One-Sided Matching Markets
(DRAFT: Not for distribution)

Federico Echenique[*]

Nicole Immorlica[†]

Vijay V. Vazirani[‡]

1 Introduction

A *one-sided matching allocation problem* consists of a set A of n agents, a set G of n indivisible goods and preferences of agents over goods[1]. The problem asks for a matching of each agent to a distinct good so as to ensure desirable normative properties, such as Pareto optimality, individual rationality and envy-freeness. A market mechanism is often used to address this problem; such a mechanism should have game-theoretic properties such as strategyproofness and core stability. Additionally, in the interest of usability of this mechanism in real world applications, the mechanism needs to run in polynomial time.

One-sided matching markets can be classified along two dimensions: the nature of the utility functions of agents and whether agents have initial endowments or not. As stated in Chapter **??**, utility functions may either have only an ordinal component, i.e., they are modeled via preference relations, or they may have a cardinal component as well. Thus we get four possibilities, which are summarized below, together with the most well-known mechanism(s) for each.

1. *(Ordinal, No Endowments):* Random Priority and Probabilistic Serial
2. *(Ordinal, Endowments):* Top Trading Cycle
3. *(Cardinal, No Endowments):* Hylland-Zeckhauser
4. *(Cardinal, Endowments):* ϵ-Approximate ADHZ

The two types of utility functions described above have their individual pros and cons, and neither dominates the other. Whereas the former are easier to elicit, the latter are more expressive, enabling an agent to not only report if she prefers one good to another but also by how much, thereby producing higher quality allocations as illustrated in Example 20.

[*]Division of the Humanities and Social Sciences, California Institute of Technology, Pasadena, CA 91125.
[†]Microsoft Research Lab – New York City, New York, NY 10011.
[‡]Department of Computer Science, University of California at Irvine, Irvine, CA 92697.
[1]This chapter will appear in:
Online and Matching-Based Market Design. Federico Echenique, Nicole Immorlica and Vijay V. Vazirani, Editors. Cambridge University Press. © 2021.

In recent decades, fundamental computer science revolutions of the Internet and mobile computing have led to the introduction of several novel and impactful marketplaces which are based on one-sided matching markets. Several of these applications naturally lend themselves to the *online model of computation* in which buyers arrive online, one at a time, and need to be matched instantaneously and irrevocably to goods, without knowledge of future arrivals. The matching produced by a mechanism for this setting is compared with the optimal matching computed with full knowledge of all arrivals.

In Section 7, we present a paradigm-setting problem for such matching markets, namely online bipartite matching, as well an optimal mechanism, called RANKING, for it. Internet-based applications using these ideas are studied in Chapters **??** and **??**; the first studies mobile-computing-based applications and the second studies ad auctions, e.g., the AdWords marketplace of Google.

2 Preliminaries

A *precedence relation* of agent $i \in A$ is a total order $\succ_i$ over the goods in G. The *ex post* allocation produced by the one-sided matching allocation problem is a one-to-one mapping of the n agents to the n goods. However, *ex ante* it may be a *random allocation* in which each good is viewed as one unit of probability shares and each agent's allocation is a total of one unit of probability shares over all goods, i.e., it is an $n \times n$ doubly stochastic matrix.

One way to view the random allocation is as a way for agents to time-share the goods. Alternatively, using the Birkhoff-von Neumann Theorem, this *ex ante* allocation is equivalent to a probability distribution over *ex post* mappings of agents to goods.

Over the set of all possible random allocations that can be made to agent i, there exists a natural partial order but no total order. The partial order is defined below. Let x be an *ex ante* random allocation to all agents. Then x_i will denote the allocation made to agent i.

Definition 1. Let x and y be two random allocations. Assume that the components of x_i and y_i are ordered by $\succ_i$. Then x_i *stochastically dominates* y_i if

$$\forall k \in \{1, ..., n\} : \sum_{j=1}^{k} x_{ij} \geq \sum_{j=1}^{k} y_{ij}.$$

We will denote this by $x_i \succeq_i^{sd} y_i$. Furthermore, x_i *strictly stochastically dominates* y_i if x_i stochastically dominates y_i and for some k, with $1 \leq k < n$, $\sum_{j=1}^{k} x_{ij} > \sum_{j=1}^{k} y_{ij}$. We will denote this by $x_i \succ_i^{sd} y_i$. Given two random allocations x and y, we will say that y is *stochastically dominated by* x if for each agent i, $x_i \succeq_i^{sd} y_i$ and for at least one agent j, $x_j \succ_j^{sd} y_j$.

For *ex ante* random allocations, the "correct" notion of efficiency is *ordinal efficiency* as defined below.

Definition 2. Random allocation x is said to be *envy free* if for any two agents i and j, agent i's allocation stochastically dominates agent j's, i.e., $x_i \succeq_i^{sd} x_j$. x is *ordinally efficient* if x is not stochastically dominated by another random allocation x'.

Definition 3. Let x be the random allocation produced by a mechanism when agent i reports her preference relation truthfully and the rest report whatever they wish. Let x' be the random allocation produced when agent i misreports her preference relation and the rest do not change. We will say that the mechanism is *strategyproof* if $x_i \succeq_i^{sd} x'_i$.

Definition 4. Let x and y be two *ex post* allocations, i.e., mappings of agents to goods. We will say that y *dominates* x if for each agent i, $y_i \succeq_i x_i$ and for at least one agent j, $y_j \succ_j x_j$; we will denote this as $y \succ x$. We will say that x is *ex post Pareto efficient* if there is no mapping of agents to goods, y, such that $y \succ x$.

3 Random Priority and Probabilistic Serial: (Ordinal, No Endowments)

The two mechanisms presented in this section have different pros and cons, and neither one dominates the other.

3.1 Random Priority (RP)

As a prelude to Random Priority, we present the following particularly simple deterministic mechanism which assigns goods integrally.

The Priority Mechanism: This mechanism is also called *Serial Dictatorship*. Initially, all n good are declared available. Pick an ordering π of the n agents and in the i^{th} iteration, let the agent $\pi(i)$ pick her most preferred good among the currently available goods; declare the chosen good unavailable.

The priority mechanism is not envy-free: if several agents prefer the same good the most, the one earliest in π will get it.

Lemma 5. *The priority mechanism is strategyproof and the allocation produced by it is Pareto optimal.*

Proof. Strategyproofness is straightforward: In each iteration, the active agent has the opportunity of obtaining the best available good, according to her preference list. Therefore, misreporting preferences can only lead to a suboptimal allocation.

We next show Pareto optimality. Let μ be the allocation produced by the priority mechanism. For contradiction, assume that μ' is an allocation that dominates μ. Let i be the first index such that $\mu'(\pi(i)) \succ_{\pi(i)} \mu(\pi(i))$. Clearly, for $j < i$, agent $\pi(j)$ is assigned the same good under μ and μ'. Therefore, in the i^{th} iteration, agent $\pi(i)$ has available good $\mu'(\pi(i))$. Since $\pi(i)$ picks the best available good, $\mu(\pi(i)) \succeq_{\pi(i)} \mu'(\pi(i))$, leading to a contradiction. $\square$

The priority mechanism suffers from the obvious drawback of not being fair since agents at the top of the list π have the opportunity of choosing their favorite goods while those at the bottom get the left-overs. RP corrects this as follows.

The RP Mechanism: This mechanism is also called *Random Serial Dictatorship*. It iterates over all $n!$ orderings of the n agents. For each ordering π, it runs the priority mechanism and when an agent chooses a good, it assigns a $\frac{1}{n!}$ share of the good to the agent. Clearly, at the end of the process, each agent is assigned a total of one unit of probability shares over all goods.

Lemma 6. *RP is strategyproof and* ex post *Pareto optimal.*

Proof. By Lemma 5, in each of the $n!$ iterations, truth-revealing is the best strategy of an agent. Therefore, RP is strategyproof.

We next argue that the random allocation output by RP can be decomposed into a convex combination of perfect matchings of agents to goods such that the allocation made by each perfect matching is Pareto optimal. This is obvious if we choose the $n!$ perfect matchings corresponding to the $n!$ orderings of agents. Clearly, two different orderings may yield the same perfect matching, therefore the convex combination may be over fewer than $n!$ perfect matchings. Therefore, RP is *ex post* Pareto optimal. $\square$

The reader may have surmised that for a similar reasoning as in Lemma 6, the Pareto optimality of the priority mechanism would lead to ordinal efficiency of RP. Interestingly enough, that is not the case, as shown in Example 7. Even worse, RP is not envy-free, despite the fact that the reason for generalizing from priority to RP was to introduce fairness, see Example 8.

Although the priority mechanism runs efficiently, RP takes exponential time, making it unpractical, for all but very small values of n, to obtain the *ex ante* random allocation which is useful for time-sharing of the goods. However, if an integral matching of agents to goods is desired, then picking one ordering of agents at random suffices; clearly this is time-efficient.

Example 7. Let $n = 4$, $A = \{1, 2, 3, 4\}$ and $G = \{a, b, c, d\}$. Let the preferences of the agents be as follows.

$$\begin{array}{llll} 1: & a & b & c & d \\ 2: & a & b & c & d \\ 3: & b & a & d & c \\ 4: & b & a & d & c \end{array}$$

Then RP will return the following random allocation.

Agent	a	b	c	d
1	5/12	1/12	5/12	1/12
2	5/12	1/12	5/12	1/12
3	1/12	5/12	1/12	5/12
4	1/12	5/12	1/12	5/12

However, it is stochastically dominated by the following random allocation.

Agent	a	b	c	d
1	1/2	0	1/2	0
2	1/2	0	1/2	0
3	0	1/2	0	1/2
4	0	1/2	0	1/2

Example 8. Let $n = 3$, $A = \{1, 2, 3\}$ and $G = \{a, b, c\}$. Let the preferences of the agents be as follows.

$$\begin{array}{llll} 1: & a & b & c \\ 2: & b & a & c \\ 3: & b & c & a \end{array}$$

Then RP will return the following allocation.

Agent	a	b	c
1	5/6	0	1/6
2	1/6	3/6	2/6
3	0	3/6	3/6

Since the total allocation of the two goods a and b to agents 1 and 2 is 5/6 and 4/6, respectively, agent 2's allocation does not stochastically dominate agent 1's allocation.

3.2 Probabilistic Serial (PS)

With respect to the four properties studied above for RP, PS behaves in exactly the opposite manner; it is time-efficient, ordinally efficient, and envy-free but not strategyproof.

The PS Mechanism: As before, we will view each good as one unit of probability shares. Furthermore, for ease of presentation, let us think of probability as a fluid which can be "poured" at the rate of one unit per hour. In one hour, PS will assign one unit of probability share to each agent using the following continuous process. Initially, each agent is allocated probability from her most preferred good; clearly, if m agents are simultaneously being allocated good j, then j is getting depleted at rate m units per hour. As soon as an agent's preferred good is fully allocated, she moves on to her most preferred good that is still available.

Since each agent has a total order over all n goods, at the end of one hour, each agent will get a unit of allocation and all goods will be fully allocated. By iteratively computing the time at which a good will get exhausted, this continuous process can be discretized. It is easy to see that the allocation computed can be written using rational numbers and the mechanism runs in polynomial time.

Lemma 9. *The allocation computed by PS is envy-free.*

Proof. At each point in the algorithm, each agent is obtaining probability share of her most favorite good. Therefore, at any time in the algorithm, agent i cannot prefer agent j's current allocation to her own. Hence PS is envy-free. $\square$

For showing that the random allocation computed by PS is ordinally efficient, we will appeal to the following property of stochastic dominance which follows directly from its definition. Assume that x and y are two allocations made to agent i having equal total probability shares; let $t \leq 1$ be this total. Assume $x \succeq_i^{sd} y$. Let $\alpha < t$ and remove α amount of the least desirable probability shares from each of x and y to obtain x' and y', respectively. Then $x' \succeq_i^{sd} y'$.

Lemma 10. *The random allocation computed by PS is ordinally efficient.*

Proof. During the run of PS on the given instance, let $t_0 = 0, t_1, \ldots, t_m = 1$ be the times at which some agent exhausts the good she is currently being allocated. By induction on k, we will prove that at time t_k, the partial allocation computed by PS is ordinally efficient among all allocations which give t_k amount of probability shares to each agent.

The induction basis, for $k = 0$, is obvious since the empty allocation is vacuously ordinally efficient. Let A^k denote the allocation at time t_k and let A_i^k denote the allocation made to agent i under A^k. Assume the induction hypothesis, namely that the assertion holds for k, i.e., A^k is ordinally efficient.

For the induction step, we need to show that A^{k+1} is ordinally efficient. Suppose not and let it be stochastically dominated by random allocation P. Let $\alpha = t_{k+1} - t_k$. For each agent i, remove α amount of the least desirable probability shares from P_i to obtain P_i'. Since $P_i \succeq_i^{sd} A_i^{k+1}$, by the property stated above, $P_i' \succeq_i^{sd} A_i^k$. By the induction hypothesis, $A_i^k \succeq_i^{sd} P_i'$ as well. Therefore, $P_i' = A_i^k$. In the time period between t_k and t_{k+1}, each agent obtains α units of probability shares of her most preferred good remaining. Therefore, $A_i^{k+1} \succeq_i^{sd} P_i$, leading to a contradiction. $\square$

To prove that a mechanism is strategyproof, we would need to show that x_i stochastically dominates x_i' where x and x' are the allocations when i reports preferences truthfully and misreports, respectively. Example 11 shows that this does not hold for PS.

Example 11. Let $n = 3$, $A = \{1, 2, 3\}$ and $G = \{a, b, c\}$. Let the preferences of the agents be as follows.

$$\begin{array}{llll} 1: & a & b & c \\ 2: & a & c & b \\ 3: & b & a & c \end{array}$$

PS will return the following allocation.

Agent	a	b	c
1	1/2	1/4	1/4
2	1/2	0	1/2
3	0	3/4	1/4

However, if Agent 3 lies and reports her preference list as $a \succ b \succ c$, then PS will return the following allocation.

Agent	a	b	c
1	1/3	1/2	1/6
2	1/3	0	4/6
3	1/3	1/2	1/6

Therefore by lying, agent 3 obtains 5/6 units of her favorite two goods, instead of only 3/4 units.

Finally, we provide an example in which RP and PS are Pareto incomparable in the sense that different agents prefer different allocations.

Example 12. Let $n = 4$, $A = \{1, 2, 3, 4\}$ and $G = \{a, b, c, d\}$. Let the preferences of the agents be as follows.

$$\begin{array}{lllll} 1: & a & b & c & d \\ 2: & a & b & d & c \\ 3: & b & a & c & d \\ 4: & c & d & a & b \end{array}$$

RP will return the following allocation.

Agent	a	b	c	d
1	1/2	1/6	1/12	1/4
2	1/2	1/6	0	1/3
3	0	2/3	1/12	1/4
4	0	0	5/6	1/6

PS will return the following allocation.

Agent	a	b	c	d
1	1/2	1/6	1/9	2/9
2	1/2	1/6	0	1/3
3	0	2/3	1/9	2/9
4	0	0	7/9	2/9

Agents 1 and 3 prefer the PS allocation, agent 4 prefers the RP allocation and agent 2 is indifferent.

4 Top Trading Cycle: (Ordinal, Endowments)

For this section, assume that the goods in G are individual houses owned by the n agents in A. The house owned by agent i is her initial endowment and will be called i. Each agent i has a total preference list $\succ_i$ over all n houses, including her own. The mechanism *top trading cycle (TTC)* presented below is iterative and in each iteration, it constructs a directed graph $H = (U, E)$ where U is the set of unallocated agents. Each agent $i \in U$ has one out-edge (i, j), where j is the best available house under i's preference list; E consists of precisely these edges. In particular, if i's best available house is i, then E contains the self-loop (i, i).

The TTC Mechanism: Initially, all houses are *available* and all agents are *unallocated*. An iteration is executed as follows: Construct graph $H = (U, E)$ defined above. Pick a cycle in H, say C, and *allocate the cycle*: for each edge $(i, j) \in C$, allocate house j to i. All agents in C are declared *allocated* and houses in C are declared *unavailable*. The mechanism terminates when $U = \emptyset$.

Since every node of H has one outgoing edge, H must contain a cycle. Therefore, the step "pick a cycle in H" will be successful. When C is allocated, the houses declared unavailable are the ones owned by agents in C. Therefore, in each iteration the available houses are precisely the initial endowments of agents in U.

In any iteration, H contains of a set of disjoint cycles. Each vertex that is not in a cycle is on a directed path that ends at one of these cycles. Suppose $(i, j) \in E$ with j in cycle C and $i \notin C$.

When C gets allocated, i will point to another vertex, say k. This will lead to the formation of a new cycle if there was a path from k to i. Observe that once a cycle gets formed in H, it remains unchanged until it is allocated. Therefore the output of the algorithm is independent of the order in which it allocates cycles. Moreover, we can also allocate all cycles simultaneously in each iteration.

If in some iteration i's best house is i, then in this or a future iteration, i will be allocated house i. Therefore, no agent will be allocated a house which she prefers less than her own and hence the allocation computed is individually rational. On the other hand, TTC is not envy-free: suppose i and i' both like house j the best, then at most one of them will be given j.

We next define the fundamental cooperative game theoretic solution concept of core.

Definition 13. A allocation μ of houses to all agents is said to be in the *core* if for any set $S \subseteq A$, if the agents in S exchange houses among themselves via an allocation, say μ', then it cannot be that all agents of S are as happy under μ' as under μ and at least one agent is strictly happier.

Next we will show that the allocation found by TTC is core stable, i.e., lies in the core; this is stronger than Pareto optimality. In the two-sided matching market model studied in Chapter **??**, core stability is equivalent to stability because only pairs of agents can generate value.

Theorem 14. *Mechanism TTC is core stable, i.e., S cannot do better under μ' than μ in the following sense:*

$$\forall i \in S,\ \mu'(i) \succeq_i \mu(i) \quad \text{and} \quad \exists j \in S \text{ s.t. } \mu'(j) \succ_j \mu(j).$$

Proof. Assume for contradiction that the above statement holds. Consider the first iteration of TTC in which an agent in S is allocated a house, and let C be the cycle found. If C contains an agent from $A \setminus S$ then there is an edge $(i,j) \in C$ such that $i \in S$ and $j \in (A \setminus S)$. Clearly, i prefers j to any house in S, $\mu(i) = j$ and $\mu'(i) \in S$. Therefore, $\mu(i) \succ_i \mu'(i)$, leading to a contradiction.

Therefore, C is entirely within S. But then μ will assign each agent i in C her best house in S. Therefore, $\mu(i) \succeq_i \mu'(i)$ hence giving $\mu(i) = \mu'(i)$. On removing all agents of C from S and applying the argument to the remaining set, we get that there is no agent $j \in S$ *s.t.* $\mu'(j) \succ_j \mu(j)$, leading to another contradiction. The theorem follows. □

Theorem 15. *Mechanism TTC is strategyproof.*

Proof. Let R and R' denote the runs of the mechanism if agent i reports her preferences truthfully or not, respectively. Suppose for contradiction that i gets matched to house i' under R and to j under R', where $j \succ_i i'$. For simplicity, assume that the algorithm allocates all cycles simultaneously in each iteration. Assume that in run R', i is allocated in iteration k. We will show that i will be allocated house j in run R as well, hence leading to a contradiction.

Under R', if edge (i,l), for $l \neq j$, in the first k iterations, this edge is inconsequential to the output of the algorithm; the only edge that is consequential is (i,j). Since there is a cycle C containing (i,j) in iteration k, there is a path from j to i by iteration k. There are two cases: j is demoted or promoted in going from i's true preferences to untrue preferences. In the first case, edge (i,j) is added to H earlier under run R than under run R'. Clearly, the path from j to i will eventually get formed under run R, and therefore cycle C will get formed and i will be allocated j. In the second case, edge (i, j) is added to H later under run R than under run R'. By the time edge (i, j) is added to H, there is already a path from j to i. Therefore, cycle C will get formed in this case as well, and i will be allocated house j. □

5 Hylland-Zeckhauser: (Cardinal, No Endowments)

The Hylland-Zeckhauser (HZ) solution uses the power of a pricing mechanism to arrive at equilibrium prices and allocations. Consequently, prices reflect the relative importance of goods based on the utilities declared by buyers, thereby making allocations as equitable as possible. Furthermore, a pricing mechanism helps prevent artificial scarcity of goods and ensures that scarce goods are allocated to agents who have the most utility for them. The allocation produced by the HZ solution is Pareto optimal and the scheme is incentive compatible in the large.

The agents provide their preferences for goods by stating their von Neumann-Mogenstern utilities. Let u_{ij} represent the utility of agent i for good j. Each agent has 1 dollar with which she buys probability shares. As in RP and PS, the HZ mechanism also views each good as one unit of probability shares, and the allocation forms a *fractional perfect matching* in the complete bipartite graph over vertex sets A and G. Let p_j denote the price of good j. Let x_{ij} be the probability share that agent i receives of good j. Then, $\sum_j u_{ij}x_{ij}$ is the *expected utility* accrued by agent i.

Definition 16. Let x and p denote arbitrary (non-negative) allocations and prices of goods. By *size, cost and value* of agent i's bundle we mean

$$\sum_{j \in G} x_{ij}, \quad \sum_{j \in G} p_j x_{ij} \quad \text{and} \quad \sum_{j \in G} u_{ij} x_{ij},$$

respectively. We will denote these by $\text{size}_x(i)$, $\text{cost}_x(i)$ and $\text{value}_x(i)$, respectively.

Definition 17. (*HZ equilibrium*) Allocations and prices (x, p) form an *equilibrium* for the one-sided matching market stated above if:

1. The *market clears*, namely the total probability share of each good j is 1 unit, i.e., $\sum_i x_{ij} = 1$.
2. The size of each agent i's allocation is 1, i.e., $\text{size}_x(i) = 1$.
3. The cost of the bundle of each agent is *at most* 1 dollar; observe that an agent need not spend her entire dollar.
4. Subject to constraints 2 and 3, each agent i maximizes her expected utility, i.e., maximize $\text{value}_x(i)$, subject to $\text{size}_x(i) = 1$ and $\text{cost}_x(i) \leq 1$.

Equilibrium prices are invariant under the operation of *scaling the difference of prices from 1* in the following sense: Let (x, p) be an equilibrium and fix any $r > 0$. Let p' be such that $\forall j \in G$, $p'_j - 1 = r(p_j - 1)$. Then (x, p') is also an equilibrium, see Exercise 3.

A standard way of fixing the scale is to enforce that the minimum price of a good is zero — this leads to simplicity in certain situations. Observe that the main goal of the HZ mechanism is to yield the "correct" allocations to agents; the prices are simply a vehicle in the market mechanism to achieve this. Hence arbitrarily fixing the scale does not change the essential nature of the problem.

Using Kakutani's fixed point theorem, the following is shown:

Theorem 18. *Every instance of the one-sided market defined above admits an equilibrium.*

In addition to the conditions stated in Definition 17, imposing the following condition ensures that every equilibrium allocation will be Pareto optimal: Each agent is allocated a utility-maximizing bundle of minimum cost, in case there are several. This *minimum cost condition* is used in the proof of Lemma 19; see Exercise 2 for an illustrative example.

Lemma 19. *Every HZ equilibrium allocation satisfying the minimum cost condition is Pareto optimal.*

Proof. Let (x, p) be equilibrium allocations and prices and assume for contradiction that x is not Pareto optimal. Let x' be an allocation which is at least as good as x for each agent and better for at least one agent, say i. Since i must have been allocated a utility maximizing bundle of goods costing at most 1 dollar in the equilibrium, it must be the case that the allocation of i under x' costs more than 1 dollar with respect to prices p.

Now, the total allocation of each good under x and under x' is one unit. Therefore, the total cost of the entire allocation x equals the total cost of x', with respect to prices p. Therefore, there must be an agent, say k, such that the cost of her bundle under allocation x' is less than the cost of her bundle under x. Since k obtained a utility-maximizing bundle of least cost in the equilibrium, k must accrue less utility under x' than under x, leading to a contradiction. $\square$

Finally, if this "market" is large enough, no individual agent will be able to improve her allocation by misreporting utilities nor will she be able to manipulate prices. For this reason, the HZ mechanism is incentive compatible in the large.

The next example illustrates how by using cardinal utilities, the HZ mechanism produces higher quality allocations than any mechanism using ordinal utilities. For a detailed discussion of the notions of ordinal and cardinal utility functions, see Chapter **??**.

Example 20. The instance has three types of goods, T_1, T_2, T_3, and these goods are present in the proportion of (1%, 97%, 2%). Based on their utility functions, the agents are partitioned into two sets A_1 and A_2, where A_1 constitute 1% of the agents and A_2, 99%. The utility functions of agents in A_1 and A_2 for the three types of goods are $(1,\ \epsilon,\ 0)$ and $(1,\ 1-\epsilon,\ 0)$, respectively, for a small number $\epsilon > 0$. The main point is that whereas agents in A_2 marginally prefer T_1 to T_2, those in A_1 overwhelmingly prefer T_1 to T_2. Clearly, the ordinal utilities of all agents in $A_1 \cup A_2$ are the same. Therefore, a mechanism based on such utilities will not be able to make a distinction between the two types of agents. On the other hand, the HZ mechanism, which uses cardinal utilities, will fix the price of goods in T_3 to be zero and those in T_1 and T_2 appropriately so that by-and-large the bundles of A_1 and A_2 consist of goods from T_1 and T_2, respectively.

6 ϵ-Approximate ADHZ: (Cardinal, Endowments)

Following is an extension of the Hylland-Zeckhauser model to the setting in which agents have initial endowments:

Definition 21. (*ADHZ equilibrium*[2]:) We require that the initial endowments form a fractional perfect matching in the complete bipartite graph over agents and goods. For given prices p of goods, each agent i sells her initial endowment to obtain a budget, say b_i; we will enhance HZ by allowing agents to have different amounts of money. Then (p, x) is an *ADHZ equilibrium* if (p, x) is an HZ equilibrium assuming budgets b_i. If so, observe that any positive scaling of p also leads to an equilibrium.

It turns that not every instance of ADHZ admits an equilibrium; see Example 22 and Exercise 5.

Example 22. Consider the ADHZ instance with $n = 3$, agents $A = \{a_1, a_2, a_3\}$, goods $G = \{g_1, g_2, g_3\}$ and utilities as given in Table 1. Observe that goods g_2 and g_3 are identical and a_1 and a_2 have identical utilities. The initial endowments of all agents are identical and contain 1/3 units of each good.

Table 1: Agents' utilities

	g_1	g_2	g_3
a_1	1	0	0
a_2	1	0	0
a_3	0	1	1

Lemma 23. *The ADHZ instance given in Example 22 does not admit an equilibrium.*

Proof. Assume for contradiction that an equilibrium exists and let p be the equilibrium price of good g_1 and q of goods g_2 and g_3. Consider these two cases:

Case 1: $q > 0$. Without loss of generality assume that $q = 1$. If $p \leq 1$, a_1 and a_2 will both demand 1 unit of g_1, leading to a contradiction. Therefore $p > 1$ and the budget of each agent is $p/3 + 2/3$. With this price structure, a_3 will not buy g_1 at all and therefore g_1 is only bought by a_1 and a_2. Additionally, these two agents must together buy 1 unit total of g_2 and g_3, costing a total of $p + 1$. Therefore $2 \cdot (p/3 + 2/3) = p + 1$, implying that $p = 1$ and leading to a contradiction.

Case 2: $q = 0$ and $p > 0$. Now a_3 buys one unit from g_2 and g_3 and g_1 is shared by a_1 and a_2. However, with a budget of $p/3$ each, g_1 and g_2 cannot afford to buy 1 total unit of g_1, leading to a contradiction.

Therefore the instance does not admit an equilibrium. □

One recourse to the lack of an equilibrium in the ADHZ model is the following hybrid between the HZ and ADHZ models.

Definition 24. (*α-slack equilibrium*): Let α be a fixed number with $\alpha \in (0, 1]$. As in ADHZ, agents have initial endowments of goods. For given prices p of goods, the budget of agent i is $m_i = \alpha + (1 - \alpha) \cdot b_i$, where b_i is the value of i's initial endowment. Then (p, x) is an *α-slack equilibrium* if (p, x) is an HZ equilibrium assuming budgets m_i.

A non-trivial proof, using the Kakutani Fixed Point Theorem, gives:

[2]A market model with cardinal utilities in which agents have initial endowments of goods is called an exchange, Walrasian or Arrow-Debreu model; hence the name ADHZ.

Theorem 25. *For any α, with $\alpha \in (0, 1]$, an α-slack equilibrium always exists.*

Another recourse is to use the notion of an approximate equilibrium. This notion has been very successfully used in the study of equilibria, both market and Nash, within computer science. Besides retaining the market as a pure exchange model, this alternative also leads to better computational properties.

Definition 26. (*ϵ-approximate ADHZ equilibrium:*) Let $1 > \epsilon > 0$ be a fixed number. As in ADHZ, agents have initial endowments of goods. For given prices p of goods, fix the budget of agent i to be m_i where

$$(1 - \epsilon) \cdot b_i \leq m_i \leq \epsilon + b_i,$$

where b_i is the value of i's initial endowment. Then (p, x) is an *ϵ-approximate ADHZ equilibrium* if (p, x) is an HZ equilibrium assuming budgets m_i. Furthermore, we require that if any two agents have the the same endowments, have the same budgets.

In Definition 26, the additive error term in the upper bound is needed since otherwise the counterexample given in Example 22 still works, and the multiplicative term in lower bound is useful for ensuring approximate individual rationality.

We note that individual rationality fundamentally clashes with envy-freeness in both ADHZ and ϵ-approximate ADHZ models, as the following example illustrates. Consider the case $n = 2$ with the initial endowments of agents 1 and 2 being two distinct goods, 1 and 2, respectively. Assume both agents prefer good 1 to 2. Then individual rationality entails that agent 1 should get good 1. However, this allocation is not envy-free. For this reason, we define the notion of *equal-type envy-freeness* in exchange markets: it demands envy-freeness only for agents with the same initial endowment.

Theorem 27. *For any ϵ with $1 > \epsilon > 0$, an ϵ-approximate ADHZ equilibrium exists. It is Pareto optimal, approximately individually rational and equal-type envy-free.*

Proof. Since an α-slack equilibrium is also an α-approximate ADHZ equilibrium, we get existence of the latter by Theorem 25. Since an ϵ-approximate ADHZ equilibrium is an HZ equilibrium for budgets defrined in a certain way, Pareto optimality of the former follows the Pareto optimality of the latter. Since every agent has a budget of at least $1 - \epsilon$ times the cost of her initial endowment, her utility can decrease by at most a factor of $1 - \epsilon$. As a result, approximate individual rationality follows. Equal-type envy-freeness follows immediately from the condition that agents with the same endowment have the same budget. $\square$

7 Online Bipartite Matching

Let B be a set of n buyers and S a set of n goods. A bipartite graph $G = (B, S, E)$ is specified on vertex sets B and S, and edge set E, where for $i \in B$, $j \in S$, $(i, j) \in E$ if and only if buyer i *likes* good j. G is assumed to have a perfect matching and therefore each buyer can be given a unique good she likes. Graph G is revealed in the following manner. The n goods are known up-front and the buyers arrive one at a time. When buyer i arrives, the edges incident at i are revealed.

We are required to design an *online algorithm*[3] $\mathcal{A}$ in the following sense. At the moment a buyer i is revealed, the algorithm needs to match i to one of its unmatched neighbors, if any; if all of i's neighbors are matched, i remains unmatched. The difficulty is that the algorithm does not "know" the edges incident at buyers which will arrive in the future and yet the size of the matching produced by the algorithm will be compared to the best *off-line matching*; the latter of course is a perfect matching. The formal measure for the algorithm is defined below.

7.1 The competitive ratio of online algorithms

Definition 28. Let $G = (B, S, E)$ be a bipartite graph as specified above. The *competitive ratio* of a deterministic algorithm $\mathcal{A}$ for the online bipartite matching problem is defined to be:

$$c(\mathcal{A}) = \min_{G=(B,S,E)} \min_{\rho(B)} \frac{\mathcal{A}(G, \rho(B))}{n},$$

where $\rho(B)$ is a specific order of arrival of vertices in B and $\mathcal{A}(G, \rho(B))$ denotes the size of the matching produced by algorithm $\mathcal{A}$ when run on G under the order of arrival of vertices in B given by $\rho(B)$.

Remark 29. The assumption that G has a perfect matching is for the sake of simplicity. It is easy to see that the entire ensuing discussion holds even if n is taken to be the size of a maximum matching in G.

Observe that the competitive ratio measures the "worst case" performance of algorithm $\mathcal{A}$. A convenient way of talking about the worst case is to assume that the graph and the order of arrival of buyers are chosen by an *adversary* who has full knowledge of our algorithm.

We have assumed that $\mathcal{A}$ is *greedy*, i.e., if on arrival, buyer i is incident at an unmatched good, then $\mathcal{A}$ will definitely match i to one of them. This is without loss of generality since by leaving i unmatched, $\mathcal{A}$ can reap an extra advantage of at most one edge later; however, it has definitely lost an edge now. As a result, $\mathcal{A}$ produces a *maximal matching*, i.e., a matching that cannot be extended to a valid matching by adding any edge of G. Since the size of a maximal matching is at least half that of a maximum matching, $c(\mathcal{A}) \geq \frac{1}{2}$; furthermore, any deterministic algorithm can be forced by the adversary to produce a matching of this size, see Exercise 6. Therefore, the ratio of $\frac{1}{2}$ is *tight for any deterministic algorithm*.

To do better, we will resort to randomization. We first extend the definition of *competitive ratio for the randomized setting*:

$$c(\mathcal{A}) = \min_{G=(B,S,E)} \min_{\rho(B)} \frac{\mathbb{E}[\mathcal{A}(G, \rho(B))]}{n},$$

where $\mathbb{E}[\mathcal{A}(G, \rho(B))]$ is the expected size of matching produced by $\mathcal{A}$; the expectation is over the random bits used by $\mathcal{A}$.

Once again, we may assume that the worst case graph and the order of arrival of buyers are chosen by an adversary who knows the algorithm. It is important to note that the algorithm is

[3]In the field of algorithm design, a distinction is made between *offline and online algorithms*: whereas the former assumes that the entire input is presented up-front, the latter assumes that the input is provided piecemeal to the algorithm, which is required to make irrevocable decisions about the arriving input without "knowing" highly relevant information which may arrive in the future.

Algorithm 30. (Algorithm RANKING)

1. **Initialization:** Pick a random permutation, π, of the goods in S.
2. **Online buyer arrival:** When a buyer, say i, arrives, match her to the first unmatched good she likes in the order π; if none, leave i unmatched.

Output the matching, M, found.

provided random bits *after* the adversary makes its choices. This leads to a qualitative contrast between deterministic and randomized online algorithms: whereas in the former, the adversary can effectively change the graph midway, in the latter it cannot, since it cannot simulate the random bits in advance.

Perhaps the simplest idea using randomization is: match buyer i to a random unmatched neighbor. We will call this algorithm RANDOM. Example 31 gives a graph and an order of arrival of buyers such that the expected size of matching produced by RANDOM is $\frac{n}{2} + O(\log n)$, i.e., only marginally better than the worst case performance of a deterministic algorithm.

Example 31. A bipartite graph $G = (B, S, E)$ over n buyers and n goods can be specified via its $n \times n$ adjacency matrix, say M_G. Assume n is even and let M_G have 1s on the diagonal as well as in the $\frac{n}{2} \times \frac{n}{2}$ upper right-hand corner, i.e., the entries (i, j) for $i \in \{1, \ldots, \frac{n}{2}\}$ and $j \in \{(\frac{n}{2} + 1), \ldots, n\}$. The rest of the entries are 0; see Figure 1. The goods will be rows of M_G and buyers will be columns. The adversarial order of arrival of buyers, i.e., columns, is: $n, (n-1), (n-2), \ldots, 2, 1$. We will divide the arrival of columns into two epochs: the arrival of columns $n, (n-1), \ldots, (\frac{n}{2}+1)$ will be called Epoch 1 and that of columns $\frac{n}{2}, \ldots, 2, 1$, Epoch 2.

Observe that G has a perfect matching, namely the edges of diagonal entries of M_G. Furthermore, any greedy online algorithm definitely matches all columns in Epoch 1; let j be one of these columns. We will say that column j got a *good match* if it is matched to row j and a *bad match* otherwise. If there are k good matches, then the size of matching produced is $\frac{n}{2} + k$ because at the end of Epoch 1, there will be k unmatched rows from the top $\frac{n}{2}$ rows of M_G and they will get matched in Epoch 2.

Observe that in Epoch 1, the top-right corner of M_G acts like a "trap" for RANDOM: columns have a high probability of getting matched to a row in the "trap" and a low probability of getting a good match. Formally, $\mathbb{E}[k] = O(\log n)$; Exercise 7 gives a way of proving this bound. Therefore, the competitive ratio of RANDOM is $\frac{1}{2} + O(\frac{\log n}{n})$.

7.2 The algorithm RANKING

A significantly better randomized algorithm is called RANKING and is given as Algorithm 30. Note that this algorithm picks a random permutation of goods only once; indeed, if it were

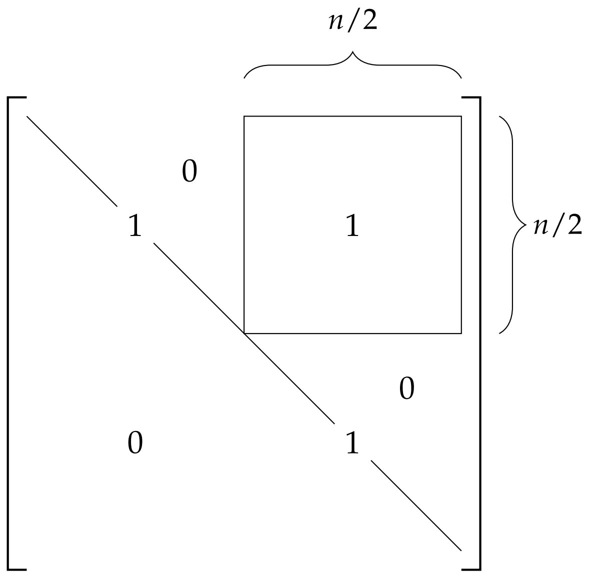

Figure 1: The matrix M_G.

to randomly permute goods with each buyer-arrival, it would be identical to RANDOM. Its competitive ratio is $(1 - \frac{1}{e})$, as shown in Theorem 42. Furthermore, as shown in Theorem 48, it is an *optimal online bipartite matching algorithm*: no randomized algorithm can do better, up to an $o(1)$ term.

Why is RANKING so much better than RANDOM? In order to arrive at an intuitive reason, let us see why the "trap" of Example 31 is not effective anymore. A run of the algorithm involves choosing a random permutation of rows, π; this choice fixes the rest of the outcomes. We will modify the notion of *good match*, given in Example 31, by defining it with respect to each run. The argument given in Example 31 again shows that if in a run there are k good matches, then the size of matching produced is $\frac{n}{2} + k$.

When does column j, which arrives in Epoch 1, get a good match? Assume that there are l rows, with indices in $[1, 2, \ldots, \frac{n}{2}]$, that are above row j under π. Clearly, none of the columns arriving before j can get matched to row j. Observe that column j will get a good match if and only if there are at least l bad matches before the arrival of column j[4], since they will help match off the l rows above row j. Thus each bad match improves the probability of a good match later in the algorithm! This "self-correcting" property has a lot to do with the good performance of RANKING.

In order to prove Theorem 42, we will cast the online bipartite matching problem in an economic setting and restate RANKING as Algorithm 33 which operates as follows. Before the execution of Step (1), the adversary will determine the order in which buyers will arrive, say $\rho(B)$. In Step (1), all goods are assigned *prices* via the specified randomized process. Observe that for each good j, $p_j \in [\frac{1}{e}, 1]$. In Step (2), buyers will arrive in the order $\rho(B)$ and will be matched to the cheapest available good.

[4]A necessary condition on π for this to happen is that $n - j \geq l$.

Remark 32. In Step (1) of Algorithm 33, p_j is taken to be e^{w_j-1}, where w_j is picked uniformly from $[0,1]$. The optimality of the function, e^{w_j-1}, for obtaining the best bound on RANKING, is established in Section 2.1.1 of Chapter **??**; see also Exercise 8.

Algorithm 33 relies on picking a random real number in $[0,1]$ corresponding to each good. For this reason, it is not suitable for the standard (fixed-precision) Turing machine model of computation. On the other hand, with probability 1 all n prices are distinct and sorting the goods by increasing prices results in a random permutation. Furthermore, since Algorithm 33 uses this sorted order only and is oblivious of the actual prices, it is equivalent to Algorithm 30. As we will see, the random variables representing actual prices are crucially important as well – in the analysis.

The *economic setting* is: each buyer i has *unit-demand* and *0/1 valuations* over the goods she likes, i.e., she accrues unit utility from each good she likes, and she wishes to get at most one of them. The latter set is precisely the set of neighbors of i in G. If on arrival of i there are several of these which are still unmatched, i will pick one having the smallest price. As stated above, with probability 1 there are no ties[5]. Therefore the buyers will maximize their utility as defined below.

7.3 Analysis of RANKING

Henceforth, we will call Algorithm 33 by the name of RANKING as well, since it is equivalent to Algorithm 30. For analyzing this algorithm we will define two sets of random variables, u_i for $i \in B$ and r_j, for $j \in S$. These will be called *utility of buyer i* and *revenue of good j*, respectively. Each run of RANKING defines these random variables as follows. If RANKING matches buyer i to good j, then define $u_i = 1 - p_j$ and $r_j = p_j$, where p_j is the price of good j in this run of RANKING. Clearly, p_j is also a random variable, which is defined by Step (1) of the algorithm. If i remains unmatched, define $u_i = 0$, and if j remains unmatched, define $r_j = 0$. Observe that for each good j, $p_j \in [\frac{1}{e}, 1]$ and for each buyer i, $u_i \in [0, 1 - \frac{1}{e}]$.

In this setting, the *consumer surplus* and *producer surplus* are

$$\mathbb{E}\left[\sum_i^n u_i\right] \quad \text{and} \quad \mathbb{E}\left[\sum_j^n r_j\right], \quad \text{respectively,}$$

where the expectations are over the randomization in Step (1) of the algorithm. Let M be the matching produced by RANKING and let random variable $|M|$ denote its size. Then the *total surplus* created by RANKING is $\mathbb{E}[|M|]$.

A key idea in the analysis of Algorithm 33 is a new random variable, u_e, for each edge $e \in E$ of graph G. This is called the *threshold for edge e* and is given in Definition **??**. This random variable can be brought into play once we pull apart the contribution of each matched edge (i,j) into the utility of buyer i and the revenue of good j; this is done in Lemma 34. Next, using the threshold, we established in Lemma 40 that for each edge (i,j) in the graph, the total expected contribution

[5] Of course, in practice we will use Algorithm 30, which has no ambiguity.

Algorithm 33. (Algorithm RANKING: Economic Viewpoint)

1. **Initialization:** $\forall j \in S$: Pick w_j independently and uniformly from $[0,1]$.
   Set price $p_j \leftarrow e^{w_j - 1}$.
2. **Online buyer arrival:** When a buyer, say i, arrives, match her to the cheapest unmatched good she likes; if none, leave i unmatched.

Output the matching, M, found.

of u_i and r_j is at least $1 - \frac{1}{e}$. Then, linearity of expectation allows us to reassemble the $2n$ terms in the right hand side of Lemma 34 so they are aligned with a perfect matching in G, and this yields Theorem 42.

Lemma 34.

$$\mathbb{E}[|M|] = \sum_{i}^{n} \mathbb{E}[u_i] + \sum_{j}^{n} \mathbb{E}[r_j].$$

Proof. By definition of the random variables,

$$\mathbb{E}[|M|] = \mathbb{E}\left[\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} r_j\right] = \sum_{i}^{n} \mathbb{E}[u_i] + \sum_{j}^{n} \mathbb{E}[r_j],$$

where the first equality follows from the fact that if $(i,j) \in M$ then $u_i + r_j = 1$ and the second follows from linearity of expectation. $\square$

While running Algorithm 33, assume that the adversary has picked the order of arrival of buyers, say $\rho(B)$, and Step (1) has been executed. We next define several ways of executing Step (2). Let $\mathcal{R}$ denote the run of Step (2) on the entire graph G. Corresponding to each good j, let G_j denote graph G with vertex j removed. Define $\mathcal{R}_j$ to be the run of Step (2) on graph G_j.

Lemma 35 and Corollary 36 establish a relationship between the sets of available goods for a buyer i in the two runs $\mathcal{R}$ and $\mathcal{R}_j$; the latter is crucially used in the proof of Lemma 38. For ease of notation in proving these two facts, let us renumber the buyers so their order of arrival under $\rho(B)$ is $1, 2, \ldots n$. Let $T(i)$ and $T_j(i)$ denote the sets of unmatched goods at the time of arrival of buyer i (i.e., just before the buyer i gets matched) in the graphs G and G_j, in runs $\mathcal{R}$ and $\mathcal{R}_j$, respectively. Similarly, let $S(i)$ and $S_j(i)$ denote the set of unmatched goods that buyer i is incident to in G and G_j, in runs $\mathcal{R}$ and $\mathcal{R}_j$, respectively.

We have assumed that Step (1) of Algorithm 33 has already been executed and a price p_k has been assigned to each good k. With probability 1, the prices are all distinct. Let L denote the set containing $(1 - p_k)$ for each good j, i.e., $L = \{1 - p_k \mid k \in A\}$. Let L_1 and L_2 be subsets of L containing goods k such that $1 - p_k > 1 - p_j$ and $1 - p_k < 1 - p_j$, respectively.

Lemma 35. *For each i, $1 \leq i \leq n$, the following hold:*

1. $(T_j(i) \cap L_1) = (T(i) \cap L_1)$.
2. $(T_j(i) \cap L_2) \subseteq (T(i) \cap L_2)$.

Proof. Clearly, in both runs, $\mathcal{R}$ and $\mathcal{R}_j$, any buyer i having an available good in L_1 will match to the most profitable one of these, leaving the rest of the goods untouched. Since $j \notin L_1$, the two runs behave in an identical manner on the set L_1, thereby proving the first statement.

The proof of the second statement is by induction on i. The base case is trivially true since $j \notin L_2$. Assume that the statement is true for $i = k$ and let us prove it for $i = k + 1$. By the first statement, the available goods for the $(k+1)^{st}$ buyer in L_1 in the runs $\mathcal{R}$ and $\mathcal{R}_j$ are identical and if there are such goods, the second statement is obviously true. Next assume that there are no such goods. Assume that in run $\mathcal{R}_j$, she gets matched to good l; if she remains unmatched, we will take l to be null. Clearly, l is the most profitable good she is incident to in $T_j(k)$. Therefore, the most profitable good she is incident to in run $\mathcal{R}$ is the best of l, the most profitable good in $T(k) - T_j(k)$, and j, in case it is available. In each of these cases, the induction step holds. $\square$

Corollary 36. *For each i, $1 \leq i \leq n$, the following hold:*

1. $(S_j(i) \cap L_1) = (S(i) \cap L_1)$.
2. $(S_j(i) \cap L_2) \subseteq (S(i) \cap L_2)$.

Next we define a new random variable, u_e, for each edge $e = (i, j) \in E$. This is called the *threshold* for edge e and is given in Definition 37. It is critically used in the proofs of Lemmas 38 and 40.

Definition 37. Let $e = (i, j) \in E$ be an arbitrary edge in G. Define random variable, u_e, called the *threshold* for edge e, to be the utility of buyer i in run $\mathcal{R}_j$. Clearly, $u_e \in [0, 1 - \frac{1}{e}]$.

Lemma 38. *Corresponding to each edge $(i, j) \in E$, the following hold.*

1. *$u_i \geq u_e$, where u_i and u_e are the utilities of buyer i in runs $\mathcal{R}$ and $\mathcal{R}_j$, respectively.*
2. *Let $z \in [0, 1 - \frac{1}{e}]$. Conditioned on $u_e = z$, if $p_j < 1 - z$, then j will definitely be matched in run $\mathcal{R}$.*

Proof. **1).** By Corollary 36, i has more options in run $\mathcal{R}$ as compared to run $\mathcal{R}_j$, and therefore $u_i \geq u_e$.

2). In run $\mathcal{R}$, if j is already matched when i arrives, there is nothing to prove. So assume that j is not matched. The crux of the matter is to prove that i does not have any option that is better than j. Since $p_j < 1 - z$, $S_j(i) \cap L_1 = \emptyset$ and therefore $S(i) \cap L_1 = \emptyset$ giving this fact. Finally, since i gets more utility from j than any good available to her in L_2, she must get matched to j. $\square$

Remark 39. The random variable u_e is called *threshold* because of the second statement of Lemma 38. It defines a value such that whenever p_j is smaller than this value, j is definitely matched in run $\mathcal{R}$.

The intuitive reason for the next, and most crucial, lemma is the following. The smaller u_e is, the larger is the range of values for p_j, namely $[0, 1-u_e)$, over which (i, j) will be matched and j will accrue revenue of p_j. Integrating p_j over this range, and adding $\mathbb{E}[u_i]$ to it, gives the desired bound. Crucial to this argument is the fact that p_j is independent of u_e. This follows from the fact that u_e is determined by run $\mathcal{R}_j$ on graph G_j, which does not contain vertex j.

Lemma 40. *Corresponding to each edge $(i, j) \in E$,*

$$\mathbb{E}[u_i + r_j] \geq 1 - \frac{1}{e}.$$

Proof. By the first part of Lemma 38, $\mathbb{E}[u_i] \geq \mathbb{E}[u_e]$.

Next, we will lower bound $\mathbb{E}[r_j]$. Let $z \in [0, 1-\frac{1}{e}]$ and let us condition on the event $u_e = z$. The critical observation is that u_e is determined by run $\mathcal{R}_j$. This is conducted on graph G_j, which does not contain vertex j. Therefore u_e is independent of p_j.

By the second part of Lemma 38, $r_j = p_j$ whenever $p_j < 1-z$. We will ignore the contribution to $\mathbb{E}[r_j]$ when $p_j \geq 1-z$. Let w be s.t. $e^{w-1} = 1-z$.

When $p_j < 1-z$, the random variable r_j is defined as follows: pick x uniformly at random from $[0, w)$ and let r_j be e^{x-1}, see Figure 2. Therefore,

$$\mathbb{E}[r_j \mid u_e = z] \geq \int_0^w e^{x-1}\, dx \;=\; e^{w-1} - \frac{1}{e} \;=\; 1 - \frac{1}{e} - z.$$

Let $f_{u_e}(z)$ be the probability density function of u_e; clearly, $f_{u_e}(z) = 0$ for $z \notin [0, 1-\frac{1}{e}]$. Therefore,

$$\begin{aligned}\mathbb{E}[r_j] \;&=\; \mathbb{E}[\mathbb{E}[r_j \mid u_e]] \;=\; \int_{z=0}^{1-1/e} \mathbb{E}[r_j \mid u_e = z] \cdot f_{u_e}(z) dz \\ &\geq \int_{z=0}^{1-1/e} \left(1 - \frac{1}{e} - z\right) \cdot f_{u_e}(z) dz \;=\; 1 - \frac{1}{e} - \mathbb{E}[u_e],\end{aligned}$$

where the first equality follows from the law of total expectation and the inequality follows from fact that we have ignored the contribution to $\mathbb{E}[r_j \mid u_e]$ when $p_j \geq 1-z$. Hence we get

$$\mathbb{E}[u_i + r_j] = \mathbb{E}[u_i] + \mathbb{E}[r_j] \geq 1 - \frac{1}{e}.$$

□

Remark 41. Observe that Lemma 40 is not a statement about i and j getting matched to each other, but about the utility accrued by i and the revenue accrued by j by being matched to various goods and buyers, respectively, over the randomization executed in Step (1) of Algorithm 33.

Theorem 42. *The competitive ratio of RANKING is at least $1 - \frac{1}{e}$.*

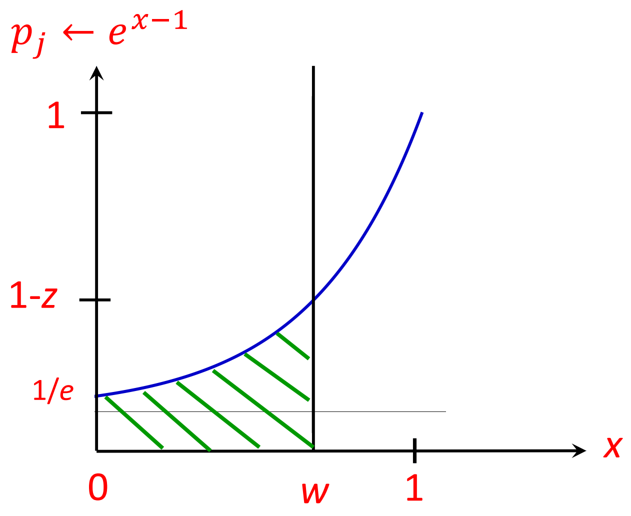

Figure 2: The shaded area is a lower bound on $\mathbb{E}[r_j \mid u_e = z]$.

Proof. Let P denote a perfect matching in G. The expected size of matching produced by RANKING is

$$\mathbb{E}\left[|M|\right] \;=\; \sum_{i}^{n} \mathbb{E}\left[u_i\right] \;+\; \sum_{j}^{n} \mathbb{E}[r_j] \;=\; \sum_{(i,j)\in P} \mathbb{E}[u_i + r_j] \;\geq\; n\left(1 - \frac{1}{e}\right),$$

where the first equality uses Lemma 34, the second follows from linearity of expectation and the inequality follows from Lemma 40 and the fact that $|P| = n$. The theorem follows. □

7.4 Upper-bounding the performance of any randomized algorithm

In this section we will show, using the power of the Minimax Theorem (in particular Lemma 44, which is derived from the Minimax Theorem), that the competitive ratio of *any* randomized online matching algorithm is bounded by $(1 - \frac{1}{e}) + o(1)$, thereby showing that RANKING is the optimal algorithm, up to an $o(1)$ term.

7.4.1 Sets of Algorithms and Inputs

Let I denote an input, involving n buyers and n goods, to RANKING. We will assume that the buyers in I have been ordered in the order of arrival dictated by the adversary. Algorithm 30 is randomized, i.e., it uses the flips of a fair coin to execute its steps. In particular, when run on input I, this algorithm uses the coin flips to randomly permute the n goods. Let $\mathcal{A}$ denote the deterministic part of this algorithm and for each permutation π on $\{1, \ldots, n\}$, let $(\mathcal{A}, \pi)$ denote the algorithm which permutes the n goods according to π before running $\mathcal{A}$ on the given instance.

Consider the collection of $n!$ deterministic algorithms $(\mathcal{A}, \pi)$ for all possible permutations π. Let $\mathcal{A}_1, \ldots, \mathcal{A}_m$, for $m = n!$, be an arbitrary numbering of these deterministic algorithms. Clearly, the expected performance of Algorithm 30 on I is the same as the expected performance of an algorithm picked from the uniform probability distribution over this collection on I.

Consider the collection of all possible graphs n buyers and n goods having a perfect matching. We will assume that the buyers have been renumbered so the adversarial order of arrival is $1, \ldots, n$. This is the set of all possible inputs of size n. Let k be its number and let the set be $\{I_1, \ldots, I_k\}$.

7.4.2 The Minimax Theorem and its useful consequence

In this section, we will introduce *two-player zero sum games* using the sets defined in Section 7.4.1. Let A denote a $k \times m$ matrix with non-negative real entries; A is called the *payoff matrix* of the game. The rows of A represent inputs $I_1, \ldots, I_k$ and its columns represent algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_m$. The $(i,j)^{th}$ entry of A, $A(i,j)$, is the size of matching computed when algorithm $\mathcal{A}_j$ is run on instance I_i.

The two players of the game are the *row player* and the *column player*; they can be thought of as the *adversary* and the *algorithm designer*, respectively. The *pure strategies* of the row (column) player are to pick one of the rows (columns) of A. If the row player picks row i and the column player picks column j, then the row player will pay $A(i,j)$ to the column player.

The column player's goal is to maximize her payoff and the row player's goal is to minimize the amount he pays. The name "zero sum" refers to the fact that the amount won by the column player is precisely the amount lost by the row player.

Observe that the row player should be prepared to pay the maximum entry in row i, since the column player can pick a column having such an entry. Therefore in order to minimize the amount he pays, the row player should pick that row in which the maximum entry is as small as possible, i.e., $\min_i \max_j \ A(i,j)$.

Similarly, in any column j, the column player is guaranteed only the minimum entry, since the row player has the option of picking such a row. Therefore, the column player should pick that column j in which the smallest entry is as large as possible, i.e., $\max_j \min_i \ A(i,j)$. In general, $\min_i \max_j \ A(i,j) \neq \max_j \min_i \ A(i,j)$, see Exercise 9.

We next introduce the notion of mixed strategies, under which equality is guaranteed. A real-valued vector will be called a *probability vector* if its entries are non-negative and add up to 1. A *mixed strategy* of the row (column) player is a probability vector of dimension k (m); henceforth, these will be denoted by p and q, respectively. Thus the row player will pick a row, say i, from distribution p and the column player will pick a column, say j, from distribution q. If so, the *expected payoff* to the column player is $p^T A q$. The central fact about zero-sum games is the following.

Theorem 43. (The Minimax Theorem)

$$\min_p \max_q \ p^T A q \quad = \quad \max_q \min_p \ p^T A q$$

The quantity on either side of the equality in Theorem 43 will be called the *value of game A* and will be denoted by $v(A)$. Once the row player has picked strategy p, the payoff $p^T Aq$ is a linear function of the entries of q and hence is optimized by picking q to be a specific column. A similar remark applies to the choice of p, once q has been picked by the column player. Hence we get the following simpler form:

$$\min_p \max_j \; p^T A e_j \;\; = \;\; \max_q \min_i \; {e_i}^T A q,$$

where e_i and e_j are k and m dimensional unit vectors having 1s in the i^{th} and j^{th} components, respectively.

Next assume that p and q are *arbitrary* probability vectors of dimension k and m, respectively. Then clearly:

$$\max_j \; p^T A e_j \;\; \geq \;\; v(A) \;\; \geq \;\; \min_i \; {e_i}^T A q. \tag{1}$$

We will use inequality (1) to upper bounding the performance of RANKING. We will take q to be the uniform probability distribution on algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_m$. This is equivalent to picking π at random, i.e., it is equivalent to running RANKING on the given input. Therefore, the right hand side is expected size of matching produced by RANKING on the worst input. This is precisely the quantity we wish to upper bound.

Towards this end, we have the freedom of choosing a probability distribution p over all possible inputs of size n. Once we choose p, we need to determine the best algorithm, say $\mathcal{A}_j$, for an input picked from this distribution. Then, according to inequality (1), the expected performance of $\mathcal{A}_j$ on this input provides an upper bound. The distribution p is given in Section 7.4.3.

For now, let us extract a general statement from inequality (1) so it can be used for upper bounding the performance of an arbitrary randomized algorithm. As was done for RANKING, a randomized algorithm can be viewed as a distribution over a suitable collection of deterministic algorithms. Let q represent this distribution and $\mathcal{M}_q$ represent the randomized algorithm we are studying. Then $\mathbb{E}[\mathcal{M}_q(I_i)]$ is the performance of this algorithm on input I_i.

In order to upper bound this performance, we are required to pick a suitable distribution p over the inputs so that the performance of the best algorithm, $\mathcal{M}_j$, on this distribution can be ascertained. The upper bound we get as a consequence is $\mathbb{E}[\mathcal{M}_j(I_p)]$. This is encapsulated in Lemma 44, whose proof follows from inequality (1).

Lemma 44. *Let $\mathcal{M}_q$ and p be the randomized algorithm and probability distribution over inputs defined above. Then,*

$$\max_j \; \mathbb{E}[\mathcal{M}_j(I_p)] \;\; \geq \;\; \min_i \; \mathbb{E}[\mathcal{M}_q(I_i)],$$

i.e., the expected performance of the best deterministic algorithm on an input picked from the chosen distribution is an upper bound on the worst case performance of randomized algorithm $\mathcal{M}_q$.

7.4.3 Upper-bounding the performance of RANDOM

In this section, we will give the distribution p on inputs promised above as well as the best deterministic algorithm for inputs chosen from p. For the distribution given below, it turns out that *every* deterministic greedy algorithm has optimal performance, making the second task somewhat simple.

Let T denote the $n \times n$ *upper triangular matrix*, i.e., $T(i,j) = 1$ if $i \leq j$ and 0 otherwise. For each permutation, π, let (T, π) denote the instance in which the rows of T are permuted according to π, and let q denote the uniform distribution on these $n!$ instances. We will assume that for any instance picked from this distribution, the adversarial order of arrival of columns is $n, (n-1), \ldots, 1$. We will associate a notion of discrete *time* with the arrival of columns. It will be convenient to assume that time starts at n and goes down to 1, i.e., it flows in the order $n, (n-1), \ldots, 1$.

Consider a run of a greedy deterministic algorithm on an instance picked from distribution p and consider the set of rows to which column t can be matched. Clearly, this is the subset of the first t rows of T which are unmatched at time t; observe that T is the unpermuted upper triangular matrix. We will call this the *set of eligible rows at time t*.

The proof of the next lemma follows via an easy induction.

Lemma 45. *Consider a run of a greedy deterministic algorithm on inputs from distribution p and a run of RANDOM on T. Then:*

1. *In both the runs, at each time t, if the set of eligible rows has cardinality k then these rows are equally likely to be any k of the first t rows of T.*
2. *At each time t, the probability that the set of eligible rows has cardinality k is the same for both runs.*

The proof of the next lemma is left as Exercise 10.

Lemma 46. *The performance of any greedy deterministic algorithm on an instance drawn from distribution p is the same as that of RANDOM on T.*

Lemma 47. *The expected size of the matching found by RANDOM when run on instance T is at most $(1 - \frac{1}{e})n + o(n)$.*

Proof. Let $c(t)$ and $r(t)$ be random variables representing the number of eligible columns and rows, respectively, at time t in a run of RANDOM on T. Observe that both random variables will monotonically decrease with decreasing t and the first time $r(t) = 0$, the remaining columns cannot be matched and the algorithm stops. Until this time, the expected change in $c(t)$ is:

$$\Delta(c) = c(t) - c(t+1) = -1,$$

since the column arriving at time t becomes ineligible.

Next, let us obtain an expression for the the expected change in $r(t)$ until $r(t) = 0$. Now, $\Delta(r) = r(t) - r(t+1)$. The number of eligible rows decreases because of two reasons. First, it decreases because the row that is matched to column t becomes ineligible.

An additional decrease takes place if at time t, row t is eligible and is not matched to column t; if so, row t also becomes ineligible. By the first fact given in Lemma 45, the eligible rows at time t are equally likely to be any $r(t)$ out of the first $c(t)$ rows. Therefore the probability that they include row t is $\frac{r(t)}{c(t)}$. Since the algorithm matches column t to a random eligible row, the conditional probability that it matches column t to a row different from t, given that row t is eligible, is $\frac{r(t)-1}{r(t)}$. Taking the product of these two gives the expected additional decrease in $r(t)$ for the second reason. Therefore,

$$\mathbb{E}[\Delta(r)] = -1 - \frac{r(t)}{c(t)} \cdot \frac{(r(t)-1)}{r(t)}.$$

Taking their ratio we get

$$\frac{\mathbb{E}[\Delta(r)]}{\mathbb{E}[\Delta(c)]} = 1 + \frac{r(t)-1}{c(t)}.$$

Applying Kurtz' Theorem, we get that as n tends to infinity, with probability tending to 1, the solution to this difference equation is closely approximated by the solution to the following differential equation with initial condition $c(n) = n$ and $r(n) = n$:

$$\frac{dr}{dc} = 1 + \frac{r(t)-1}{c(t)}.$$

The solution to this differential equation is

$$r = 1 + c\left(\frac{n-1}{n} + \ln\frac{c}{n}\right).$$

It is easy to see that when $r(t) = 1$, $c(t) = \frac{n}{e} + o(n)$. The lemma follows. □

Finally, Theorem 42 and Lemmas 44, 46 and 47 give:

Theorem 48. *RANKING is an optimal online bipartite matching algorithm up to an $o(1)$ term.*

Remark 49. The expected size of the matching found by RANKING and RANDOM is the same for the instance T, see Exercise 11. Therefore, if one could show that T is the worst instance for RANKING, then Lemma 44 and Lemma 46 would directly prove optimality of RANKING.

8 Exercises

1. ([SV15]) Observe that under ordinal utilities, stochastic dominance only leads to a partial order over the set of possible allocations to one agent. Instead, if we compare allocations using lexicographic preferences, we obtain a total order. Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ be two allocations for agent i, where the goods have been ordered per i's preference list. We will say that x is *lexicographically better* than y if $x_k > y_k$ where k is the first index j for which $x_j \neq y_j$. Show that with respect to lexicographic preferences, PS enjoys all four properties of time-efficiency, Pareto optimality, envy-freeness and strategyproofness.

Hint: For strategyproofness, show and use the fact that for any false preference list given by agent i there is some good j that is *sacrificed* in the sense that i obtains less of j in the false run than in the true run, thereby obtaining a lexicographically inferior allocation.

2. Consider the following instance of HZ: it has $n = 2$ and utility functions $u_{11} = u_{21} = u_{22} = 1$ and $u_{12} = 0$. The prices $(2, 0)$ and the allocation $x_{11} = x_{12} = x_{21} = x_{22} = 0.5$ form an equilibrium. However, observe that agent 1 is not allocated a minimum cost utility-maximizing bundle. Find an equilibrium in which the latter property also holds. Then compute the utilities accrued and thereby show that the above-stated allocation is not Pareto optimal.

Hint: Let the prices be $(1, p)$, for any $p \in [0, 1]$.

3. ([VY20]) Let (x, p) be an equilibrium for an HZ instance I. Prove that an HZ equilibrium is invariant under the following operations.

1. *Scaling the difference of prices from 1:* Fix $r > 0$ and obtain p' from p as follows: $\forall j \in G$, $p'_j - 1 = r(p_j - 1)$. Then (x, p') is also an equilibrium.
2. *Affine transformation of the utility function of any agent:* Let i's utility function be $u_i = \{u_{i1}, u_{i12}, \ldots, u_{in}\}$. For any two numbers $s > 0$ and $h \geq 0$, define $u'_i = \{u'_{i1}, u'_{i12}, \ldots, u'_{in}\}$ as follows: $\forall j \in G$, $u'_{ij} = s \cdot u_{ij} + h$. Let I' be the instance obtained by replacing u_i by u'_i in I. Then (p, x) is an equilibrium for I if and only if it is for I'.

   Hint: Start by writing an LP whose optimal solutions capture optimal bundles to an agent i for given prices p.

Remark 50. Observe the fundamentally different ways in which an HZ equilibrium and a Nash bargaining solution change under an affine transformation of the utilities of agents.

4. ([VY20]) Define the *bivalued utilities case* of HZ as follows: For each agent i, we are given a set $\{a_i, b_i\}$, where $0 \leq a_i < b_i$, and the utilities u_{ij}, $\forall j \in G$, are picked from this set. The *dichotomous utilities case* is a special case of the bivalued utilities case in which for each agent i, $a_i = 0$ and $b_i = 1$. Use Exercise 3 to give a reduction from the bivalued utilities case to the dichotomous utilities case of HZ.

5. ([GTV20]) For an exchange market, the *demand graph* is a directed graph with agents as vertices and with an edge (i, j) if for a good k which agent j has in her initial endowment, $u_{ik} > 0$. In the case of an Arrow-Debreu market with linear utilities, a sufficiency condition for the existence of equilibrium is that this graph be strongly connected. Is this condition sufficient to get around counterexamples to the existence of an equilibrium in ADHZ as well? Via a counterexample, provide a negative answer to this question. Additionally, ensure that the counterexample has dichotomous utilities, i.e., $0/1$ utilities

6. Prove that in a bipartite graph, the size of a maximal matching is at least half that of a maximum matching. Via an example, show that the bound of half is tight.

Let $\mathcal{A}$ be any deterministic online bipartite matching algorithm. Show that the adversary can produce a bipartite graph $G = (B, S, E)$ and an order of arrival of buyers $\rho(B)$ on which $\mathcal{A}$ finds a matching of size exactly half the perfect matching.

7. Consider the following $\frac{n}{2}$ experiments and random variables: for $1 \le i \le \frac{n}{2}$, pick a random element from the set $\{0, 1, \ldots, i\}$ and define X_i to be 1 if 0 was picked and 0 otherwise. Prove that

$$\mathbb{E}[k] \le \sum_{i=1}^{n/2} \mathbb{E}[X_i] = O(\log n),$$

where k is defined in Example 31.

8. In Step (1) of Algorithm 33, try other distributions for picking the prices of goods, e.g., the uniform distribution on $[0, 1]$. Is there a distribution that yields a ratio larger than $\left(1 - \frac{1}{e}\right)$ by small amount, i.e., $o(1)$?

9. Let A be the $k \times m$ payoff matrix of a zero-sum game in which the row player is attempting to minimize the amount he pays and the column player is maximizing the amount she gets.

1. Prove that $\min_i \max_j A(i,j) \ \ge \ \max_j \min_i A(i,j)$.
2. Give an example of a game in which $\min_i \max_j A(i,j) \ > \ \max_j \min_i A(i,j)$.
3. Prove the Minimax Theorem using the LP-duality theorem.

10. Prove Lemma 46 by using the two fact given in Lemma 45.

11. Prove that the expected size of matching found by RANKING and RANDOM is the same for the instance T, i.e., the complete upper-triangular matrix.

9 Notes

The mechanisms RP and PS are due to Bogomolnaia and Moulin [BM01], HZ is due to Hylland and Zeckhauser [HZ79], α-Slack is due to Echenique, Miralles and Zhang [EMZ19], and ϵ-Approximate ADHZ is due to Garg, Trobst and Vazirani [GTV20]. The TTC mechanism was discovered by David Gale and reported in [SS74]. Example 22 is from [HZ79].

The result presented in Section 7 is due to Karp, Vazirani and Vazirani [KVV90]. The analysis of RANKING presented here is taken from Vazirani [Vaz21]; it is based on ideas due to Devanur, Jain and Kleinberg [DJK13] and Eden, Feldman, Fiat and Segal [EFFS21]. The Minimax Theorem is due to von Neumann [vN28] and Lemma 44, which builds on the Minimax Theorem, is due to Yao [Yao77]. Kurtz's Theorem is from [Kur70].

10 Bibliography

References

[BM01] Anna Bogomolnaia and Hervé Moulin. A new solution to the random assignment problem. *Journal of Economic theory*, 100(2):295–328, 2001.

[DJK13] Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 101–107. SIAM, 2013.

[EFFS21] Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. An economic-based analysis of ranking for online bipartite matching. In *SIAM Symposium on Simplicity in Algorithms*, 2021.

[EMZ19] Federico Echenique, Antonio Miralles, and Jun Zhang. Constrained pseudo-market equilibrium. *arXiv preprint arXiv:1909.05986*, 2019.

[GTV20] Jugal Garg, Thorben Trobst, and Vijay V Vazirani. One-sided matching markets with endowments: Equilibria and algorithms. *arXiv preprint arXiv:2009.10320*, 2020.

[HZ79] Aanund Hylland and Richard Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political economy*, 87(2):293–314, 1979.

[Kur70] Thomas G Kurtz. Solutions of ordinary differential equations as limits of pure jump markov processes. *Journal of applied Probability*, 7(1):49–58, 1970.

[KVV90] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

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

[SV15] Leonard J Schulman and Vijay V Vazirani. Allocation of divisible goods under lexicographic preferences. In *Foundations of Software Technology and Theoretical Computer Science*, 2015.

[Vaz21] Vijay V. Vazirani. Online bipartite matching and adwords. *arXiv preprint arXiv:2107.10777*, 2021.

[vN28] John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

[VY20] Vijay V. Vazirani and Mihalis Yannakakis. Computational complexity of the Hylland-Zeckhauser scheme for one-sided matching markets. In *Innovations in Theoretical Computer Science*, 2020.

[Yao77] Andrew C. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer science (FOCS)*, pages 222–227, 1977.