I}|\big)$ suffices for DA because the un-rejections graph UnrejGr in Section 4.2 has at most one node for each element of $\{d_*, d_\dagger\} \times \mathcal{I}$ (and our proofs in that section hold as written for many-to-one markets).

[BGN22] Moshe Babaioff, Yannai A. Gonczarowski, and Noam Nisan. The menu-size complexity of revenue approximation. *Games and Economic Behavior*, 134:281–307, 2022. 6

[BILW20] Moshe Babaioff, Nicole Immorlica, Brendan Lucier, and S. Matthew Weinberg. A simple and approximately optimal mechanism for an additive buyer. *J. ACM*, 67(4):24:1–24:40, 2020. 6

[BK96] Jeremy Bulow and Paul Klemperer. Auctions versus negotiations. *The American Economic Review*, pages 180–194, 1996. 6

[CFL14] Stephen A Cook, Yuval Filmus, and Dai Tri Man Lê. The complexity of the comparator circuit value problem. *ACM Transactions on Computation Theory (TOCT)*, 6(4):1–44, 2014. 1, 6

[CT19] Linda Cai and Clayton Thomas. Representing all stable matchings by walking a maximal chain. Mimeo, 2019. 20

[CT22] Linda Cai and Clayton Thomas. The short-side advantage in random matching markets. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 257–267. SIAM, 2022. 6

[DDT17] Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. Strong duality for a multiple-good monopolist. *Econometrica*, 85(3):735–767, 2017. 6

[DF81] E. L. Dubins and A. D. Freedman. Machiavelli and the gale-shapley algorithm. *The American Mathematical Monthly*, 88:485–494, 08 1981. 41, 42

[Dob16] Shahar Dobzinski. Computational efficiency requires simple taxation. In *FOCS*, 2016. 6

[DR21] Shahar Dobzinski and Shiri Ron. The communication complexity of payment computation. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 933–946, 2021. 6

[DRV22] Shahar Dobzinski, Shiri Ron, and Jan Vondrák. On the hardness of dominant strategy mechanism design. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 690–703. ACM, 2022. 6

[EFF+17a] Alon Eden, Michal Feldman, Ophir Friedler, Inbal Talgam-Cohen, and S. Matthew Weinberg. The competition complexity of auctions: A bulow-klemperer result for multi-dimensional bidders. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, page 343, 2017. 6

[EFF+17b] Alon Eden, Michal Feldman, Ophir Friedler, Inbal Talgam-Cohen, and S. Matthew Weinberg. A simple and approximately optimal mechanism for a buyer with complements. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, page 323, 2017. 6

[GHT22] Yannai A Gonczarowski, Ori Heffetz, and Clayton Thomas. Strategyproofness-exposing mechanism descriptions. *arXiv preprint arXiv:2209.13148*, 2022. 1, 2, 5, 6, 12, 13, 16, 18, 21, 24, 25, 32, 33, 48

[GI89] Dan Gusfield and Robert Irving. *The stable marriage problem: Structure and algorithms*. The MIT Press, Cambridge, MA, 1989. 5, 32, 43, 44

[GL21] Louis Golowich and Shengwu Li. On the computational properties of obviously strategy-proof mechanisms. *arXiv preprint arXiv:2101.05149*, 2021. 6

[GMRV18] Rohith Reddy Gangam, Tung Mai, Nitya Raju, and Vijay V Vazirani. A structural and algorithmic study of stable matching lattices of "nearby" instances, with applications. *arXiv e-prints*, pages arXiv–1804, 2018. 6

[GNOR19] Yannai A Gonczarowski, Noam Nisan, Rafail Ostrovsky, and Will Rosenbaum. A stable marriage requires communication. *Games and Economic Behavior*, 118:626–647, 2019. 1, 5, 6, 25, 26, 32, 45, 47

[Gon14] Yannai Gonczarowski. Manipulation of stable matchings using minimal blacklists. In *ACM Conference on Economics and Computation (EC)*, 2014. 1

[Gon18] Yannai A. Gonczarowski. Bounding the menu-size of approximately optimal auctions via optimal-transport duality. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 123–131, 2018. 6

[GS62] D. Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–14, 1962. 1, 41

[Ham79] Peter J Hammond. Straightforward individual incentive compatibility in large economies. *The Review of Economic Studies*, 46(2):263–282, 1979. 8

[HN13] Sergiu Hart and Noam Nisan. The menu-size complexity of auctions. In *the 14th ACM Conference on Electronic Commerce (EC)*, 2013. 6

[HN17] Sergiu Hart and Noam Nisan. Approximate revenue maximization with multiple items. *J. Economic Theory*, 172:313–347, 2017. 6

[HR09] Jason D. Hartline and Tim Roughgarden. Simple versus optimal mechanisms. In *ACM Conference on Electronic Commerce*, pages 225–234, 2009. 6

[IL86] Robert W Irving and Paul Leather. The complexity of counting stable marriages. *SIAM J. Comput.*, 15(3):655–667, August 1986. 1, 6

[IM05] Nicole Immorlica and Mohammad Mahdian. Marriage, honesty, and stability. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 53–62, 2005. 1, 6

[KMQ21] Yash Kanoria, Seungki Min, and Pengyu Qian. In which matching markets does the short side enjoy an advantage? In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1374–1386. SIAM, 2021. 1, 6

[Knu76] D.E. Knuth. Mariages stables et leurs relations avec d'autres problèmes combinatoires. *Montréal: Les presses de l'Université de Montréal*, 1976. 1

[Kup20] Ron Kupfer. The influence of one strategic agent on the core of stable matchings. In *WINE*, 2020. 6

[Li17] Shengwu Li. Obviously strategy-proof mechanisms. *American Economic Review*, 107(11):3257–87, 2017. 6

[LL21] Jacob D Leshno and Irene Lo. The cutoff structure of top trading cycles in school choice. *The Review of Economic Studies*, 88(4):1582–1623, 2021. 1, 4, 5, 6, 28, 46, 47, 48

[MR21] Pinaki Mandal and Souvik Roy. Obviously strategy-proof implementation of assignment rules: A new characterization. *International Economic Review*, 63(1):261–290, 2021. 6

[MV18] Tung Mai and Vijay V. Vazirani. Finding Stable Matchings That Are Robust to Errors in the Input. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 60:1–60:11, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 6

[MW71] D. G. McVitie and L. B. Wilson. The stable marriage problem. *Communications of the ACM*, 14(7), 1971. 41

[Pit89] Boris Pittel. The average number of stable matchings. *SIAM J. Discret. Math.*, 2(4):530–549, November 1989. 1, 6

[Pit92] Boris Pittel. On likely solutions of a stable marriage problem. *Ann. Appl. Probab.*, pages 358–401, 1992. 1

[PT21] Marek Pycia and Peter Troyan. A theory of simplicity in games and mechanism design. *CEPR Discussion Paper No. DP14043*, 2021. 6

[RNS21] Samantha Robertson, Tonya Nguyen, and Niloufar Salehi. Modeling assumptions clash with the real world: Transparency, equity, and community challenges for student assignment algorithms. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, 2021. 1

[Rot82a] Alvin E. Roth. The economics of matching: stability and incentives. *Mathematics of Operations Research*, 7(4):617–628, 1982. 42

[Rot82b] Alvin E Roth. Incentive compatibility in a market with indivisible goods. *Economics letters*, 9(2):127–132, 1982. 41

[Rot86] Alvin E. Roth. On the allocation of residents to rural hospitals: A general property of two-sided matching markets. *Econometrica*, 54(2):425–427, 1986. 41

[RP77] Alvin E Roth and Andrew Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4(2):131–137, 1977. 41

[RRVV93] Alvin E Roth, Uriel G Rothblum, and John H Vande Vate. Stable matchings, optimal assignments, and linear programming. *Mathematics of operations research*, 18(4):803–828, 1993. 1

[RST+21] Aviad Rubinstein, Raghuvansh R Saxena, Clayton Thomas, S Matthew Weinberg, and Junyao Zhao. Exponential communication separations between notions of selfishness. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 947–960, 2021. 6

[Seg07] Ilya Segal. The communication requirements of social choice rules and supporting budget sets. *Journal of Economic Theory*, 136(1):341–378, 2007. 1, 6, 25, 26, 32, 46, 47, 48

[SS74] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974. 1, 41

[SS15] Daniela Saban and Jay Sethuraman. The complexity of computing the random priority allocation matrix. *Mathematics of Operations Research*, 40(4):1005–1014, 2015. 1, 6

[SSW18] Raghuvansh R. Saxena, Ariel Schvartzman, and S. Matthew Weinberg. The menu complexity of "one-and-a-half-dimensional" mechanism design. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2026–2035, 2018. 6

[Sub94] Ashok Subramanian. A new approach to stable matching problems. *SIAM Journal on Computing*, 23(4):671–700, 1994. 1, 6

[Tho21] Clayton Thomas. Classification of priorities such that deferred acceptance is osp implementable. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 860–860, 2021. 6

[Tro19] Peter Troyan. Obviously strategy-proof implementation of top trading cycles. *International Economic Review*, 60(3):1249–1261, 2019. 1, 6

# A Additional Preliminaries

**Many-to-one matching markets.** When we study many-to-one matching rules (particularly in Section 5), each $h \in \mathcal{I}$ is equipped with a fixed capacity $q_h \geq 1$, and we require that all matchings $\mu \in \mathcal{M}$ must satisfy $|\mu(h)| \leq q_h$ for each $h \in \mathcal{I}$. To define each of the mechanisms we consider (TTC, APDA, IPDA, and SD) in many-to-one markets, one can use the following standard trick: For each institution $h_i \in \mathcal{I}$ with capacity $q_h$, define $q_h$ distinct institutions $h_i^1, h_i^2, \ldots, h_i^{q_h}$, each with capacity 1 and a priority list identical to $h_i$'s list in $Q$, and replace each $h_i \in \mathcal{I}$ on each preference list of each applicant $d \in \mathcal{A}$ with $h_i^1 \succ h_i^2 \succ \ldots \succ h_i^{q_h}$. Then, each mechanism is defined as the outcome in this corresponding one-to-one market. In economics terms, this means we study responsive preferences.

We now give some well-known properties of TTC and DA which our results rely on.

**Properties of top trading cycles.**

**Lemma A.1** (Follows from SS74; RP77)**.** *The matching output by the* TTC *algorithm in Definition 2.1 is independent of the order in which trading cycles are chosen and matched.*

**Lemma A.2** ([SS74, Rot82b])**.** TTC *is strategyproof and nonbossy.*

**Properties of stable matching mechanisms.**

**Lemma A.3** (GS62)**.** *The output of* APDA *(or* IPDA*) is a stable matching.*

**Corollary A.4** (DF81)**.** *The matching output by the* APDA *algorithm in Definition 2.2 is independent of the order in which applicants are selected to propose.*

**Corollary A.5** (GS62, MW71)**.** *In the matching returned by* APDA*, every applicant is matched to her favorite stable partner. Moreover, every $h \in \mathcal{I}$ is paired to their worst stable match in $\mathcal{I}$. The same holds in reverse for* IPDA*.*

We also need the following classical characterization relating the set of matched agents in each stable outcome.

**Theorem A.6** (Lone Wolf / Rural Hospitals Theorem, Rot86)**.** *The set of unmatched agents is the same in every stable matching. Moreover, in a many-to-one stable matching market, if there is some stable matching where an institution receives fewer matches than their capacity, then that institution receives precisely the same set of matches in every stable outcome.*

For DA and all stable matching mechanisms, we make the following observation about the menu:

**Lemma A.7.** *If $f$ and $g$ any two stable matching mechanism (with respect to priorities $Q$), we have $\mathsf{Menu}_d^f(P_{-d}) = \mathsf{Menu}_d^g(P_{-d})$ for all $d, P_{-d}$.*

*Proof.* Consider any $h \in \mathsf{Menu}_d^f(P_{-d})$, and let $\mu = f(P_d, P_{-d})$ for any $P_d$ such that $\mu(d) = h$. Then $\mu$ will also be stable under preferences $P'$ which are the same as $P = (P_d, P_{-d})$, except that $d$ ranks only $\{h\}$, and thus by Theorem A.6, we must have $\mu'(d) = h = \mu''(d)$, where $\mu' = f(P')$ and $\mu'' = g(P')$. Thus, $h \in \mathsf{Menu}_d^g(P_{-d})$, and by symmetry $\mathsf{Menu}_d^f(P_{-d}) = \mathsf{Menu}_d^g(P_{-d})$. □

By this lemma, when we give bounds related to the complexity of the menu in stable matching mechanisms, the distinction between APDA and IPDA is not important.

We have the following strategyproofness result for APDA:

**Theorem A.8** (Rot82a, DF81)**.** APDA *is strategyproof (for the applicants).*

Note that IPDA is not strategyproofness (and thus, while we sometimes discuss the menu in IPDA, an applicant is not matched in IPDA to their top choice from their menu). However, APDA is bossy, and IPDA is not:

**Proposition A.9.** IPDA *is nonbossy.*

*Proof.* Consider any $Q, P$ where $\mu = \mathsf{IPDA}_Q(P)$, and consider any $\mu(d) = h$. Let $P'$ be identical to $P$, except $d$ truncates her preference list after $h$ (i.e. all institutions strictly below $h$ are marked unacceptable). By Corollary A.5, no stable partner of $d$ was below $h$ on $d$'s list, so the set of stable matchings is identical under $P$ and $P'$ and in particular $\mu = \mathsf{IPDA}_Q(P')$. Now let $P''$ be identical to $P'$, except $d$ truncates her preference list *above* $h$ (i.e., now $d$'s list consists only of $\{h\}$). Since $d$ never receives a proposal from any institution above $h$ during the run of $\mathsf{IPDA}_Q(P')$, we know that this run of DA is identical to $\mathsf{IPDA}_Q(P'') = \mu$. But, $P''$ is identical for any initial value of $P_d$ such that $\mu(d) = h$, so $\mu$ must be identical for any such $P_d$, which finishes the proof. □

# B Proofs Omitted from Section 5.3

## B.1 Complexity of Verifying Counting

First, we formally define the more standard notion of a verification protocols in the context of non-matching problems.

**Definition B.1.** A (public-outcome) *verification protocol* of a function $f : \mathcal{T}_1 \times \ldots \times \mathcal{T}_n \to A$ is defined by a set $\mathcal{C}$, a function $\mathsf{outcome} : \mathcal{C} \to A$, and a profile of functions $(\mathsf{check}_d)_{d\in\{1,\ldots,n\}}$, where $\mathsf{check}_d : \mathcal{T}_d \times \mathcal{C} \to \{0, 1\}$ such that:

1. (Completeness) For every $t = (t_d)_{d\in\{1,\ldots,n\}} \in \mathcal{T}$, there exists a $C \in \mathcal{C}$ such that $f(t) = \mathsf{outcome}(C)$, and $\mathsf{check}_d(t_d, C) = 1$ for each $d \in \{1, \ldots, n\}$.
2. (Soundness) For every $t = (t_d)_{d\in\{1,\ldots,n\}} \in \mathcal{T}$ and $A \neq f(t)$, if $C \in \mathcal{C}$ is such that $A = \mathsf{outcome}(C)$ over $R$, then $\mathsf{check}_d(t_d, C) = 0$ for at least one $d \in \{1, \ldots, n\}$.

The *cost* of a verification protocol is $\log_2 |\mathcal{C}|$. The *(public-outcome) verification complexity* of a function $f$ is the minimum of the cost of all (public-outcome) verification protocols for $f$.

**Definition B.2.** The *bit-counting* function has, for some number of players $n$, input sets $\mathcal{T}_i = \{0, 1\}$ for each $i = 1, \ldots, n$, outcome set $A = \{0, 1, \ldots, n\}$, and outcome function $f(b_1, \ldots, b_n) = \sum_{i=1}^n b_i$.

In the proof of Proposition 5.8, we showed than any verification protocol for TTC or for any stable matching mechanism could be used to define a (public-outcome) verification protocol for the bit-counting problem with no increase in verification cost. We now show that the verification cost of the bit counting problem is nearly as high as possible:

**Proposition B.3.** *The verification complexity of bit-counting with $n$ players is $\Omega(n)$.*

*Proof.* For simplicity, suppose that $n$ is a multiple of 2. Let $\mathcal{F} \subseteq \{0, 1\}^n$ denote the set of inputs $\vec{b} = (b_1, \ldots, b_n)$ such that $\sum_{i=1}^n b_i = n/2$. We will show that for any verification protocol for bit-counting, we have $|\mathcal{C}| \geq |\mathcal{F}|$. To prove this, suppose for contradiction that $|\mathcal{C}| < |\mathcal{F}|$. Observe that two distinct $\vec{b}, \vec{b}' \in \mathcal{F}$ must then correspond to the same certificate $C \in \mathcal{C}$. Now, let $j \in \{1, \ldots, n\}$ be any index such that $b_j \neq b'_j$, and define $\vec{b}''$ such that $b''_j = b'_j$ and $b''_i = b_i$ for each $i \neq j$. Then,

we know $\mathsf{check}_i(b_i'', C) = 1$ for each $i \in \{1, \ldots, n\}$, since each value of $b_i''$ is equal to one of either $b_i$ or $b_i'$. But we also know that $\mathsf{outcome}(C) = \sum_{i=1}^{n} b_i \neq \sum_{i=1}^{n} b_i''$, so the verification protocol must violate soundness. This is a contradiction, showing that we must have $|\mathcal{C}| \geq |\mathcal{F}|$.

Thus, the verification complexity of bit counting is at least $\log |\mathcal{F}| = \log \binom{n}{n/2} = \Omega(n)$. □

## B.2 Deterministic Blackboard Protocol for TTC

The following simple protocol mentioned in Section 5.3 may be folklore:

**Theorem B.4.** *There is a deterministic, blackboard communication protocol for* TTC *(even in a model without the priorities as common knowledge) with communication cost* $\widetilde{O}(|\mathcal{A}|)$.

*Proof.* The protocol proceeds in steps, with one applicant or institution acting per step. First, the protocol selects an arbitrary institution $h_1$, who announces and points to their top choice of applicant $d_1$. Then, $d_1$ announces and points to their top institution $h_2$, who announces and points. This continues, with the protocol keeping track of some tentative chain $h_1 \to d_1 \to h_2 \to d_2 \to \ldots \to h_k \to d_k$, until some cycle is found, say $d_k \to h_j$ for some $j \in \{1, 2, \ldots, k\}$. Every participant in the cycle is permanently matched, and the applicants are removed from the market (and the capacity of the institutions are each reduced by 1). Then, if $j > 1$, this process resumes, starting from $d_{j-1}$ pointing to their top ranked institution with remaining capacity.

If each applicant only pointed once during this protocol, our desired communication bound would follow immediately. Actually, applicants may point more than once, however, an applicant will point a second time *at most once for each cycle which is matched*. Since each cycle permanently matches at least one applicant, there are at most $|\mathcal{A}|$ matched cycles overall, and thus there are at most $O(|\mathcal{A}|)$ occurrences where an applicant is called to point, and thus the protocol uses $\widetilde{O}(|\mathcal{A}|)$ bits of communication overall. □

## B.3 Verification Protocol for APDA and IPDA

We now prove the following result from Section 5.3:

**Theorem 5.11.** *The verification complexity of both* APDA *and* IPDA *are* $\widetilde{\Theta}(|\mathcal{A}|)$.

The remainder of this section is dedicated to the proof. First, note that the lower bound is given in Proposition 5.8.

For the upper bound, it suffices to consider balanced markets with $n = |\mathcal{A}| = |\mathcal{I}|$, and prove that the verification complexity in this market is $\widetilde{O}(n)$. To see why, observe that the outcome of APDA (or IPDA) in a many-to-one market corresponds to the outcome in the one-to-one market where every incitation $h \in \mathcal{I}$ in the original market is replaced by institutions $h^1, h^2, \ldots, h^{q_h}$ (each with priority list identical to $h$'s, and where $h$ is replaced on each applicant's preference list by $h^1 \succ h^2 \succ \ldots \succ h^{q_h}$). Thus, a verification protocol with cost $\widetilde{O}(|\mathcal{A}|)$ for many-to-one markets can be defined by running a verification protocol on the corresponding one-to-one market, and for the rest of this section we focus on one-to-one markets. Thus, for the rest of this section, we assume that $n = |\mathcal{A}| = |\mathcal{I}|$.

We now make the main definitions needed for our verification protocols, the "improvement graphs". This definition is closely related to notions from [GI89], the standard reference work on the lattice properties of the set of stable matchings, and we will crucially use their results in our proof of correctness below.

**Definition B.5.** Consider any one-to-one matching market with priorities $Q$ and preferences $P$, and consider any matching $\mu$.

First, define a directed graph $\mathsf{ImprGr}^{\mathcal{I}} = \mathsf{ImprGr}^{\mathcal{I}}(Q, P, \mu)$. The vertices of $\mathsf{ImprGr}^{\mathcal{I}}$ are all applicants $d \in \mathcal{A}$. The edges of $\mathsf{ImprGr}^{\mathcal{I}}$ are all those pairs $(d, d')$ such that there exists an $h \in \mathcal{I}$ such that $h \neq \emptyset$ is the highest institution on $d$'s preference list such that $d \succ_h \mu(h)$, and moreover $\mu(h) = d'$. For such a $d, d', h$, we denote this edge by $d \xrightarrow{h} d'$.

Second, define a directed graph $\mathsf{ImprGr}^{\mathcal{A}} = \mathsf{ImprGr}^{\mathcal{A}}(Q, P, \mu)$, with precisely the same definition as $\mathsf{ImprGr}^{\mathcal{I}}$, except interchanging the roles of applicants and institutions.

Observe that the vertices of each of $\mathsf{ImprGr}^{\mathcal{I}}$ and $\mathsf{ImprGr}^{\mathcal{A}}$ have out-degree at most 1, and thus these graphs take only $\widetilde{O}(n)$ bits to describe.

Our key lemma will show next that a matching equals $\mathsf{IPDA}$ if and only if $\mathsf{ImprGr}^{\mathcal{I}}$ is acyclic. To begin to gain intuition for this, observe that if $\mu$ is stable and $d \xrightarrow{h} d'$ is an edge in $\mathsf{ImprGr}^{\mathcal{I}}$, then $h$ must be below $\mu(d)$ on $d$'s preference list. Informally, the lemma will follow because edges in $\mathsf{ImprGr}^{\mathcal{I}}$ represent possible ways that matches could be exchanged in a way that makes the matching better for the applicants and worse for the institutions, and cycles in $\mathsf{ImprGr}^{\mathcal{I}}$ characterize such changes to the matching that could be accomplished without violating stability.

**Lemma B.6.** *Fix any set of preferences $P$ and priorities $Q$. For any matching $\mu$, we have $\mu = \mathsf{IPDA}_Q(P)$ if and only if $\mu$ is stable with respect to $P$ and $Q$, and the graph $\mathsf{ImprGr}^{\mathcal{I}}(Q, P, \mu)$ is acyclic. Dually, $\mu = \mathsf{APDA}_Q(P)$ if and only if $\mu$ is stable and $\mathsf{ImprGr}^{\mathcal{A}}(Q, P, \mu)$ is acyclic.*

*Proof.* We prove the first result, namely, that $\mathsf{IPDA}$ is characterized by $\mathsf{ImprGr}^{\mathcal{I}}$. The proof relies heavily on results from [GI89, Section 2.5.1]. First, observe that in the language of [GI89], the edges $d \xrightarrow{h} d'$ of $\mathsf{ImprGr}^{\mathcal{I}}$ are exactly those $d, h, d'$ such that $h = s_\mu(d)$ and $d' = next_\mu(d)$.[23]

First, we show that if $\mu$ is stable and $\mathsf{ImprGr}^{\mathcal{I}}$ has a cycle, then $\mu \neq \mathsf{IPDA} = \mathsf{IPDA}_Q(P)$. Suppose $d_1 \xrightarrow{h_1} d_2 \xrightarrow{h_2} d_3 \xrightarrow{h_3} \dots \xrightarrow{h_{k-1}} d_k \xrightarrow{h_k} d_1$ is some cycle in $\mathsf{ImprGr}^{\mathcal{A}}$. Then, according to the definitions given in [GI89, Page 88], we know $\rho = [(d_1, h_k), (d_2, h_1), \dots, (d_k, h_{k-1})]$ is a rotation exposed in $\mu$. Define a matching $\mu'$ such that $\mu'(d_i) = h_i$ for each $i = 1, \dots, k$ (with indices mode $k$), and $\mu'(d) = \mu(d)$ for all $d$ not contained in the above cycle (in the language of [GI89], we have $\mu' = \mu/\rho$, the elimination of rotation $\rho$ from $\mu$). [GI89, Lemma 2.5.2] shows that $\mu' \neq \mu$ is also stable and is preferred by all institutions to $\mu$. Thus, by the fact that $\mathsf{IPDA}$ is the institution-optimal stable outcome, we know $\mu \neq \mathsf{IPDA}$.

Second, we show that if $\mu$ is stable and $\mu \neq \mathsf{IPDA}_Q(P)$, then $\mathsf{ImprGr}^{\mathcal{I}}$ must have a cycle. By [GI89, Lemma 2.5.3], there must be some rotation $\rho = [(d_1, h_k), (d_2, h_1), \dots, (d_k, h_{k-1})]$ which is exposed in $\mu$. By the definition of $\mathsf{ImprGr}^{\mathcal{I}}$ and the definition of a rotation from [GI89], this rotation must correspond to a cycle $d_1 \xrightarrow{h_1} d_2 \xrightarrow{h_2} d_3 \xrightarrow{h_3} \dots \xrightarrow{h_{k-1}} d_k \xrightarrow{h_k} d_1$ in $\mathsf{ImprGr}^{\mathcal{I}}$.[24] So $\mathsf{ImprGr}^{\mathcal{I}}$ is

[23] We remark that [GI89, Section 2.5.1, Page 87] mentions the following: "Note that $s_\mu(d)$ might not exist. For example, if $\mu$ is the [institution]-optimal [stable] matching, then $s_\mu(d)$ does not exist for any [applicant] $d$." The second sentence of this quote would seem to eliminate the need for our verification protocol to communicate any graph whatsoever, since if $\mu = \mathsf{IPDA}$, then this graph should be empty. However, the claim made in that second sentence is false. For example, if $h_1 : d_1 \succ d_2$, $h_2 : d_2 \succ d_1$, $h_a : d_1 \succ d_2 \succ d_3$, and $d_1 : h_2 \succ h_a \succ h_1$, $d_2 : h_1 \succ h_a \succ h_2$, $d_3 : h_a$, then $\mu = \mathsf{IPDA} = \{(d_1, h_2), (d_2, h_1), (d_3, h_a)\}$, and $s_\mu(d_1) = s_\mu(d_2) = h_a \neq \emptyset$. (Note that $s_\mu(d_3) = \emptyset$, and thus our graph $\mathsf{ImprGr}^{\mathcal{I}}$ has edges $d_1 \to d_3 \leftarrow d_2$, and is indeed acyclic.) In contrast to this remark which [GI89] makes in passing, all of the lemmas and theorems of [GI89, Section 2.5.1] which we rely on for are proof are correct.

[24] In fact, our $\mathsf{ImprGr}^{\mathcal{I}}$ is defined in a way very close to the graph $H(\mu)$ which [GI89] consider in the proof of their Lemma 2.5.3; our graph $\mathsf{ImprGr}^{\mathcal{I}}$ simply considers the entire set of applicants, instead of just those who have a different partner in $\mu$ and $\mathsf{IPDA} \neq \mu$, and our graph generalizes theirs to arbitrary matchings $\mu$, possibly including $\mathsf{IPDA}$ (which fixes the minor error mentioned in footnote 23).

cyclic.

Since $\mathsf{ImprGr}^{\mathcal{A}}$ is defined precisely by interchanging the roles of applicants and institutions, we know that $\mathsf{ImprGr}^{\mathcal{A}}$ characterizes $\mathsf{APDA}$ in the same way. □

Now, we can define verification protocol $V^{\mathsf{IPDA}}$ (resp., $V^{\mathsf{APDA}}$) for $\mathsf{IPDA}$ (resp., $\mathsf{APDA}$). When priorities are $Q$ (recall that the priorities $Q$ are commonly known to each applicant before the mechanism begins[25]) and preferences are $P$, the certificate $C$ records the matching $\mu = \mathsf{IPDA}_Q(P)$ (resp., $= \mathsf{APDA}_Q(P)$) and graph $G = \mathsf{ImprGr}^{\mathcal{I}}$ (resp., $= \mathsf{ImprGr}^{\mathcal{A}}$). By Lemma B.6, it will suffice for the applicants to collectively verify that the matching $\mu$ is stable, and that the graph $G$ is correctly constructed. As already noted in Section 5.3, it is not hard to verify that $\mu$ is stable once $\mu$ is known: each applicant $d$ can tell using $\mu$ and $Q$ which institutions $h$ are such that $d \succ_h \mu(h)$ according to $Q$, and each applicant can simply check that there are no such $h$ where $h \succ_d \mu(d)$. Thus, to complete the proof, it suffices to show the following lemma:

**Lemma B.7.** *If priorities $Q$ are fixed ahead of time, there exists a predicate $\mathsf{check}_d(P_d, \mu, G) \in \{0,1\}$ for each $d \in \mathcal{A}$, matchings $\mu$, and for all graphs $G$, we have:*

$$\bigwedge_{d \in \mathcal{A}} \mathsf{check}_d(P_d, \mu, G) = 1 \quad \iff \quad G = \mathsf{ImprGr}^{\mathcal{I}}(Q, P, \mu).$$

*Moreover, the same claim holds (with a different predicate $\mathsf{check}_d(\cdot)$) replacing $\mathsf{ImprGr}^{\mathcal{I}}$ with $\mathsf{ImprGr}^{\mathcal{A}}$.*

*Proof.* We first prove the claim for $\mathsf{ImprGr}^{\mathcal{I}}$. This case is not too hard to see: each $d$ just needs to verify that the edge outgoing from their node in $G$ exactly corresponds to $d \xrightarrow{h} d'$, where $d' = \mu(h)$ and $h$ is $d$'s highest-ranked institution such that $d \succ_h \mu(h)$ (if it exists). Since $d$ knows the priorities, $d$ also knows the set of $h$ such that $d \succ_h \mu(h)$, and can thus perform this verification.

We now prove the claim for $\mathsf{ImprGr}^{\mathcal{A}}$, which requires a somewhat more delicate predicate $\mathsf{check}_d(\mu, G)$. For clarity, note that the full definition of $\mathsf{ImprGr}^{\mathcal{A}}$ is as follows: There is a vertex for each $h \in \mathcal{I}$, and an edge $h \xrightarrow{d} h'$ whenever $h, h', d$ are such that $\mu(d) = h'$, and $d \neq \emptyset$ is the highest-ranked applicant on $h$'s preference list such that $h \succ_d \mu(d)$. We claim that it suffices for each applicant $d$'s predicate $\mathsf{check}_d(\mu, G)$ to verify the following things:

- For each edge $h_x \xrightarrow{d} h_y$, we have $h_x \succ_d h_y = \mu(d)$.
- For each edge $h_x \xrightarrow{d_y} h_y$ with $d_y \neq d$: if we let $d_x = \mu(h_x)$, then whenever $d_x \succ_{h_x} d \succ_{h_x} d_y$ according to the priorities $Q$, we must have $\mu(d) \succ_d h_x$.
- For each $h$ with out-degree zero in $G$, we have $\mu(d) \succ_d h$.

The first conditions is necessary for each $d$ by the definition of $\mathsf{ImprGr}^{\mathcal{A}}$. If the first and second condition are both true for every $d \in \mathcal{A}$, then we know that every edge in $G$ is correctly constructed according to $\mathsf{ImprGr}^{\mathcal{A}}$, because if these conditions hold then each $d_y$ in some edge $h_x \xrightarrow{d_y} h_y$ must be the highest-ranked applicant below $d_x = \mu(h_x)$ such that $h_x \succ_{d_y} \mu(d_y)$. Finally, the third condition for every $d \in \mathcal{A}$ grantees that an institution has out-degree zero in $G$ if and only if they should have out-degree zero according to $\mathsf{ImprGr}^{\mathcal{A}}$. This shows that $G = \mathsf{ImprGr}^{\mathcal{A}}$ if and only if each of the above three predicates are true for each $d \in \mathcal{A}$, as desired. Since the above conditions can be verified for each $d$ using only knowledge of the priorities, $\mu$, $G$, and $d$'s own preferences, this proves the lemma. □

[25] Several steps of this protocol (both checking stability and checking the correctness of the graph $\mathsf{ImprGr}^{\mathcal{I}}$ or $\mathsf{ImprGr}^{\mathcal{A}}$) would require $\Omega(n^2)$ bits if priorities are not common knowledge, c.f. [GNOR19].

This finishes the proof of Theorem 5.11: the verification protocols $V^{\mathsf{IPDA}}$ and $V^{\mathsf{APDA}}$ communicate a certificate containing the matching $\mu$ and corresponding graph $G$, which requires only $\widetilde{O}(n) = \widetilde{O}(|\mathcal{A}|)$ bits, and applicants check that $\mu$ is stable and that the graph is correctly constructed as above.

**Remark B.8.** While our model (and our lower bounds) assume that the entire priority lists of each institution are common knowledge, all of the protocols we mention or construct Section 5 can actually be implemented with a more mild assumption on the knowledge of the priorities. Assume that there are scores $e_h^d \in [0,1]$ for each $d \in \mathcal{A}$ and $h \in \mathcal{I}$, and let $d \succ_h d'$ if and only if $e_h^d > e_h^{d'}$. Assume that each applicant $d$ start off knowing $\{e_h^d\}_{h \in \mathcal{I}}$, but any other information must be communicated through the certificate $C$. First, observe that the protocols adapted from [AL16] and [LL21] in Section 5.2 work exactly as written.

Now, consider the verification protocol for TTC from Observation 5.10. This verification protocol can still post a transcript of the deterministic protocol for TTC from Theorem B.4, since that deterministic protocol works even when the priorities must be communicated. However, note that the institutions are not agents in this protocol which can help us verify the matching. Thus, every time that an institutions $h$ points to an applicant $d$ during the transcript, the protocol should also announce the priority score of applicant $d$ at institution $h$; and each applicant who is not yet matched in the transcript should check that they do not have higher priority at $h$. This will suffice to verify the transcript of the deterministic protocol.

For DA, an analogous trick works. When the protocol to verify IPDA posts certificate $\mu, \mathsf{ImprGr}^{\mathcal{I}}$, it should also post the priorities scores of $\mu(h)$ at $h$ for each $h \in \mathcal{I}$, and applicants will still be able to verify each edge $d \xrightarrow{h} d'$ in $\mathsf{ImprGr}^{\mathcal{I}}$, as in Lemma B.7. When the protocol to verify APDA posts certificate $\mu, \mathsf{ImprGr}^{\mathcal{A}}$, it should also post, for every edge $h \xrightarrow{d} h'$, the priorities scores of $d$ at both $h$ and $h'$. Then, again each applicant knows what they needs in order to perform the verification as in Lemma B.7.

**Remark B.9.** While Theorem 5.11 holds for both APDA and IPDA, it cannot be extended to any matching mechanism. To see why, consider a market with applicants $d_1^0, \dots, d_n^0, d_1^1, \dots, d_n^1$ and institutions $h_1^0, \dots, h_n^0, h_1^1, \dots, h_n^1$, where each applicant and institution will have full-length preference lists. Suppose each institution $h_i^j$ for $i \in \{1, \dots, n\}$ and $j \in \{0,1\}$ ranks $d_i^j$ first and ranks $d_i^{1-j}$ last, with any fixed order between them. Now consider the set of preferences where each applicant $d_i^j$ for $i \in \{1, \dots, n\}$ and $j \in \{0,1\}$ ranks $h_i^{1-j}$ first and ranks $d_i^j$ last, where any possible ordering over the other institutions appears between first and last. Observe that, for any profile of preferences in this class, the matchings $\mu_0 = \{(d_i^j, h_i^j)\}_{i \in \{1,\dots,n\}, j \in \{0,1\}}$ and $\mu_1 = \{(d_i^j, h_i^{1-j})\}_{i \in \{1,\dots,n\}, j \in \{0,1\}}$ are both stable. Thus, consider a stable matching mechanism $f$ which (on inputs in this class) outputs either $\mu_0$ or $\mu_1$, depending on some arbitrary function $g : \mathcal{T} \to \{0,1\}$ of the preference lists of the $2n$ applicants, and suppose that the function $g$ has verification complexity $\Omega(n^2)$ (such a function can be constructed by standard techniques using a counting argument). Then, the verification complexity of $f$ will be at least the verification complexity of $g$, i.e., $\Omega(n^2)$.

# C Relation to other frameworks and results

## C.1 Verification of DA

This section discusses how the results of [AL16] and [Seg07] relate to the results of Section 5.2 through Section 6.1 for DA.

[Seg07] is concerned with the verification of social-choice functions and social-choice correspondences. Social-choice functions are the natural generalization of matching rules to scenarios other than matching, such as voting or auctions. Social-choice correspondences are generalizations of social-choice functions, where there can be multiple valid outcomes that correspond to the same inputs of the players. For example, "the APDA outcome" is a social-choice function, while "a stable outcome" is a social-choice correspondence. [Seg07] shows that, for a large class of social-choice correspondences including the stable matching correspondence, the verification problem (equivalent to our verification complexity in Definition B.1, which differs from Definition 5.7 only in that every agent should know the outcome of the protocol) reduces to the task of communicating "minimally informative prices."

When [AL16] relate their work—in particular the cutoff representation of the matching of DA, which we discussed in the proof of Observation 5.4—to the work of [Seg07], they mention that their cutoffs coincide with [Seg07]'s minimally informative prices. Since [Seg07] shows that these prices verify that a matching is stable, this may seem to imply that the $\widetilde{\Theta}(|\mathcal{I}|)$-bit vector of cutoffs should suffice to verify that a matching is stable, contradicting our Proposition 5.8, which says that $\Omega(|\mathcal{A}|)$ bits are needed (even to verify that a matching is stable). However, there is no actual contradiction: the model and characterization in [Seg07] implicitly assume that each agent in the verification protocol knows the complete matching, ruling out the $\widetilde{\Theta}(|\mathcal{I}|)$-bit representation protocol implicitly discussed in [AL16] and formalized by our Observation 5.4. Indeed, as we discussed in Section 5.3, simply writing down the matching suffices to verify in our model that a matching is stable using $\widetilde{\Theta}(|\mathcal{A}|)$ bits (though verifying that the matching is the outcome of APDA or IPDA, as we do in Theorem 5.11, is more involved).

There is also a strong conceptual relation between our results in Section 5.3 for DA and the results of [Seg07, Section 7.5] and [GNOR19]. Namely, all of these results bound the complexity of verifying stable matchings. However, there is a large technical difference between our model and prior work: we treat the priorities as fixed and known by all applicants. This difference turns out to have dramatic implications: while both [Seg07] and [GNOR19] achieve $\Omega(n^2)$ lower bounds (in different models) for the problem of verifying a stable matching where $n = |\mathcal{A}| = |\mathcal{I}|$, we get an upper bound of $\widetilde{O}(n)$ (Theorem 5.11).

## C.2 Cutoff structure of TTC

This section discusses how the results of [LL21] relate to our results for TTC, especially in Section 5.2.

As mentioned in the proof of Theorem 5.5, [LL21] prove that the outcome of TTC can be described in terms of $|\mathcal{I}|^2$ "cutoffs," where in the language of our paper, a cutoff is simply an index on an institution's priority list. Based on examples and the intuition that their description provides, they state that "the assignment cannot [in general] be described by fewer than $\frac{1}{2}n^2$ cutoffs." However, they do not formalize what is precisely meant by this claim.

[LL21] briefly discuss one formal result in the direction of an impossibility result. We attempt to capture the essence of their approach as follows:

**Proposition C.1** (Adapted from [LL21, Footnote 8])**.** *In* TTC*, for any $n$, there exists a market with fixed priorities $Q$, institution $h$, and applicants $U = \{d_1, \ldots, d_n\} \subseteq \mathcal{A}$, where $n = \Theta(|\mathcal{I}|) = \Theta(|\mathcal{A}|)$, with the following property: Let $P^*_U$ denote preferences of applicants in $U$ such that each $d \in U$ ranks only $\{h\}$. Then, for any $S \subseteq U$, there exist priorities $P_{-U}$ of applicants outside $U$ such that if $\mu = \mathsf{TTC}_Q(P^*_U, P_{-U})$, then for each $d \in S$ we have $\mu(d) = h$, and for each $d \in U \setminus S$, we have $\mu(d) \neq h$.*

*On the other hand, in* APDA*, suppose there is any $Q, h$, and set $U = \{d_1, \ldots, d_k\} \subseteq \mathcal{A}$ with the following property: For any $S \subseteq U$, there exist priorities $P_{-U}$ such that if $\mu = \mathsf{APDA}_Q(P_U^*, P_{-U})$, we have $\mu(d) = h$ for each $d \in S$ and $\mu(d) \neq h$ for each $d \notin S$. Then we have $k = |U| \leq 1$.*

*Proof.* For TTC, consider a market with applicants $d_i, d_i'$ for $i = 1, \ldots, n$ and institutions $h_i$ for $i = 0, 1, \ldots, n$. Now, let institution $h_0$ have capacity $n$ and priority list $d_1' \succ \ldots \succ d_n' \succ d_1 \succ d_n$, and for each $i = 1, \ldots, n$ let $h_i$ have capacity 1 and have priority list $d_i \succ d_i' \succ L$, where $L$ is arbitrary. Let each $d_i$ submit list $h_0 \succ h_i$, and consider the $2^n$ preference profiles where each $d_i'$ might submit list $h_0 \succ h_i$, or might submit list $h_i \succ h_0$. Then, by the way TTC matches applicants, $d_i$ will match to $h_0$ if and only if $d_i'$ submits list $h_i \succ h_0$, as desired.

For APDA, consider any fixed $Q$, $h$, and $\{d_1, d_2\} \subseteq U$ where (without loss of generality) $d_1 \succ_h^{Q_h} d_2$. For any $P$ such that $d_1$ and $d_2$ both rank only $h$, if $\mu$ is stable and $\mu(d_1) = h$, then we must have $\mu(d_2) = h$, by stability. This concludes the proof (and in fact shows that the claim holds for any stable matching mechanism, not just APDA). □

While Proposition C.1 provides an interesting formal sense in which TTC (but not DA) is complex, this approach only considers admission to one institution $h$ at a time, and correspondingly to our understanding might only conceivably show an $\Omega(n)$ lower bound and not the desired $\Omega(n^2)$ lower bound that we formalize in Section 5.2.

**Remark C.2.** In contrast to Proposition C.1, [LL21, Footnote 8] from which it is adapted is phrased in terms of separately defined definitions of budget sets for each of TTC and APDA. For TTC, the budget sets are defined in [LL21]; see [LL21] for the precise definition (whose details are not required to follow this discussion). While neither [LL21] nor the most closely related work [AL16] seem to explicitly define the budget set in APDA, we confirmed with the authors of [LL21] (private communication, October 2022) that they intended the budget set of $d$ in APDA to be the sets which we denote as $\mathsf{StabB}_d(\mathsf{APDA}_Q(P))$ (see Definition 5.2). Note that neither of these sets equals the menu in the corresponding mechanism. This was observed for TTC in [LL21, Footnote 10] and for APDA in the beginning of our Section 6.1 (and in [GHT22]). Our Theorem 6.3 directly implies that the result in [LL21, Footnote 8] for DA would no longer hold if the budget set $\mathsf{StabB}_d$ were replaced with the menu $\mathsf{Menu}_d^{\mathsf{APDA}}$. We rephrased the insights of [LL21, Footnote 8] to produce Proposition C.1, which replaces the notion of an institution $h$ being in a budget set of applicant $d$ with the question of whether $d$ matches to $h$ when $d$ only ranks $h$. We did this in order to compare the mechanisms DA and TTC through a single complexity lens which is perhaps more consistent with the rest of the current paper, rather than using budget sets, which despite their very appealing properties, to our knowledge must be defined separately for the two mechanisms.

We also remark that [LL21] occasionally use the word "verification," and speak of agents "verifying their match." However (unlike [Seg07]), to our best understanding they seem to use this term in an informal sense of checking that an assignment is accurate, rather than as any sort of verification according to formal computer-science notions.

# D Serial Dictatorship

In this section, we detail all of our complexity measures for the Serial Dictatorship mechanism. Recall that SD is defined with respect to a fixed priority order $\succ$ over applicants, and the matching is determined by picking applicants in descending order of the priority and permanently matching them to their top-ranked remaining institution. Later in the section, we also discuss $\mathsf{SD}^{\mathsf{rot}}$, as defined in Definition 3.5.

**Theorem D.1.** *The type-to-matching complexity of* $\mathsf{SD}$ *is* $\widetilde{\Theta}(n)$.

*Proof.* Suppose $\mathcal{A} = \{1, 2, \ldots, n\}$ and $\succ$ ranks applicants in order $1 \succ 2 \succ \ldots n$. Our goal is to bound $\max_{d_*} \log_2 |\{\mathsf{SD}(\cdot, P_{-d_*}) \mid P_{-d_*} \in \mathcal{T}_{-d_*}\}|$; observe that it is without loss of generality to take $d_* = 1$. Observe also that an $\widetilde{\Omega}(n)$ lower bound follows from the need to write down the matching that results if applicant 1 selects no items.

Now, fix preferences $P_{-1}$ of applicants other than $d_* = 1$. Our proof will proceed by finding another profile of preferences $P_{\mathsf{small}}$, such that (1) each applicant's preference list in $P_{\mathsf{small}}$ has length at most two, and (2) for all $P_1 \in \mathcal{T}_1$, we have $\mathsf{SD}(P_1, P_{\mathsf{small}}) = \mathsf{SD}(P_1, P_{-1})$.

To begin, we define a set of "filtered" preferences of applicants other than applicant 1. Define $P^1_{\mathsf{filt}} = P_{-1} \in \mathcal{T}_{-1}$. Now, for each $i = 2, 3, \ldots, n-1$ in order, define $P^i_{\mathsf{filt}}$ by modifying $P^{i-1}_{\mathsf{filt}}$ as follows: if $h \in \mathcal{I}$ is the top-ranked institution on $i$'s list in $P^{i-1}_{\mathsf{filt}}$, remove $h$ from the preference list of each applicant $d$ with $d > i$. Define $P_{\mathsf{filt}}$ as $P^{n-1}_{\mathsf{filt}}$.

First, we show that $P_{\mathsf{filt}}$ always produces the same matching as $P_{-1}$:

**Lemma D.2.** *For any* $P_1 \in \mathcal{T}_1$*, we have* $\mathsf{SD}(P_1, P_{-1}) = \mathsf{SD}(P_1, P_{\mathsf{filt}})$.

To prove this, consider each step of the above recursive process, where some applicant $i$ ranked $h$ first on $P^{i-1}_{\mathsf{filt}}$, and we constructed $P^i_{\mathsf{filt}}$ by removing $h$ from the list of all $d > i$. Now, observe that for any possible $P_1$, when $d$ picks an institution in $\mathsf{SD}(P_1, P^i_{\mathsf{filt}})$, institution $h$ must already be matched (either to applicant $i$, or possibly an earlier applicant). Thus, removing $h$ from the list of $d$ cannot make a difference under any $P_1$, and for each $P_1 \in \mathcal{T}_1$, we have $\mathsf{SD}(P_1, P_{-1}) = \mathsf{SD}(P_1, P^i_{\mathsf{filt}})$, by induction. This proves Lemma D.2.

Next, we show that only the first two institutions on each preference list in $P'_{-1}$ can matter:

**Lemma D.3.** *For any* $P_1 \in \mathcal{T}_1$*, each applicant will be matched to either their first or second institution in their preference list in* $P_{\mathsf{filt}}$.

To prove this, suppose player 1's top choice according to $P_1$ is institution $h_1 \in \mathcal{I}$. The mechanism will run initially with each applicant $i > 1$ taking their first choice (all of which are distinct in $P_{\mathsf{filt}}$), until some applicant $i_2$ which ranks $h_1$ as their first choice. This $i_2$ will take their second choice $h_2$ (since $h_2$ cannot yet be matched, because each prior applicant took their first choice). But, applying this same argument for applicants $d > i_2$, we see that all such applicants will be matched to their first choice until $i_3 > i_2$ which ranks $h_2$ first. This $i_3$ will get their second choice $h_3$ (which cannot be $h_1$, since $h_1$ was ranked first by $i_2$). This will continue again until some $i_4$ whose first choice is $h_3$ and second choice is $h_4 \notin \{h_1, h_2, h_3\}$; this applicant $i_4$ will match to $h_4$. This argument applies recursively until the mechanism is finished, proving Lemma D.3.

Now, consider preferences $P_{\mathsf{small}}$, which contain only the first two institutions on the list of each applicant in $P_{\mathsf{filt}}$. Then, by both lemmas above, we know that for each $P_1 \in \mathcal{T}_1$, we have $\mathsf{SD}(P_1, P_{-1}) = \mathsf{SD}(P_1, P_{\mathsf{small}})$. Because each list in $P_{\mathsf{small}}$ is of size at most 2, it takes $\widetilde{O}(n)$ bits to write down $P_{\mathsf{small}}$, and thus the type-to-menu complexity of serial dictatorship is $\widetilde{O}(n)$. □

Moreover, the above lemma proves that the matchings $\{\mathsf{SD}(P_1, P_{-1}) \mid P_1 \in \mathcal{T}_1\}$ have a regular structure, which is somewhat analogous to the structure uncovered in Section 4.2 in regards to the type-to-menu complexity of $\mathsf{DA}$. In particular, there is some "default matching" of applicants $i > 1$ (which corresponds to the matching when applicant $d_1$ submits an empty preference list and matches to no institutions), and there is a directed acyclic graph of "displaced matches" which might occur based on the choice of player 1 (where this graph can be constructed with a vertex corresponding to each applicant by considering $P_{\mathsf{small}}$, and having each applicant with preference list $h_i \succ h_j$ point to the applicant who ranks $h_j$ first).

We also remark that this same construction can be adapted to show that the type-to-menu

complexity of SD is small (though this also follows as a corollary of Theorem 4.4).

For the remainder of the complexity measures we consider, getting a tight bound for SD was either mentioned in the main body of this paper or is more or less trivial. Similarly to the representation protocol for DA in Observation 5.4, a representation protocol for SD can specify where in the priority order each institution ceases to have available capacity. The verification complexity of SD is lower bounded by Proposition 5.8, and a matching upper bound can be constructed using the trivial deterministic protocol. The all-menus complexity is trivial to represent using a transcript of the same deterministic protocol. The all-type-to-one-match complexity of SD is proven to be $\Omega(n^2)$ in Section 6.2.

Finally, we study $\mathsf{SD}^{\mathsf{rot}}$. While Lemma 3.6 shows that this mechanism has high type-to-matching complexity, it turns out to have low type-to-menu complexity. (In this way, the complexity measures of $\mathsf{SD}^{\mathsf{rot}}$ are similar to the complexity measures of DA, though interestingly, we only know how to embed $\mathsf{SD}^{\mathsf{rot}}$ into TTC but not into DA.) This gives some formal sense in which our bounds on the type-to-matching complexity of TTC and the type-to-menu complexity of TTC, which are both $\Omega(n^2)$, must hold for different reasons.

**Theorem D.4.** *The type-to-menu complexity of* $\mathsf{SD}^{\mathsf{rot}}$ *is* $\widetilde{O}(n)$.

*Proof.* We bound the number of bits required to represent the function $g(P_{d_*}) = \mathsf{Menu}^{\mathsf{SD}^{\mathsf{rot}}}_{d_\dagger}(P_{d_*}, P_{-\{d_*,d_\dagger\}})$ for some $d_*, d_\dagger$. Recall that in $\mathsf{SD}^{\mathsf{rot}}$, applicant $d_0$ picks some institution $h^{\mathsf{rot}}_j$, and the other applicants are matched among $\{h_1, \ldots, h_n\}$ according to $\mathsf{SD}_{d_j,d_{j+1},\ldots,d_n}(\cdot)$. First, note that if we consider any $d_* \neq d_0$, the question can only be as hard as for SD, and if we consider any $d_\dagger = d_i$ for $i < n$, then the applicants $d_{i'}$ for $i' > i$ cannot possibly effect $d_\dagger$'s menu. Thus, it is without loss of generality to take $d_* = d_0$ and $d_\dagger = d_n$.

Now, the key lemma is the following:

**Lemma D.5.** *Fix a set of preferences $P$, and suppose $h$ is in $d_n$'s menu under the mechanism $\mathsf{SD}_{d_j,d_{j+1},\ldots,d_n}(P_j, P_{j+1}, \ldots)$ for some $j$. Then, $h$ is in $d_n$'s menu under $\mathsf{SD}_{d_{j+1},d_{j+2},\ldots,d_n}(P_{j+1}, P_{j+2}, \ldots)$ as well.*

To prove this lemma, consider the run of $\mathsf{SD}_{d_{j+1},d_{j+2},\ldots,d_n}$ compared to $\mathsf{SD}_{d_j,d_{j+1},\ldots,d_n}$. Similarly to the proof of Lemma D.2, observe that when each applicant $d_k$ for $j < k \leq n$ picks from their menu, they can have only fewer options in $\mathsf{SD}_{d_j,d_{j+1},\ldots,d_n}$ than in $\mathsf{SD}_{d_{j+1},d_{j+2},\ldots,d_n}$. In particular, this applies to $d_n$, proving Lemma D.5.

Thus, to represent the function $g(\cdot)$, we claim that it suffices write down an ordered list $S_1, S_2, \ldots, S_n \subseteq \mathcal{I}$, which is defined as follows: For each $i > 1$, the set $S_i \subseteq \mathcal{I}$ is the subset of institutions that are on $d_n$'s menu in $\mathsf{SD}_{d_{i+1},d_{i+2},\ldots,d_n}$ but not in $\mathsf{SD}_{d_i,d_{i+1},\ldots,d_n}$. For $i = 1$, set $S_1$ is the menu of $d_n$ in $\mathsf{SD}_{d_1,d_2,\ldots,d_n}$. Then, Lemma D.5 implies that for any $P_{d_*}$, if $j_*$ is such that applicant 0 picks $h^{\mathsf{rot}}_{j_*}$, we have $g(P_{d_*}) = \bigcup_{j=1}^{j_*} S_j$. Moreover, Lemma D.5 implies that each institution can appear in at most one set $S_i$, so we can represent this list in $\widetilde{O}(n)$ bits, as claimed. □