Lecture 10

Kidney Exchange and Stable Matching

This lecture is our last on mechanism design, and it covers some of the greatest hits of mechanism design without money. Kidney exchanges, the case study covered in Section 10.1, have been deeply influenced by ideas from mechanism design over the past ten years. These exchanges enable thousands of successful kidney transplants every year. Stable matching and the remarkable deferred acceptance algorithm (Section 10.2) form the basis of modern algorithms for many assignment problems, including medical residents to hospitals and students to elementary schools. This algorithm also enjoys some beautiful mathematical properties and incentive guarantees (Section 10.3).

10.1 Case Study: Kidney Exchange

10.1.1 Background

Many people suffer from kidney failure and need a kidney transplant. In the United States, more than 100,000 people are on the waiting list for such transplants. An old idea, used also for other organs, is deceased donors; when someone dies and is a registered organ donor, their organs can be transplanted into others. One special feature of kidneys is that a healthy person has two of them and can survive just fine with only one of them. This creates the possibility of *living* organ donors, such as a family member of the patient in need.

Unfortunately, having a living kidney donor is not always enough; sometimes a patient-donor pair is *incompatible*, meaning that the donor's kidney is unlikely to function well in the patient. Blood and tissue types are the primary culprits for incompatibilities. For example, a patient with O blood type can only receive a kidney from a donor with the same blood type, and similarly an AB donor can only donate to an AB patient.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

Suppose patient P1 is incompatible with her donor D1 because they have blood types A and B, respectively. Suppose P2 and D2 are in the opposite boat, with blood types B and A, respectively (Figure 10.1). Even though (P1, D1) and (P2, D2) have probably never met, exchanging donors seems like a pretty good idea, with P1 receiving her kidney from D2 and P2 from D1. This is called a *kidney exchange*.

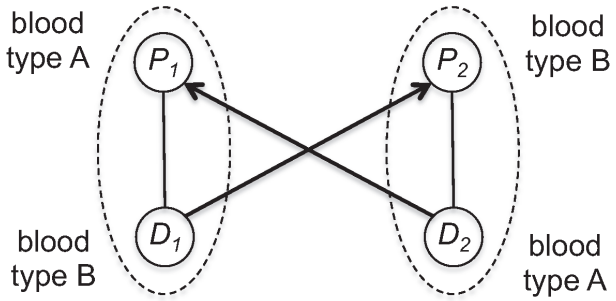

Figure 10.1: A kidney exchange. (P1, D1) and (P2, D2) form incompatible donor pairs. P1 receives her kidney from D2 and P2 from D1.

A few kidney exchanges were done, on an ad hoc basis, around the beginning of this century. These isolated successes made clear the need for a nationwide kidney exchange, where incompatible patient-donor pairs can register and be matched with others. How should such an exchange be designed? The goal is to enable as many matches as possible.

Currently, compensation for organ donation is illegal in the United States, and in every country except for Iran.[1] Kidney exchange is legal, and is naturally modeled as a mechanism design problem without money.

10.1.2 Applying the TTC Algorithm

Can we model kidney exchange as a house allocation problem (Section 9.4)? The idea is to treat each patient-donor pair as an agent, with the incompatible living donor acting as a house. A patient's

[1] It is no coincidence that Iran also does not have a waiting list for kidneys. Will other countries eventually follow suit and permit a monetary market for kidneys?

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

total ordering over the donors can be defined according to the estimated probability of a successful kidney transplant, based on the blood type, the tissue type, and other factors.

The hope is that the TTC algorithm finds cycles like that in Figure 10.2, where with patient-donor pairs as in Figure 10.1, each patient points to the other's donor as her favorite. Reallocating donors according to this cycle corresponds to the kidney exchange in Figure 10.1. More generally, the reallocation of donors to patients suggested by the TTC algorithm can only improve every patient's probability of a successful transplant (Theorem 9.8).

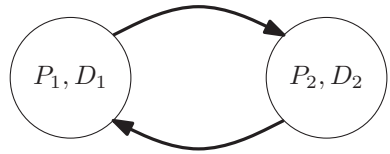

Figure 10.2: A good case for the TTC algorithm. Each circle represents an incompatible patient-donor pair, and each arrow represents a kidney transplant from the donor in the first pair to the patient in the second pair.

The dealbreaker is that the TTC algorithm may suggest reallocations that use long cycles, as in Figure 10.3. Why are long cycles a problem? A cycle of length two (Figure 10.2) already corresponds to four surgeries—two to extract donors' kidneys, and two to implant them in the patients. Moreover, these four surgeries *must happen simultaneously*. Incentives are the reason: in the example in Figure 10.1, if the surgeries for P1 and D2 happen first, then there is a risk that D1 will renege on her offer to donate her kidney to P2.[2] One problem is that P1 unfairly got a kidney for free. The much more serious problem is that P2 is as sick as before and, since her donor D2 donated her kidney, P2 can no longer participate in a kidney exchange. Because of this risk, non-simultaneous surgeries are almost never used in kidney exchange.[3] The constraint of simultaneous surg-

[2]Just as it is illegal to sell a kidney for money in most countries, it is also illegal to write a binding contract for a kidney donation.

[3]Sequential surgeries are used in a slightly different situation. There are a handful of altruistic living donors who want to donate a kidney even though they don't personally know anyone who needs one. An altruistic living donor can be the start of a chain of reallocations. Chains as long as 30 have been implemented,

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

eries, with each surgery needing its own operating room and surgical team, motivates keeping reallocation cycles as short as possible.

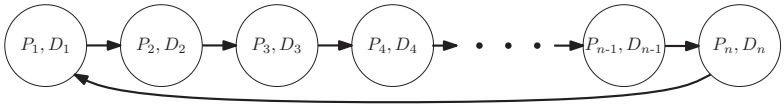

Figure 10.3: A bad case for the TTC algorithm.

Another critique of the TTC approach is that modeling a patient's preferences as a total ordering over donors is overkill. Patients don't really care which kidney they get as long as it is compatible. Binary preferences over donors are therefore more appropriate.

10.1.3 Applying a Matching Algorithm

The twin goals of binary preferences and short reallocation cycles suggest using graph matching. A *matching* of an undirected graph is a subset of edges that share no endpoints. The relevant graph for kidney exchange has a vertex set V corresponding to incompatible patient-donor pairs (one vertex per pair), and an undirected edge between vertices $(P1, D1)$ and $(P2, D2)$ if and only if P1 and D2 are compatible and P2 and D1 are compatible. Thus, the example in Figure 10.1 corresponds to the undirected graph in Figure 10.4. A matching in this graph corresponds to a collection of pairwise kidney exchanges, each involving four simultaneous surgeries. Maximizing the number of compatible kidney transplants corresponds to maximizing the size of a matching.

How do incentives come into play? We assume that each patient has a set E_i of compatible donors belonging to other patient-donor pairs, and can report any subset $F_i \subseteq E_i$ to a mechanism. Proposed kidney exchanges can be refused by a patient for any reason, so one way to implement a misreport is to refuse exchanges in $E_i \setminus F_i$. A patient cannot credibly misreport extra donors with whom she is incompatible. We assume that every patient has binary preferences, preferring every outcome where she is matched to every outcome where she is not.

and at this scale surgeries have to be sequential. With a chain initiated by an altruistic living donor, there is no risk of a patient losing her living donor before receiving a kidney.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

Figure 10.4: Applying a matching algorithm. Each circle represents an incompatible patient-donor pair, and each edge represents a pairwise kidney exchange, meaning transplants from each donor to the patient in the other pair.

The mechanism design goal is to maximize the number of kidney transplants. Direct-revelation solutions have the following form.

A Mechanism for Pairwise Kidney Exchange

1. Collect a report F_i from each agent i.
2. Form the graph $G = (V, E)$, where V corresponds to agent-donor pairs and $(i, j) \in E$ if and only if the patients corresponding to i and j report as compatible the donors corresponding to j and i, respectively.
3. Return a maximum-cardinality matching of the graph G.

Is this mechanism DSIC, in the sense that truthful reporting of the full set E_i is a dominant strategy for each patient i?[4] The answer depends on how ties are broken between different maximum matchings in the third step. There are two senses in which the maximum matching of a graph can fail to be unique. First, different sets of edges can be used to match the same set of vertices (see Figure 10.5). Since a patient does not care whom she is matched to, as long as she is matched, there is no reason to distinguish between different matchings that match the same set of vertices. More significantly, different maximum matchings can match different subsets of vertices. For example, in Figure 10.6, the first vertex is in every maximum matching, but only one of the other vertices can be included. How should we choose?

[4]With no payments, every agent is automatically guaranteed nonnegative utility.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

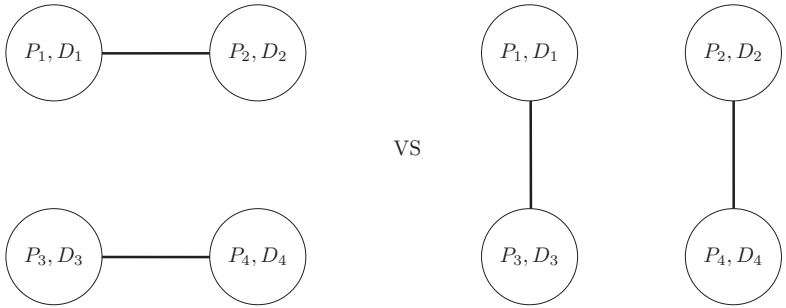

Figure 10.5: Different matchings can match the same set of vertices.

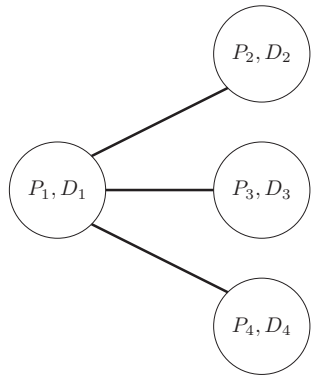

Figure 10.6: Different maximum matchings can match different subsets of vertices.

One solution is to prioritize the agent-donor pairs before the mechanism begins. Most hospitals already rely on priority schemes to manage their patients. The priority of a patient on a waiting list is determined by the length of time she has been on it, the difficulty of finding a compatible kidney, and other factors.

Precisely, we implement the third step of the mechanism as follows, assuming that the vertices $V = \{1, 2, \ldots, n\}$ of G are ordered from highest to lowest priority.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

Priority Mechanism for Pairwise Kidney Exchange

initialize M_0 to the set of maximum matchings of G
for $i = 1, 2, \ldots, n$ **do**
  let Z_i denote the matchings in M_{i-1} that match vertex i
  if $Z_i \neq \emptyset$ **then**
    set $M_i = Z_i$
  else if $Z_i = \emptyset$ **then**
    set $M_i = M_{i-1}$
return an arbitrary matching of M_n

That is, in each iteration i, we ask if there is a maximum matching that respects previous commitments and also matches vertex i. If so, then we additionally commit to matching i in the final matching. If previous commitments preclude matching i in a maximum matching, then we skip i and move on to the next vertex. By induction on i, M_i is a nonempty subset of the maximum matchings of G. Every matching of M_n matches the same set of vertices—the vertices i for which Z_i is nonempty—so the choice of the matching in the final step is irrelevant.

Exercise 10.1 asks you to prove that the priority mechanism for pairwise kidney exchange is DSIC.

Theorem 10.1 (Priority Mechanism Is DSIC) *In the priority mechanism for pairwise kidney exchange, for every agent i and reports by the other agents, no false report $F_i \subset E_i$ yields a better outcome for i than the truthful report E_i.*

10.1.4 Incentives for Hospitals

Many patient-donor pairs are reported to national kidney exchanges by hospitals, rather than by the pairs themselves. The objective of a hospital, to match as many of its patients as possible, is not perfectly aligned with the societal objective of matching as many patients overall as possible. The key incentive issues are best explained through examples.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

Example 10.2 (Benefits of Full Reporting) Suppose there are two hospitals, H_1 and H_2, each with three patient-donor pairs (Figure 10.7). Edges connect patient-donor pairs that are mutually compatible, as in Section 10.1.3. Each hospital has a pair of patient-donor pairs that it could match internally, without bothering to report them to a national exchange. We don't want the hospitals to execute these internal matches. If H_1 matches 1 and 2 internally and only reports 3 to the exchange, and H_2 matches 5 and 6 internally and only reports 4 to the exchange, then the exchange can't match 3 and 4 and no further matches are gained. If H_1 and H_2 both report their full sets of three patient-donor pairs to the national exchange, then 1, 2, and 3 can be matched with 4, 5, and 6, respectively, and all of the patients receive new kidneys. In general, the goal is to incentivize hospitals to report all of their patient-donor pairs to the exchange, to save as many lives as possible.

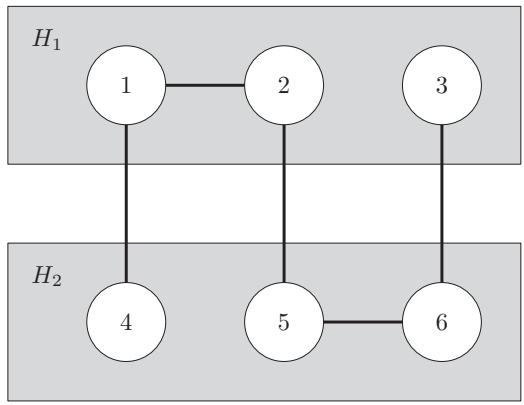

Figure 10.7: Example 10.2. Full reporting by hospitals leads to more matches.

Example 10.3 (No DSIC Maximum Matching Mechanism) Consider again two hospitals, now with seven patients (Figure 10.8). Suppose the exchange always computes a maximum-cardinality matching of the patient-donor pairs that it knows about. With an odd number of vertices, every matching leaves at least one patient unmatched. If H_1 hides patients 2 and 3 from the exchange while H_2 reports truthfully, then H_1 guarantees that all of its patients

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

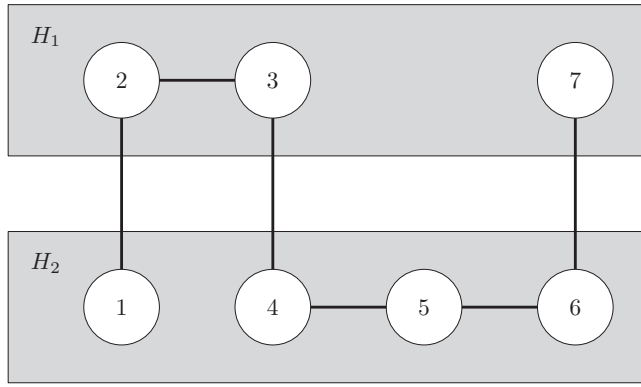

Figure 10.8: Example 10.3. Hospitals can have an incentive to hide patient-donor pairs.

are matched. The unique maximum matching in the reported graph matches patient 6 with 7 (and 4 with 5), and H_1 can match 2 and 3 internally. On the other hand, if H_2 hides patients 5 and 6 while H_1 reports truthfully, then all of H_2's patients are matched. In this case, the unique maximum matching in the reported graph matches patient 1 with 2 and 4 with 3, while H_2 can match patients 5 and 6 internally. We conclude that, no matter which maximum matching the exchange chooses, at least one of the hospitals has an incentive to withhold patient-donor pairs from the exchange. Thus, there is irreconcilable tension between the societal and hospital objectives, and there is no DSIC mechanism that always computes a maximum matching.

Duly warned by Example 10.3, current research on mechanism design for hospital reporting considers relaxed incentive constraints and approximate optimality.

10.2 Stable Matching

Stable matching is the canonical example of mechanism design without money. Killer applications of stable matching include assigning medical school graduates to hospital residencies and students to elementary schools. The following model and algorithm are directly useful for these and other applications with amazingly few modifications.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

10.2.1 The Model

We consider two finite sets V and W of vertices—the "applicants" and the "hospitals"—with equal cardinality. Each vertex has a total ordering over the vertices of the other set. For example, in Figure 10.9, the applicants have a common ranking of the hospitals, while the hospitals have very different opinions about the applicants.

Let M be a perfect matching of V and W, assigning each vertex to one vertex from the other set. Vertices $v \in V$ and $w \in W$ form a *blocking pair for* M if they are not matched in M, v prefers w to her match in M, and w prefers v to its match in M. A blocking pair spells trouble, because the two vertices are tempted to secede from the process and match with each other. A perfect matching is *stable* if it has no blocking pairs.

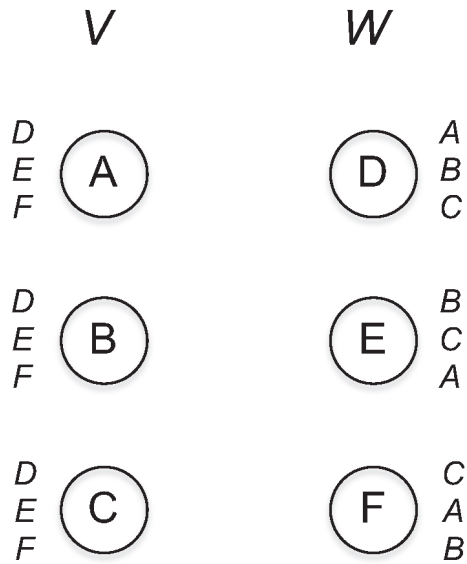

Figure 10.9: An instance of stable matching. Each vertex is annotated with its total ordering over the vertices of the opposite side, with the most preferred vertex on top.

10.2.2 The Deferred Acceptance Algorithm

We next discuss the elegant deferred acceptance algorithm for computing a stable matching.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

Deferred Acceptance Algorithm

while there is an unmatched applicant $v \in V$ **do**
  v attempts to match with her favorite hospital w
  who has not rejected her yet
  if w is unmatched **then**
    v and w are tentatively matched
  else if w is tentatively matched to v' **then**
    w rejects whomever of v, v' it likes less and is
    tentatively matched to the other one
all tentative matches are made final

Example 10.4 (The Deferred Acceptance Algorithm) Consider the instance in Figure 10.9. Suppose in the first iteration we choose the applicant C, who tries to match with her first choice, D. The hospital D accepts because it currently has no other offers. If we pick the applicant B in the next iteration, she also proposes to the hospital D. Since hospital D prefers B to C, it rejects C in favor of B. If we pick applicant A next, the result is similar: D rejects B in favor of A. A possible trajectory for the rest of the algorithm is: applicant C now proposes to her second choice, E; applicant B then also proposes to E, causing E to reject C in favor of B; and finally, C proposes to her last choice F, who accepts.

We note several properties of the deferred acceptance algorithm. First, each applicant systematically goes through her preference list, from top to bottom. Second, because a hospital only rejects an applicant in favor of a better one, the applicants to whom it is tentatively matched only improve over the course of the algorithm. Third, at all times, each applicant is matched to at most one hospital and each hospital is matched to at most one applicant.

Stable matchings and the deferred acceptance algorithm have an astonishing number of remarkable properties. Here are the most basic ones.

Theorem 10.5 (Fast Computation of a Stable Matching) *The deferred acceptance algorithm completes with a stable matching after at most n^2 iterations, where n is the number of vertices on each side.*

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

Corollary 10.6 (Existence of a Stable Matching) *For every collection of preference lists for the applicants and hospitals, there exists at least one stable matching.*

Corollary 10.6 is not obvious a priori. For example, there are some simple variants of the stable matching problem for which a solution is not guaranteed.

Proof of Theorem 10.5: The bound on the number of iterations is easy to prove. Each applicant works her way down her preference list, never trying to match to the same hospital twice, resulting in at most n attempted matches per applicant and n^2 overall.

Next, we claim that the deferred acceptance algorithm always completes with every applicant matched to some hospital (and vice versa). For if not, some applicant must have been rejected by all n hospitals. An applicant is only rejected by a hospital in favor of being matched to a better applicant, and once a hospital is matched to an applicant, it remains matched to some applicant for the remainder of the algorithm. Thus, all n hospitals must be matched at the end of the algorithm. But then all n applicants are also matched at the end of the algorithm, a contradiction.

To prove the stability of the final matching, consider an applicant v and hospital w that are not matched to each other. This can occur for two different reasons. In the first case, v never attempted to match to w. Since v worked her way down her preference list starting from the top, she ends up matched to a hospital she prefers to w. If v did attempt to match to w at some point in the algorithm, it must be that w rejected v in favor of an applicant it preferred (either at the time that v attempted to match to w, or subsequently). Since the sequence of applicants to whom w is matched only improves over the course of the algorithm, it ends up matched to an applicant it prefers to v. ■

*10.3 Further Properties

The deferred acceptance algorithm is underdetermined, leaving open how the unmatched applicant is chosen in each iteration. Do all possible choices lead to the same stable matching? In Figure 10.9 there is only one stable matching, so in that example the answer is

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

yes. In general, however, there can be more than one stable matching. In Figure 10.10, the applicants and the hospitals both disagree on the ranking of the others. In the matching computed by the deferred acceptance algorithm, both applicants get their first choice, with A and B matched to C and D, respectively. Giving the hospitals their first choices yields a different stable matching.

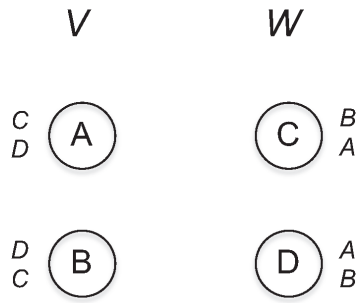

Figure 10.10: There can be multiple stable matchings.

Our next result implies, in particular, that the outcome of the deferred acceptance algorithm is independent of how the unmatched applicant is chosen in each iteration. For an applicant v, let $h(v)$ denote the highest-ranked hospital (in v's preference list) to which v is matched in *any* stable matching.

Theorem 10.7 (Applicant-Optimality) *The stable matching computed by the deferred acceptance algorithm matches every applicant $v \in V$ to $h(v)$.*

Theorem 10.7 implies the existence of an "applicant-optimal" stable matching, where every applicant simultaneously attains her best-case scenario. A priori, there is no reason to expect the $h(v)$'s to be distinct and therefore form a matching.

Proof of Theorem 10.7: Consider a run of the deferred acceptance algorithm, and let R denote the set of pairs (v, w) such that w rejected v at some point. Since each applicant systematically works her way down her preference list, if v is matched to w at the conclusion of the algorithm, then $(v, w') \in R$ for every w' that v prefers to w. Thus, the following claim would imply the theorem: for every $(v, w) \in R$, no stable matching pairs up v and w.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

Let R_i denote the pairs (v, w) such that w rejected v at some point in the first i iterations. We prove by induction on i that no pair in R_i is matched in any stable matching. Initially, $R_0 = \emptyset$ and there is nothing to prove. For the inductive step, suppose that in the ith iteration of the deferred acceptance algorithm, w rejects v in favor of v'. This means that one of v, v' attempted to match to w in this iteration.

Since v' systematically worked her way down her preference list, for every w' that v' prefers to w, $(v', w') \in R_{i-1}$. By the inductive hypothesis, no stable matching pairs up v' with a hospital she prefers to w—in every stable matching, v' is paired with w or a less preferred hospital. Since w prefers v' to v, and v' prefers w to any other hospital she might be matched to in a stable matching, there is no stable matching that pairs v with w (otherwise v', w form a blocking pair). ■

Also, the deferred acceptance algorithm outputs the worst-possible stable matching from the perspective of a hospital (Exercise 10.6).[5]

Suppose the preference lists of the applicants and hospitals are private. Can we obtain a DSIC mechanism by asking all vertices to report their preference lists and running the deferred acceptance algorithm? As Theorem 10.7 might suggest, the deferred acceptance algorithm is DSIC for the applicants but not for the hospitals (see Problem 10.1 and Exercise 10.7).

Theorem 10.8 (Incentive Properties) *Consider the mechanism that runs the deferred acceptance algorithm on reported preferences by the applicants and hospitals.*

(a) For every applicant v and reported preferences by the other applicants and hospitals, v is never strictly better off reporting falsely than truthfully.

(b) There exists preferences for a hospital w and reports for the other applicants and hospitals such that w is strictly better off reporting falsely than truthfully.

[5]Modifying the algorithm so that the hospitals initiate matches and the applicants reject reverses both of these properties.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

The Upshot

- ☆ Kidney exchange enables two or more patients with incompatible living donors to receive kidney transplants from each other's donors.
- ☆ The TTC algorithm can be applied to the kidney exchange problem to exchange donors to improve everyone's compatibility, but the algorithm can result in infeasibly long cycles of exchanges.
- ☆ Matching algorithms can be used to restrict to pairwise kidney exchanges and incentivize patients to accept any donor with whom she is compatible.
- ☆ Hospitals can have an incentive to match incompatible patient-donor pairs internally rather than reporting them to a national exchange.
- ☆ A stable matching pairs applicants and hospitals so that no applicant and hospital would both be better off by matching to each other.
- ☆ The deferred acceptance algorithm computes an applicant-optimal stable matching.
- ☆ The deferred acceptance algorithm leads to a mechanism that is DSIC for the applicants but not for the hospitals.

Notes

The application of the TTC algorithm to kidney exchange is due to Roth et al. (2004). Building on the work of Abdulkadiroğlu and Sönmez (1999) on assigning students to college dorm rooms, Roth et al. (2004) also extend the TTC algorithm and its incentive guarantee to accommodate both deceased donors (houses

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

without owners) and patients without a living donor (agents without houses). The application of graph matching to pairwise kidney exchange is from Roth et al. (2005). Roth et al. (2007) consider three-way kidney exchanges, with simultaneous surgeries on three donors and three patients. Three-way exchanges can significantly increase the number of matched patients, and for this reason are becoming common. Allowing four-way and larger exchanges does not seem to lead to significant further improvements. Sack (2012) describes a chain of 30 kidney transplants, triggered by an altruistic living donor. Incentives for hospitals are studied by Ashlagi et al. (2015).

Gale and Shapley (1962) formalize the stable matching problem, present the deferred acceptance algorithm, and prove Theorems 10.5 and 10.7. The variant of the algorithm described here, with the unmatched applicants attempting to match one-by-one rather than simultaneously, follows Dubins and Freedman (1981). Incredibly, it was later discovered that essentially the same algorithm had been used, since the 1950s, to assign medical residents to hospitals (Roth, 1984)![6] Theorem 10.8 is due to Dubins and Freedman (1981) and Roth (1982a). Exercise 10.6 is observed by McVitie and Wilson (1971), and Problem 10.2 is discussed in Gale and Sotomayor (1985).

Exercises

Exercise 10.1 *(H)* Prove Theorem 10.1.

Exercise 10.2 Exhibit a tie-breaking rule between maximum-cardinality matchings such that the corresponding pairwise kidney exchange mechanism is not DSIC.

Exercise 10.3 Extend Example 10.3 to show that there is no DSIC matching mechanism that always matches more than half of the maximum-possible number of patient-donor pairs.

Exercise 10.4 Prove that there exists a constant $c > 0$ such that, for arbitrarily large n, the deferred acceptance algorithm can require at least cn^2 iterations to complete.

[6] The original implementation was the hospital-optimal version of the deferred acceptance algorithm, but it was changed to favor the applicants in the 1990s (Roth and Peranson, 1999).

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011

Exercise 10.5 Suppose each applicant and hospital has a total ordering over the vertices of the opposite side and also an "outside option." In other words, vertices can prefer to go unmatched over some of their potential matches.

(a) Extend the definition of a stable matching to accommodate outside options.

(b) Extend the deferred acceptance algorithm and Theorem 10.5 to compute a stable matching in the presence of outside options.

Exercise 10.6 *(H)* For a hospital w, let $l(w)$ denote the lowest-ranked applicant (in w's preference list) to whom w is matched in any stable matching. Prove that, in the stable matching computed by the deferred acceptance algorithm, each hospital $w \in W$ is matched to $l(w)$.

Exercise 10.7 Exhibit an example that proves Theorem 10.8(b).

Problems

Problem 10.1 *(H)* Prove Theorem 10.8(a).

Problem 10.2 Consider a hospital in the deferred acceptance algorithm. We say that one preference list *strictly dominates* another if the former always leads to at least as good an outcome as the latter for the hospital (holding the reports of other applicants and hospitals fixed), and in at least one case leads to a strictly better outcome. Prove that, for every hospital, every preference list with a misreported first choice is strictly dominated by a preference list with a truthfully reported first choice.

Downloaded from https://www.cambridge.org/core. McGill University Libraries, on 08 Oct 2019 at 14:55:45, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/CBO9781316779309.011