Economic Theory
https://doi.org/10.1007/s00199-023-01508-x

RESEARCH ARTICLE

On slots' scheduling

Anna Bogomolnaia[1,2]

Received: 14 June 2022 / Accepted: 26 June 2023

Abstract
Server works in discrete time, and is equipped with a given sequence of per-date capacities. It has to accommodate a set of agents with unit jobs, arriving at different dates. It can process a job in several installments, however no monetary transfers are allowed. Server is given jobs' birth dates and it only knows that agents want their jobs done as soon as possible, but not agents' complete preferences over delays (thus, this is the model with ordinal input). We investigate how scheduling rules, coming from both assignment and queueing literature, fare in this setting. The tension between fairness and incentive compatibility, inherent to the assignment models, disappears on this domain, as both Serial and Random Priority assignment rules become strategy-proof and non-envious. This is also true for Uniform rule; but First Come First Serve or First Come Last Serve rules are not strategy-proof and generate envy.

Keywords Random assignment · Scheduling · Strategy-proofness · Fairness

JEL classification C78 · D47 · D63

1 Introduction

We consider a scheduling problem for a central agency, which we call "server", who owns an ordered set of resources (we refer to them as "slots") and wants to distribute those slots between a group of agents. Server's only interest is agents' welfare. Thus, it wants to allocate slots in an efficient and fair way. As it is often the case for centralized assignment problems, monetary payments are not available (due to legislative or ethical reasons), but fractional allocations are possible.

Each agent needs one unit of resource total, but the server's capacity can vary with the slots (and it is not necessarily integer).

✉ Anna Bogomolnaia
anna.bogomolnaia@glasgow.ac.uk

1 University of Glasgow, Glasgow, UK

2 CES, CNRS, Rennes, France

Published online: 28 July 2023

Agents have common perception of slots' ranking (from best to worst), which is publicly known. However, each agent has her own private limit on what is the highest ranked slot she can accept.

The benchmark example is when slots are dates on the timeline. Think of a machine which can process certain amount of jobs on each day. Jobs arise at different dates, and their owners want them to be done as soon as possible after that. Another case is when all agents want their jobs to be done as late as possible, but before a deadline (an example would be the delivery of perishable goods for scheduled events).

An alternative common ordering is the order of slots/objects by quality, where agents want the best quality, but cannot deal with a too high one. Examples would be firms competing for governmental projects, where each firm wants the largest one among those it can manage due to technical constraints; people wanting to join a sport club or an educational institution with the best reputation among those not too much above their abilities; agents wanting to purchase the best quality good within their budget (or the cheapest good above certain quality level), etc.

Agents only reveal the information on their highest ranked slots acceptable ("birth dates"). Thus, server operates based on ordinal input. Agents' preferences are private information. They can potentially misreport birth dates, and they can have arbitrary preferences over own fractional assignments, when a job is done over several slots (consistent with "earlier better" idea).

We look for suitable allocation rules in this setting. We impose anonymity (or at least ETE) and efficiency on the rules. Those are very natural and mild restrictions in our setting; and, as we will see, efficiency structure turns to be rather simple on our domain.

Our model combines the features of the assignment problem with ordinal input (on a restricted preference domain) and of the queueing problem (with the arrival times known from the beginning). We thus investigate the behavior of allocation rules, coming from both queueing and assignment literatures. Queueing models usually assume that server does not know the future, and so it cannot condition the allocation of a current slot on the potential arrival of future jobs ("no forecast"). However, classical assignment rules also satisfy no-forecast requirement in our model, so all rules we discuss apply equally well to both settings.

Coming from queueing literature, First Come First Serve (FCFS) rules always favor agents born earlier, First Come Last Serve (FCLS) favor those born later, while Uniform (U) rule sequentially distributes each slot equally between all currently interested agents. Looking at traditional assignment rules, Random Priority (RP) rule orders agents randomly and lets them pick their best allocations in turn; Serial rule (S) lets all agents simultaneously acquire shares of their best slots with the same speed.

We show that U, RP, and S rules are efficient, strategy-proof, and non-envious. Thus, the persistent incompatibility between efficiency, fairness and incentive compatibility, prevalent in assignment models, even on restricted domains, disappears.

Interestingly, FCFS (as well as FCLS) rules are neither strategy-proof nor envy-free.

The preference domain we study might look rather simple, but it is both important for practical applications and difficult to analyze.

In particular, a large part of the literature on random assignment over last two decades was devoted to two main rules: Random Priority and Serial. Even under

significant restrictions on preferences, there is a strong trade-off between going for either RP or S rule. S is usually not strategy-proof, while RP fails no-envy and efficiency criteria. This is true, for example, on the single-peaked domain; moreover, strategy-proofness efficiency and fairness are not compatible there (see Kasajima 2013). The "symmetric" to ours domain where all agents are born on day 1, but have different "death dates", proved much easier to investigate (see Bogomolnaia and Moulin 2002); RP is still not efficient.

This conflict disappears on our domain for the first time. Both rules are shown to satisfy all traditional properties in their strong form.

2 Related literature

Study of random assignment model with ordinal input and no transfers proliferated over last two decades. Mostly, it assumes all "slots" and "jobs" to be of the same unit size. While the idea of RP rule is an old one, S rule was introduced by Bogomolnaia and Moulin (2001), who also first noticed the strong tension between traditional requirements of efficiency, fairness, and incentive-compatibility. No rule satisfies all three properties; S rule is not strategy-proof, while RP rule is envious (and inefficient).

Lately, several authors further explored the extent of (im)possibility frontiers on the general domain, weakening the above requirements (see, for example, Nesterov 2017).

A number of works is devoted to the analysis of S rule, and its comparison with PR rule. The two are found to behave similarly in large economies (Che and Kojima 2010). S rule was extended to the domain with indifferences, see Katta Sethuraman (2005). Several axiomatic characterizations of S rule were proposed (RP is more elusive in this respect), as in Hashimoto et al. (2014) Bogomolnaia and Heo (2012) or Bogomolnaia (2015).

Closer to the current work, there is a handful of articles which investigate different domain restrictions.

Schulman and Vazirani (2015) discusses lexicographic preferences over fractional assignments (this is also one of a few works where "slots" and "jobs" can have different, and in particular non-integer, capacities). They show that on this domain S rule becomes strategy-proof, when capacity of all slots is larger than the size of any job.

A few works consider similar to ours "linear" domains, where slots ("objects") are naturally ordered along a line.

When preferences are single-peaked, Kasajima (2013) shows that strategy-proofness, efficiency, and equal treatment of equals are still incompatible. Hougaard et al. (2014) focuses on "aggregate gap minimizing" property on single-peaked domain.[1] It is shown to be incompatible with either strategy-proofness or no-envy.

In Bogomolnaia and Moulin (2002), all (unit) slots are ordered in the same way, and all jobs are born on date one, but they have different (private) deadlines. Here, S

[1] On our domain and for the integer case, this notion is reduced to "aggregate delay minimizing". However, here, contrary to the more general single-peaked domain, this condition is simply equivalent to efficiency.

becomes strategy-proof, but no other rule is once we also require efficiency and mild fairness condition of equal treatment of equals.

A small, but interesting set of papers is devoted to the model where agents' utility is quasi-linear in money, and monetary transfers are allowed. When agents with unit jobs have private per period waiting costs, the well-known efficient and strategy-proof Clarke–Groves mechanism is shown to be budget-balanced on the queueing domain. Both the case of all agents born in period 1 and the case of different birth dates were considered. This study was started in Suijs (1996) and Mitra (2000); see alsoMitra and Sen (2010) and Ghosh et al. (2021).

Traditional literature on the queueing models (usually with random arrival and arbitrary jobs' sizes, very different interpretation of fairness or even efficiency, monetary transfers used, and often no concerns for incentives' issue) is rather far from our subject. The main similarity lies in allowing for randomization (or processing jobs in installments). See for instance (Demers et al. 1989; Friedman and Henderson 2003), or Friedman et al. (2015).

Queuing protocols are sometimes discussed and compared based on fairness properties of a different nature. For example, Moulin and Stong (2002) consider jobs of different size and characterize "fair queueing" discussed in Demers et al. (1989), which is based on the round robin rule, by consistency and invariance properties. Moulin (2007) is concerned with minimizing delays or "slowdown", when jobs have different sizes.

Finally we mention a stream of work on fair cost sharing in queueing/scheduling models where agents arrive at the same time and must pay for service either in cash or delay time. They usually have different job sizes, which drives the analysis as it should result in different contributions to the total cost. See, for example, Moulin and Shenker (1992).

3 The model

Server operates in discrete time, and can process certain amount of jobs per time slot (period, date, etc.). Let $N = (1, ..., k, ..., K)$ be the ordered set of slots (potentially infinite, i.e. it may be $K = \infty$). Each slot $k \in N$ has the processing capacity $z_k \in \mathbb{R}_+$. Server is fully described by the vector $z \in \mathbb{R}_+^{|N|}$, the (potentially infinite) vector of given slots' capacities.

$A = \{a_1, ..., a_n\}$ is the set of n agents. Agents have unit jobs they want to be processed by the server. Their jobs arise at different moments in time, and agents want the job done as soon as it is ready. Thus, for any agent a, there is a "birth slot" $R_a \in N$, the slot on which her job becomes ready. Given R_a, the preferences of a over "integer" allocations, where her job is assigned to one slot only, are uniquely defined as: $R_a \succ R_a + 1 \succ R_a + 2 \succ ... \succ \emptyset \succ [\forall k \in \mathbb{N}, k < R_a]$. Birth dates are private information. A (reported) birth profile is $R = (R_{a_1}, ..., R_{a_n})$.

An instance of the problem is given by $\Gamma = (A, N, R, z)$.

We are interested in a scheduling problem. Agents report their birth dates R_a beforehand.[2] Server then needs to schedule jobs, based on R, respecting feasibility constraints: no one is assigned a slot before her job is born, and server's capacity is never exceeded. Server may (and often is obliged to) use fractional assignments, where an agent's job is processed in installments over several slots. However, no monetary transfers are allowed.

Clearly, agents' preferences over fractional assignments are not fully determined by the birth dates R_a. We assume that an agent a can have any preferences over fractional allocations, consistent with her birth date R_a and the interest to be served earlier. Thus, R_a only provides server with an incomplete information on how she ranks allocations, based on first order stochastic dominance.[3]

In many practical applications, a job can only be processed in one shot. It is naturally can be assumed then, that all capacities z_k are integers.

In this case, no "fractional" assignments in the literal interpretation above are feasible. However, with all $z_k \in \mathbb{Z}_+$, a fractional assignment has a natural alternative interpretation as a "random assignment"–a lottery over "deterministic" assignments (where each job is processed in at most one slot). A variant of Birkoff theorem guarantees that such a representation exists.[4] We refer to such problems with all $z_k \in \mathbb{Z}_+$ as "integer" problems. They constitute an important subclass of our model, and can be treated within it, keeping in mind the randomization interpretation.

We can also take an alternative "online" queueing modeling view. The "online" assumption means no forecast power, and thus implies that the way a slot k is distributed cannot be affected by agents, "born" after the moment k.

It is easy to see that the mechanisms discussed below satisfy this "no forecast" property. This is true for Priority, and hence Random Priority (with arbitrary weights over its deterministic components); for First Came First Served (or FCLS), for Serial rule, for Uniform, and for many others.

A (fractional) allocation is a matrix $P = (p_{ak})_{a\in A, k\in N}$ with n rows, corresponding to agents, and K (potentially infinite number of) columns, corresponding to slots. Here $p_{ak} \in [0, 1]$ is the share of slot k, assigned to the job of agent a.

Note: Given an instance Γ, we can sometimes truncate N with $K = \infty$ to be finite without loss of generality. For example, in the case of an integer problem with all $z_k > 0$ ($z_k \in \mathbb{N}$ then) it is enough to consider a finite number of slots $K = \max\limits_{a\in A} R_a + n$. However in general it may be not possible, as z_k could be rather small.[5]

An allocation is *feasible*, if all following is true:

[2] We can instead assume that server works "online", and only becomes aware of a job once its (announced) birth occurs. All our rules work equally well in both settings. See below.

[3] An agent may even prefer her job to be processed only partially, but earlier, to it being done in full, but later (as long as two allocations are not comparable by first order stochastic dominance). For example, she might prefer any fractional allocation where she gets a positive fraction of her job processed in a given slot $l \geq R_a$ (even if her job is never done in full), to the allocation where her job is fully processed in the slot $l + 1$.

[4] Since all z_k are integer, without loss of generality T can be assumed finite in any given instance.

[5] We can also assume that N always contains slot "∞ " with infinite capacity, where no one is born, and which is the worst option for everyone. Then all agents can be fully satisfied always.

(i) the sum of agents' shares of any given slot does not exceed its capacity: $P_k = \sum_{a \in A} p_{ak} \leq z_k$ for any slot k;
(ii) it never allocates (even partially) to an agent a slot coming before her birth (i.e. a slot she values less than the outside option $\emptyset$): $p_{ak} = 0$ for all (a, k) with $k < R_a$; and
(iii) it does not allocate any agent more then she needs: $\sum_{k \in N} p_{ak} \leq 1$ for all a.[6]

We say that an agent a with $\sum_{k \in N} p_{ak} = 1$ is "fully satisfied" at P.

An allocation is *exact*, if it is feasible and fully satisfies all agents (this may be not possible, even for $K = \infty$, if z_k are small enough).

A rule specifies a feasible allocation for any given instance $\Gamma = (A, N, R, z)$.

An *ETE* ("equal treatment of equals") rule is the one which only depends on numbers $n_k = |\{a : R_a = k\}|$—quantities of agents, born at each slot k, and treats equally agents who report the same birth date.

Our goal is to propose reasonable allocation rules in this setting (both new and adapted from existing ones in similar models), and to evaluate how they fair on normative properties—traditional ones of efficiency, fairness, and incentive compatibility, as well as new ones, specifically applicable to our model. We thus first discuss properties we might be interested in.

Our model is based on ordinal input only: all server receives is the (reported) profile R of birth dates. Hence, it does not know full agents' preferences over options to process a job in several installments. It thus can only rely on partial comparisons of fractional allocations, based on stochastic dominance, when evaluating agents' welfare.

We say that an agent a "ordinally prefers" the vector of slots' shares $P^a = (p_{ak})_k$ to $Q^a = (q_{ak})_k$, and write $P^a \, \mathrm{dom}_a \, Q^a$, if P^a first order stochastically dominates Q^a, i.e., if $\sum_{R_a \leq k \leq i} p_{ak} \geq \sum_{R_a \leq k \leq i} q_{ak}$ for all $i \geq R_a$.[7] She strictly ordinally prefers P^a to Q^a, if at least one of those inequalities is strict, and is indifferent otherwise.

These ordinal preferences, based on stochastic dominance, can be fully derived by the server from the agent's birth date. They are, however, much weaker than underlying preferences. For each agent a, born at R_a, dom_a defines an incomplete (though transitive) relation on the set of fractional allocations, which is a subset of her full (complete, but unknown to the server) preference relation over them. It is assumed that she can have any preferences fully consistent with dom_a. Note also, that if an agent ordinally prefers P^a to Q^a, then all agents born in the same slot would.

In line with the established tradition in the literature on fractional allocations with ordinal reports, these ordinal preferences will be our main tool to evaluate whether different properties are satisfied. It means, in particular, that the properties to be discussed are formulated in the strongest possible form.

[6] We can easily dispense with the requirement (iii) and just assume that an agent does not use the most late slots in case she is assigned the total share of more then 1. Hence, we sometimes skip this requirement for the sake of better notation or easier argument. We can also lift the requirement (ii) without loss of generality.

[7] Note that our agent a does not care about p_{ak}, q_{ak} with $k < R_a$. This is important to keep in mind when we later define properties such as envy-freeness or strategy-proofness.

4 Efficiency structure

We start by looking at efficiency. In general, in fair division problems with no money, the structure of efficient allocations is very difficult to describe or analyze, even for very restricted domains. However on our domain it turns out to be rather simple and tractable.

Given that the server does not know full preferences over options to process a job in several installments, it cannot evaluate whether a fractional allocation is fully (Pareto) efficient. In this setting we can talk about following notions, traditionally used in the literature.

Ex-post efficiency (mainly for integer problems):

An allocation is "ex-post efficient" if it can be represented as a lottery over Pareto efficient deterministic assignments. Note that an allocation can have several such representations, and it is not immediately obvious whether if one representation only uses efficient assignments, then all of them would. In the general assignment model it is not true (see Abdulkadiroglu Sönmez [2005]). However, as we will see below, it is true in our model.

Here we call an allocation P deterministic if any agent a has $p_{ak} > 0$ for at most one k (so it is slightly more general than the 0-1 deterministic allocations traditionally considered in the literature).

Ordinal efficiency:

A feasible allocation P is "ordinally efficient" if there is no other feasible allocation Q, such that all agents ordinally prefer Q to P, i.e. $Q^a \operatorname{dom}_a P^a$ for every $a \in A$ (with at least one strictly preferring it).

In what follows we use the word "efficiency" to mean ordinal efficiency (which, as we will see, coincides with ex-post one, whenever the later is well defined).

The following construction will be useful to characterize the set of efficient allocations. Given an instance $\Gamma = (A, N, R, z)$, it creates a natural partition of the set of slots N into "segments", and "space between". Those segments then can be considered independently when deciding on how to distribute slots to jobs, once we impose the efficiency requirement. We call it $N(\Gamma)$, the "segment partition of N".

Recall that $n_k = |\{a : R_a = k\}|$ is the number of agents/jobs born at slot k.

Step 1. Let $l_1 = \min\{l \in N | \exists a \text{ s.t. } R_a = l\}$, the first slot where some job is born, and $r_1 = \min\{\{r \in N | \sum_{l_1 \le k \le r} n_k \le \sum_{l_1 \le k \le r} z_k\} \cup \{K\}\}$. Thus, r_1 is the earliest slot at which the total number of agents born up to it does not exceed the total capacity of all slots up to r_1; and if there is no such slot then it is the "last" slot K (which may be infinity). Let $I_1 = [l_1, r_1] = (l_1, l_1 + 1, ..., r_1)$, a segment in N.

If either $r_1 = K$ or $R_a \le r_1$ for all $a \in A$ then we stop. Otherwise we go on to create next segment in the same manner. Specifically:

Step m. Let $l_m = \min\{l \in N, l > r_{m-1} | \exists a \text{ s.t. } R_a = l\}$, and let $r_m = \min\{\{r \in N | \sum_{l_m \le k \le r} n_k \le \sum_{l_m \le k \le r} z_k\} \cup \{K\}\}$. Thus, l_m the first slot beyond I_{m-1} where some job is born, and r_m is the earliest slot after that at which the total number of agents born between l_m and r_m (inclusive) does not exceed the total capacity of all slots

between l_m and r_m (inclusive). If there is no such slot r_m, then it is the "last" slot K. Let $I_m = [l_m, r_m]$, a segment in N.

If either $r_m = K$ or $R_a \leq r_m$ for all $a \in A$ then we stop. Otherwise we go on to create next segment.

Since for each segment there is at least one agent born in it, this process will finish after at most n steps.

We will get the (uniquely defined by Γ) partition of N into $I_1, ..., I_M$ and $I_0 = N \setminus \bigcup_{1 \leq m \leq M} I_m$. Here I_0 is the set of "empty" parts of server's timeline, where no one is born, lying in between our segments.

Fix an instance $\Gamma = (A, N, R, z)$. Given an allocation P, we refer to $P_k = \sum_{a \in A} p_{ak}$ as the "server's workload" at the date k under P.

Lemma 1 *The following are equivalent:*

(i) *An allocation P is (ordinally) efficient*

(ii) *For any $k \in N$, if $P_k < z_k$ then for all a with $R_a \leq k$ we have $\sum_{R_a \leq i \leq k} p_{ai} = 1$ (if on the date k the slot capacity is not exhausted, then all agents born up to date k are fully satisfied by this date)*

(iii) *The vector $(P_k)_{k \in N}$ is feasible, and "stochastically dominates" the vector $(S_k)_{k \in N}$ for any feasible allocation S, i.e. $\sum_{1 \leq i \leq k} P_i \geq \sum_{1 \leq i \leq k} S_i$ for all k*

(iv) *< for integer assignments > P is ex-post efficient*

(v) *Given an arbitrary (ordinally) efficient allocation Q, we have $P_k = Q_k$ for any $k \in N$*

(vi) *For the segment partition $I_1, ..., I_M$ of N, all the following is true:*

- *for each I_m with $m < M$, and for $I_M = [l_M, r_M]$ if $r_M < K$, $\sum_{a \in A} p_{ak} = z_k$ for all $k \in [l_m, r_m - 1]$ and all agents born on I_m are fully satisfied within I_m;*
- *if $I_M = [l_M, r_M]$ with $r_M = K < \infty$ then $\sum_{a \in A} p_{ak} = z_k$ for all $k \in [l_M, r_M - 1]$; in addition, $\sum_{a \in A} p_{ar_M} < z_{r_M}$ implies that all agents are fully satisfied;*
- *if $I_M = [l_M, r_M]$ with $r_M = K = \infty$ then $\sum_{a \in A} p_{ak} = z_k$ for all $k \in I_M$*[8]

Proof An easy proof is left to the reader.

The most important for us will be the characterization of the set of efficient allocations, provided by (iii) and (v):

There exists a (unique) feasible vector of server's workloads $E(\Gamma) = (E_k)_{k \in N}$, which stochastically dominates any other. The efficient allocations are exactly the ones with the workload vector $E(\Gamma)$.

In addition, if an allocation is efficient, then in the segment partition of N it assigns each interval I_m only to agents born on it, and each slot on I_m, except probably the last

[8] Here, if $r_M < K$, then all agents are fully satisfied. If, however, $r_M = K < \infty$ and $\sum_{a \in A}; p_{ar_M} = z_{r_M}$, or if $r_M = K = \infty$, then either all agents are fully satisfied ("in the limit" for $K = \infty$), or some of the agents born on $[l_M, r_M]$ are not satisfied.

one, is fully distributed. Hence, if we look for efficient rules only, we can decompose our problem into segments, and deal with each segment separately. Thus, in this case without loss of generality we can assume that the whole N is just one segment (as defined above).

In what follows, we will include the efficiency requirement in the definition of a rule. We will thus only consider feasible and efficient allocations.

Note also, that:

Corollary (i) *Any lottery over efficient allocations is itself efficient.*

(ii) *Given* Γ*, all efficient allocations* P *have the same size* $|P| = \sum_{k\in N} P_k = \sum_{a\in A,k\in N} p_{ak}$*. We refer to it as the "maximal size" at the instance* Γ*.*[9]

(iii) *Let* $\Gamma = (A, N, R, z)$ *and* $\Gamma' = (A, N, R, z')$*, with* $\sum_{k=1}^{i} z_k \geq \sum_{k=1}^{i} z'_k$ *for all* $i = 1, ..., r-1$*,* $\sum_{k=1}^{r} z_k \leq \sum_{k=1}^{r} z'_k$*, and* $z_k = z'_k$ *for* $k > r$*. Then* Γ' *has at least the same maximal size as* Γ*. I.e., if we redistribute server capacities to be still available, but at later date(s), this does not decrease the maximal size.*

5 Other properties and some rules

A rule $f : \Gamma \mapsto P$ specifies a feasible and efficient allocation $f(\Gamma) = P$ for any given instance $\Gamma = (A, N, R, z)$.

Often A, N, and z will be fixed, and we will study how a rule behaves when R, the agents' report, varies. We will hence often omit other parameters, and write $f : R \mapsto P = f(R)$.

We want to know till what extent feasible and efficient rules can satisfy traditional normative concerns for fairness and incentive compatibility.

In terms of fairness, we care about symmetry or ETE, and absence of envy. We might also look at the (resource or population) monotonicity conditions, requiring that any change in the instance Γ affects all agents in the same way.

Since agents' birth dates are private, an appropriate incentive compatibility condition would be strategy-proofness.

The notion of ordinal preferences (as implied by announced birth dates), which we used to define efficiency, is also used in other normative properties of importance. Remember that the server has incomplete information about agents' preferences, and so cannot always evaluate how an agent would compare two allocations. However, all our properties are defined in their strongest variant, where the server can guarantee them based on the birth profile R only, no matter underlying individual preferences over fractional assignments.

Anonymity and Equal Treatment of Equals (ETE)

[9] No allocation can have larger size, but an inefficient one can be of maximal size—for example when all agents are fully served, but later than in an efficient allocation.

An allocation P is ETE, iff it gives identical assignments for agents born on the same date. Thus, $R_a = R_b$ implies $P^a = (p_{ak})_k = P^b = (p_{bk})_k$.

An ETE rule is the one which always returns ETE allocations.

An anonymous rule is the one which does not depend on agents' names. Specifically, assume R' differs from R only in that two agents, a and b, exchange their birth date announcements: $R'_a = R_b$, $R'_b = R_a$, but $R'_c = R_c$ for all $c \neq a, b$. The rule f is anonymous, if this results in exchanging allocations assigned to these two agents, while all other agents get the same allocations as before.

I.e., for any such instances $\Gamma = (A, N, R, z)$ and $\Gamma' = (A, N, R', z)$ we have $f(\Gamma) = P$ and $f(\Gamma') = P'$, with $(P')^a = P^b$, $(P')^b = P^a$, and $(P')^c = P^c$ for all $c \neq a, b$.

No-envy

Agent a does not envy agent b for sure (in the ordinal sense) at the allocation P, if agent a ordinally prefers her assignment $P^a = (p_{ak})_k$ to the assignment $P^b = (p_{bk})_k$ of agent b.

A rule is non-envious, if there is no instance at which some agent for sure envies some other agent.

Resource monotonicity

A rule f is resource-monotonic, iff, for any vectors z and z' of slots capacities with $z'_k \geq z_k$ for all k, we have that all agents ordinally prefer $f(A, N, R, z')$ to $f(A, N, R, z)$.

Population monotonicity

A rule f is population-monotonic, iff, whenever some of the agents are removed but the remaining agents keep the same birth dates, all remaining agents are better off. Specifically, let $B \subset A$, and $R' = R|_B$ (R restricted to B). Then any agent $a \in B$ ordinally prefers $f(B, N, R', z)$ to $f(A, N, R, z)$.

Strategy-proofness

Assume that an agent a is born at R_a, but announces a different birth date R'_a. When comparing two allocations P and Q, this agent ordinally prefers P to Q (under her real birth date R_a) iff $\sum_{R_a \leq k \leq i} p_{ak} \geq \sum_{R_a \leq k \leq i} q_{ak}$ for all $i \geq R_a$.

A rule f is strategy-proof, iff at any instance Γ each agent a ordinally prefers (under R_a) her assignment in $f(\Gamma)$ to her assignment in any $f(A, N, R', z)$, where the vector of birth dates R' only differs from R in that the birth date of this agent a changes from R_a to some other R'_a.

Consistency

A rule f is consistent iff, whenever we remove an agent together with her personal assignment, the remaining agents share the remaining slots capacities in the same way.

Thus, for any $\Gamma = (A, N, R, z)$, and any $a \in A$ who is assigned by f the vector of shares $(p^a_1, ..., p^a_k, ...) = (f(\Gamma))^a$, let $\Gamma' = (A', N, R', z')$ with $A' = A \backslash \{a\}$, $R' = R|_{A'}$, and $z'_k = z_k - p^a_k$ for all $k \in N$. The rule f is consistent iff $\left(f(\Gamma')\right)^b = (f(\Gamma))^b$ for all $b \neq a$.

We now turn to look at possible assignment rules.

Fix some ordering $\pi = (a_1, ..., a_n)$ of the set A of agents. A Priority allocation, corresponding to the ordering π, is the (feasible and efficient) one where agents earlier

in π are fully served before server starts working on jobs of agents later in π. Formally, for any a_i, a_j with $i < j$, and for any slot $r \geq R_{a_i}$, we have that $p_{a_j r} > 0$ implies $\sum_{R_{a_i} \leq k \leq r} p_{a_i k} \geq 1$.[10] A Priority rule ${}^{\pi}f$, based on the fixed ordering π, returns Priority allocation corresponding to the ordering π for each instance.

The famous in the queueing theory "First Come First Serve" (FCFS) rules are those where any agent born earlier is served earlier: for any a, b with $R_a < R_b$ and any slot $r \geq R_a$ we have that $p_{br} > 0$ implies $\sum_{R_a \leq k \leq r} p_{ak} \geq 1$.

A variant of FCFS rule ${}^{\pi}F$, corresponding to the given ordering of agents $\pi = (a_1, ..., a_n)$, obtains when for each reported birth profile R we implement the Priority allocation, based on the ordering of agents $\overline{\pi}(R)$, in which agents with earlier birth dates come earlier, and the ordering of agents with the same birth dates is the same as in π (thus, π is used to "break ties").

The symmetric FCFS rule F is the uniform averaging over all ${}^{\pi}F$ for all possible π. In any allocation $F = F(R)$ we have: $R_a < R_b$ implies $F_{br} = 0$ for all slots $r \geq R_a$ where $\sum_{R_a \leq k \leq r} F_{ak}(R) < 1$, and $R_a = R_b$ implies identical allocations $F^a(R) = F^b(R)$.

Another traditional queueing rules are "First Come Last Serve" (FCLS), where agents born later are given absolute priority as soon as they are born: for any a, b with $R_a > R_b$ and for any slot $r \geq R_a$, we have that $p_{br} > 0$ implies $\sum_{R_a \leq k \leq r} p_{ak} \geq 1$.

Similarly to the above, one can define FCLS rules ${}^{\pi}L$ corresponding to given fixed orderings π, used to break ties between agents born on the same date; as well as the symmetric FCLS rule L.

Note that ${}^{\pi}F$, ${}^{\pi}L$ are not Priority rules (agents are ordered by their birth dates, so the ordering is different for different instances).

FCFS and FCLS rules are efficient, and may be arguably considered as "fair" in some circumstances. In particular, it is easy to see that F, L, and all ${}^{\pi}F$, ${}^{\pi}L$ are resource and population monotone, and that F, L are anonymous. However, they are not incentive compatible and potentially envious.

Proposition 1 *Any FCFS or FCLS rule is neither strategy proof nor envy-free.*

Proof (i) Fix any FCFS rule f, and let $n = 6$, $T \geq 7$, and all $z_k = 1$. Consider the profile R^1 with four agents a_1, a_2, a_3, a_4 born at slot 1, and two agents a_5, a_6 born at slot 2. Agents a_1 to a_4 are fully served in first four slots. Without loss of generality, let a_1 be the one who gets the largest share of slots 2 and 3 (combined). Hence, $f_{a_1 2}(R^1) + f_{a_1 3}(R^1) \geq 1/2$.

Now consider the profile R, with three agents a_2, a_3, a_4 born at slot 1, and three agents a_1, a_5, a_6 born at slot 2. In $f(R)$ agent a_1 does not get anything before slot 4. Hence, at the profile R she might be willing to manipulate by announcing birth date 1, which would result in the reported profile R^1. (Given that she is not

[10] Priority allocations are not deterministic: a job can be processed over several slots, and a slot can process several jobs (including partially). However, each job is processed over some segment of consecutive slots, and those segments do not overlap, except that the end of one such segment can coincide with the beginning of the other.

interested in slot 1, she may get only as little as 1/2 of her job done at all under R^1, but at least half of her job will be done earlier than at R.)
Further, under R, agents a_1, a_5, a_6 may envy agents a_2, a_3, a_4. Indeed, if, say, agent a_1 only cares about at least $2/3$ of her job being processed early enough, by the date 3, then she would envy the one out of a_2, a_3, a_4 who gets the smallest share in slot 1 (useless for agent a_1).

(ii) Fix now any FCLS rule l, and let $n = 3$, $T \geq 4$, and all $z_k = 1$.
Consider the profile R with three agents a, b, c, all born at slot 1. Under the true birth profile announced, at least one of these agents gets not more than $2/3$ of her job done in slots 1, 2. Without loss of generality, $l_{a1}(R) + l_{a2}(R) \leq 2/3$. If this agent a now declares to be born at slot 2, then under the resulting reported profile R' she would get the slot 2 in full: $l_{a1}(R') = 0$, but $l_{a2}(R') = 1$. Thus, her assignment under R' is not dominated by one under R, which violates strategy-proofness.
Further, under R' at least one agent out of b and c is not fully satisfied in slots 1, 2, and so she may envy agent a.

□

Note: ${}^{\pi}F$, ${}^{\pi}L$ are not even weakly strategy-proof (i.e., even if we only care to prevent manipulations which result is ordinal improvement). F, L however are easy to see to be weakly strategy-proof.

Uniform rule U (again, coming from queueing models) distributes in turn each slot equally to all the agents, who are already born but not yet fully satisfied at that slot, subject to no personal allocation exceeding the total share of 1 unit.

We define it recursively. We have $U_{a1} = \min\left\{\frac{z_1}{n_1}, 1\right\}$ for all agents a born in slot 1, and $U_{a1} = 0$ for all agents a born later.

For any slot $r > 1$ and any agent a with $R_a \leq r$ let $d_{ak} = 1 - \sum_{R_a \leq k < r} U_{ak}$, the residual demand of agent a at slot r (in particular, $d_{bk} = 1$ for agents b born at r). Pick the largest $\lambda \in (0, 1]$ such that $\sum_{a: R_a \leq r} \min\{\lambda, d_{ak}\} \leq z_r$ (it obviously exists and is unique, since left hand side is continuous and increasing in λ). Define $U_{ar} = \min\{\lambda, d_{ak}\}$ for all agents a with $R_a \leq r$, and $U_{ar} = 0$ otherwise.

Clearly, U is efficient and anonymous. Moreover, it is easy to see that:

Lemma 2 *Uniform rule is envy-free, resource and population monotone, and strategy-proof. It is also consistent.*

6 Assignment rules: random priority and serial

We now turn to discuss classical well-known rules from the assignment literature: Random Priority (RP) and Serial (S).

On the general domain both of them are (ordinally) efficient, anonymous, and resource and population monotone. However, RP is strategy-proof but envious, while S is envy-free while not strategy-proof. This is true even on many significantly restricted domains, like single-peaked one. Moreover, anonymity, no-envy

and strategy-proofness are generally not compatible. Many other, stronger results are obtained on impossibility of combining (even mild) efficiency, incentive compatibility and some form of fairness, even on restricted domains.

In our model however, this impossibility finally disappears. As we saw, Uniform rule satisfies all the above mentioned properties. Moreover, RP and S do it too. We will show that RP is now non-envious, while S is strategy-proof. However, contrary to Uniform rule, this is far from obvious. First, we formally define RP and S.[11]

Random Priority rule ${}^{RP}f$ is simply a uniform averaging between Priority rules over all possible orderings of agents.

Serial rule ${}^{S}f$ returns "Serial allocation" for each reported profile R.

The Serial allocation $S = S(R)$ is the result of the "serial consumption process", where agents simultaneously grab shares ("eat") from their best still available slots, with the same constant unit speed, over the time period [0, 1]. Consumption process is a matrix function $S[t]$, where $S[0]$ is zero n by K matrix, $S[1] = S$, and $S[t]$ is the matrix with elements $s_{ak}[t]$ representing shares of slot k, consumed by agent a up to time t.

Let $\tau_k \in [0, 1]$ be the first time moment when slot k is fully consumed/distributed in the Serial consumption process $S[t]$. Those τ_k are "switching" moments in the consumption algorithm, corresponding to S: at each moment τ_k, all agents who were consuming slot k just before it, stop eating it and switch to the nearest next along the "line" N slot, which is not fully consumed as yet (note that they all go to the same slot). I.e., they all switch to the slot $l = \min\{r : r > k, \tau_r > \tau_k\}$.

There is clearly a finite number of switching moments on the total [0, 1] consumption period. Between two consecutive switching moments, no agent changes slot she eats from. Note also, that once an agent arrives to a slot k, she stays there until the moment τ_k when it is exhausted.

We will use the convention that N is a horizontal left to right "timeline", and say that in a consumption process agents move "from left to right".

We will also say that at the moment t agent a is "behind" ("strictly behind") agent b, iff at this moment agent a consumes at a slot (strictly) to the left of the slot agent b consumes.

Illustration For illustration purposes, imagine slots located along the road going down a mountain. Think of slot k as a field of crop of size z_k, located at a given horizontal level k on the mountain, with each next level $k + 1$ strictly below level k. For any agent a, her "birth date" R_a is interpreted here as the level on which she lives. Agents can only move down the road ("to the right").

At time $t = 0$ they all go out of their houses on the levels they live, and start consuming crop with the same constant unit speed (each agent a initially eats from the field R_a). Once a group of agents finish consuming a given field, they all go down to the next field which is still not fully consumed (if there is any), and join those who are eating there. The process ends at time $t = 1$ (or when no feasible slots are available anymore for anyone, if it happens before that).

[11] Without loss of generality, we can think that the segment partition of N consists of one segment $I_1 = [1, K]$. Thus, each slot except the last one is "over-demanded".

Example: Let $T = 6$, $z_k = 1$ for all slots k, $|A| = 9$, $R_{a_1} = R_{a_2} = 1$, $R_{a_3} = ... = R_{a_8} = 2$, $R_{a_9} = 4$.

For $t \in [0, \frac{1}{6}]$ all agents consume shares of their birth slots. At $t = \tau_2 = \frac{1}{6}$, slot 2 is exhausted, and six agents $a_3, ..., a_8$ move right, to consume from slot 3. At $t = \tau_3 = \frac{2}{6} = \frac{1}{3}$, slot 3 is exhausted, and six agents $a_3, ..., a_8$ move further right to consume from slot 4, joining agent a_9 there. At this moment, there is only $\frac{2}{3}$ available at slot 4, and it is being now consumed by seven agents, $a_3, ..., a_8$. These seven agents would finish consuming slot 4 at $t = \tau_4 = \frac{1}{3} + \frac{2}{21} = \frac{9}{21} = \frac{3}{7}$. Note that agents a_1, a_2 are eating from slot 1 up to $t = \frac{1}{2} > \frac{3}{7}$, so they would not join slot 4 before it is finished. At $t = \frac{3}{7}$ seven agents $a_3, ..., a_9$ together move to slot 5 and consume from it until $t = \tau_1 = \frac{1}{2}$, when slot 1 is exhausted. At $t = \tau_1 = \frac{1}{2}$, agents a_1, a_2 join the rest of agents at slot 5 (which is not exhausted at that moment). On the time interval $[\frac{3}{7}, \frac{1}{2}]$ each agent $a_3, ..., a_9$ eats $\frac{1}{14}$ of slot 5, so by $t = \frac{1}{2}$ half of it is consumed. Now all nine agents share remaining half of slot 5, and exhaust it at $t = \tau_5 = \frac{1}{2} + \frac{1}{18} = \frac{5}{9}$. Note that each of agents $a_3, ..., a_9$ consumes in total $\frac{1}{14} + \frac{1}{18} = \frac{16}{126} = \frac{8}{63}$ of slot 5, while agents a_1, a_2 consume $\frac{1}{18}$ each. Finally, at $t = \tau_5 = \frac{5}{9}$ all nine agents move to the last slot 6, and fully consume it, getting $\frac{1}{9}$ of it each. At $t = \tau_6 = \frac{5}{9} + \frac{1}{9} = \frac{2}{3}$ slot 6 is exhausted. There are no feasible slots available anymore for any agent, so the process stops.

$$S[\frac{1}{6}] = \begin{pmatrix} a\backslash k & 1 & 2 & 3 & 4 & 5 & 6 \\ a_1, a_2 & 1/6 & 0 & 0 & 0 & 0 & 0 \\ a_3, ..., a_8 & 0 & 1/6 & 0 & 0 & 0 & 0 \\ a_9 & 0 & 0 & 0 & 1/6 & 0 & 0 \end{pmatrix};$$

$$S[\frac{1}{3}] = \begin{pmatrix} a\backslash k & 1 & 2 & 3 & 4 & 5 & 6 \\ a_1, a_2 & 1/3 & 0 & 0 & 0 & 0 & 0 \\ a_3, ..., a_8 & 0 & 1/6 & 1/6 & 0 & 0 & 0 \\ a_9 & 0 & 0 & 0 & 1/3 & 0 & 0 \end{pmatrix};$$

$$S[\frac{3}{7}] = \begin{pmatrix} a\backslash k & 1 & 2 & 3 & 4 & 5 & 6 \\ a_1, a_2 & 3/7 & 0 & 0 & 0 & 0 & 0 \\ a_3, ..., a_8 & 0 & 1/6 & 1/6 & 2/21 & 0 & 0 \\ a_9 & 0 & 0 & 0 & 3/7 & 0 & 0 \end{pmatrix};$$

$$S[\frac{1}{2}] = \begin{pmatrix} a\backslash k & 1 & 2 & 3 & 4 & 5 & 6 \\ a_1, a_2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ a_3, ..., a_8 & 0 & 1/6 & 1/6 & 2/21 & 1/14 & 0 \\ a_9 & 0 & 0 & 0 & 3/7 & 1/14 & 0 \end{pmatrix};$$

$$S[\frac{5}{9}] = \begin{pmatrix} a\backslash k & 1 & 2 & 3 & 4 & 5 & 6 \\ a_1, a_2 & 1/2 & 0 & 0 & 0 & 1/18 & 0 \\ a_3, ..., a_8 & 0 & 1/6 & 1/6 & 2/21 & 8/63 & 0 \\ a_9 & 0 & 0 & 0 & 3/7 & 8/63 & 0 \end{pmatrix};$$

$$S[1] = S[\frac{2}{3}] = \begin{pmatrix} a\backslash k & 1 & 2 & 3 & 4 & 5 & 6 \\ a_1, a_2 & 1/2 & 0 & 0 & 0 & 1/18 & 1/9 \\ a_3, ..., a_8 & 0 & 1/6 & 1/6 & 2/21 & 8/63 & 1/9 \\ a_9 & 0 & 0 & 0 & 3/7 & 8/63 & 1/9 \end{pmatrix};$$

Lemma 3 (i) *In the Serial allocation, $s_{aR_a} + ... + s_{ak} \geq \tau_k$ for any $a \in A$, $k \in N$, $k \geq R_a$. If $s_{ak} > 0$, then $s_{aR_a} + ... + s_{ak} = \tau_k$.*

(ii) *In the Serial consumption process, whenever $R_a < R_b$ agent a is strictly behind agent b until she "joins" b, and after that their consumption is identical.*

Proof (i) Indeed, $s_{aR_a} + ... + s_{ak}$ is the total amount of slots R_a to k, consumed by agent a. She eats slots from left to right, always consuming from the earliest still not exhausted slot. Given that τ_k is the moment when slot k is finished, she is consuming on $[1, ..., k]$ up to τ_k. If $s_{ak} > 0$, then she consumes slot k up to the moment it is eaten away.

(ii) is obvious.

□

We will also need to define more general "eating" procedures in the same spirit.

"Serial allocation with delays" $SD = SD(R, \sigma)$ obtains when all agents always consume their best still available goods, and do it with the same uniform speed, but have different starting times. Each agent a only starts consumption at a prescribed time σ_a (she remains idle on the time interval $[0, \sigma_a]$). $SD(R)$ is thus determined by the vector of delays $\sigma = (\sigma_{a_1}, ..., \sigma_{a_n})$.

Alternatively, fix R, and pick any allocation P. It can be thought of as a result of a (general) "consumption process (defined by P)" $P[t]$, where agents simultaneously eat from slots, with the same constant unit speed, over the time period [0, 1]. Consumption still occurs in decreasing order of preferences (according to R), "left to right"–each agent consumes first from her best slot, then goes to second best, etc. However, now they do not eat each next slot up to the moment it is exhausted. Instead, each agent a only eats the share p_{ak} of each slot k, which is "prescribed" to her by the given allocation P, and then she is forced to move on.

Moreover, let $p^a = (p_i, p_{i+1}, ..., p_k, ...)$ be an allocation of slots for an agent a, born at slot $R_a = i$. We can think of it as a result of an individual consumption process $p^a(t)$, defined for $t \in [0, 1]$. This agent a is assumed to consume slots along the time period [0, 1] with constant unit speed, moving from left to right. And she is entitled exactly to the share p_k of each slot $k \geq i$.

At time t this agent a will be consuming at a slot $r = r(t, p^a)$, such that $\sum_{j=i}^{r-1} p_j \leq t \leq \sum_{j=i}^{r} p_j$.

Let $p^a = (p_i, p_{i+1}, ..., p_k, ...)$ and $q^a = (q_i, q_{i+1}, ..., q_k, ...)$ be two possible individual allocations for an agent a, born at slot $R_a = i$. By the definition she ordinally prefers p^a to q^a (we write it as p^a dom q^a), if and only if $\sum_{j=i}^{k} p_j \geq \sum_{j=i}^{k} q_j$ for all $k \geq i$.

Lemma 4 *(obvious from the definition above)* p^a *dom* q^a*, if and only if* $r(t, p^a) \leq r(t, q^a)$ *for all* $t \in [0, 1]$.
In words, p^a *dom* q^a *iff agent a at* $P(t)$ *is always "behind" her position at* $Q(t)$.

Consider now an instance $\Gamma = (A, N, R, z)$.

We denote by $|S(\Gamma)|$ the total quantity of slots assigned to all agents together under $S(\Gamma)$ (it is the sum of all elements of the $S(\Gamma)$ matrix).

More generally, we denote by $|P(\tau)|$ the sum of all elements of the matrix $P(\tau)$, the partial allocation at time τ in the consumption process $P[t]$, and by $|P(\tau, r)|$ the sum of all elements in first r columns of $P(\tau)$—the total amount of first r slots consumed by the moment τ under the consumption process $P[t]$.

Let $Z[j,k] = \sum_{j \leq i \leq k} z_i$ be the total server capacity over the slots j to k; and let $N[j,k]$ be the set of agents born between slots j and k (inclusive), with $n[j,k] = \sum_{j \leq i \leq k} n_i = |N[j,k]|$ be the total number of agents born between slot j and slot k.

As a preliminary result, we give formulae to find the slot where an agent with given birth date consumes in $S[t]$, and more generally in $SD(\sigma)[t]$, at any given time $t \in [0, 1]$.

Lemma 5 (i) *In the Serial consumption process* $S[t]$ *under birth profile* R

(ia) *The switching moment* τ_k *(the time at which slot k is exhausted) is* $\tau_k = \min_{j \leq k} \frac{Z[j,k]}{n[j,k]}$.
If an agent a gets positive share of k, then $\tau_k = \min_{j \leq R_a} \frac{Z[j,k]}{n[j,k]}$.

(ib) *At any moment t agents a born at* R_a *consume from the slot* $k(a,t) \geq R_a$ *such that*

$$k(a,t) = \min_{k \geq R_a} \left\{ k : \min_{j \leq k} \frac{Z[j,k]}{n[j,k]} \geq t \right\} = \min_{k \geq R_a} \left\{ k : \min_{j \leq R_a} \frac{Z[j,k]}{n[j,k]} \geq t \right\}$$

or, $k(a,t)$ *is the earliest slot k such that* $tn[j,k] \leq Z[j,k]$ *for all* $j \leq k$.
In other words (given (i)), $k(a,t)$ *is the earliest slot* $k \geq R_a$ *with* $\tau_k \geq t$ *(i.e., the earliest one not exhausted before time t).*

(ii) *In the Serial consumption process with delays, under birth profile R and vector* σ *of delays*

(iia) *The switching moment* τ_k *is the smallest* τ *among those for which* $\sum_{a \in N[j,k]} (\tau - \sigma_a) = Z[j,k]$ *for some* $j \leq k$

(iib) *At any moment t agent a born at* R_a *with delay* σ_a *consumes from the slot* $k(a,t) \geq R_a$*, the earliest slot k with* $\tau_k \geq t$*. In other words, it is the earliest slot k such that* $\sum_{a \in N[j,k]} (t - \sigma_a) \leq Z[j,k]$ *for all* $j \leq k$.

Proof (ia) Note that all agents born up to slot k consume at slots $[1, ..., k]$ during period $[0, \tau_k]$ (as long as slot k is still partially available they do not go beyond it).
In particular, for any $j \leq k$, all agents born on $[j, ..., k]$ consume at slots $[j, ..., k]$ during period $[0, \tau_k]$. Hence, $\tau_k n[j,k] \leq Z[j,k]$ for any $j \leq k$.

Let now l be the earliest slot such that agents born at it receive positive share of slot k in the allocation S. Each such agent consumes at slot k at least during time period $[\widetilde{\tau}, \tau_k]$, where $\widetilde{\tau}$ is the latest switching moment before τ_k. Then:

- All agents born on $[l, ..., k]$ consume at slot k during time $[\widetilde{\tau}, \tau_k]$. Indeed, k is < partially > available and feasible for them; but no other slot from $[l, ..., k]$ is available, or agents born at l would be eating there too.
- All agents born on $[1, ..., l-1]$ consume at slots $[1, ..., l-1]$ during the whole time $[0, \tau_k]$. Indeed, they do not consume slots after k, since k is available; if any of them would move to eat a slot from $[l, ..., k-1]$ during this time, then they would remain on this interval up to τ_k, hence there would be a slot on $[l, ..., k-1]$ available up to τ_k and agents born at l would be consuming it instead of k. Hence, during time period $[0, \tau_k]$ slots $[l, ..., k]$ are eaten only by agents from $N[j, k]$.
- All slots $[l, ..., k]$ are exhausted by time τ_k (slots $[l, ..., k-1]$ are finished by $\widetilde{\tau}$, since at that time agents born at l already eat at slot k).
  Given all this, we have $\tau_k n[l, k] = Z[l, k]$. And if an agent a gets positive share of k, then $l \leq R_a$.

(ib) is obvious given (ia); (ii) is proven in the same way as (i).

□

Proposition 2 *Serial rule is resource and population monotonic, and consistent.*

Proof Let $\Gamma = (A, N, R, z)$; $\Gamma' = (A, N, R, z')$, with $z_k = z'_k + \varepsilon$ for some k and $\varepsilon > 0$, and $z_l = z'_l$ for all $l \neq k$; and $\Gamma'' = (A', N, R', z)$, with $A' = A \cup \{e\}$, $R'|_A = R$.

Pick any $a \in A$. By Lemma above, the slot at which a consumes at any time t is $k(a, t) = \min\limits_{k \geq R_a} \left\{ k : \min\limits_{j \leq R_a} \frac{Z[j,k]}{N[j,k]} \geq t \right\}$, which is at Γ later than at Γ', but earlier than at Γ''. This is because, when we move from Γ to Γ' all $Z[j, k]$ (weakly) increase, while when we move Γ to Γ'' all $N[j, k]$ (weakly) increase.

Consistency is obvious.[12] □

Lemma 6 *Let $P = SD(R, \sigma^1)$ and $Q = SD(R, \sigma^2)$ be Serial allocations with delays where only one agent a is delayed: and $\sigma_a^2 > \sigma_a^1 \geq 0$, while all $\sigma_b = 0$ for all $b \neq a$.*

This agent a then ordinally prefers her assignment P^a to her assignment Q^a.

Proof At any moment t, in both $P[t]$ and $Q[t]$, agent a eats at the earliest slot $k(a, t, \sigma)$ (at or after R_a) among those with switching moment at least t.

Fix $k \geq R_a$. In each of $P[t]$ and $Q[t]$, its switching moment $\tau_k(\sigma) = \min\{\tau \in [0, 1] \,|\, \tau(n[j, k] - 1) + (\tau - \sigma_a) = Z[j, k] \text{ for some } j \leq k\}$.

Denote by $\tau(\sigma, t, k, j)$ the unique τ for which $\tau(n[j, k]-1)+(\tau - \sigma_a) = Z[j, k]$. Since $\sigma_a^2 > \sigma_a^1$, we have $\tau(\sigma^2, t, k, j) > \tau(\sigma^1, t, k, j)$ for each j, and hence $\tau_k(\sigma^2) > \tau_k(\sigma^1)$, for all k. Therefore, $k(a, t, \sigma^1) \leq k(a, t, \sigma^2)$, i.e. agent a in $P[t]$ always consumes behind agent a in $Q[t]$. □

[12] Note that Serial rule is obviously consistent even on the general domain with strict preferences over deterministic objects.

Proposition 3 *Serial rule is strategy-proof.*

Proof Let $\Gamma = (A, N, R, z)$, $a \in A$, and R' is such that $R'_b = R_b$ for all $b \neq a$. Assume agent a at true profile R contemplates manipulation by announcing R'_a.

Let $S[t]$ be the Serial consumption process given R, and $P[t]$ be the Serial consumption process given R'. Let $S^a = (s_{a1}, ..., s_{ak}, ...)$ and $P^a = (p_{a1}, ..., p_{ak}, ...)$ be agent a's allocations in S and P.[13]

There are two possibilities.

(1) Suppose $R'_a < R_a$. Define $\Gamma'' = (A, N, R, z')$, where $z'_k = z_k - p_{ak}$ for $k < R_a$ and $z'_k = z_k$ otherwise. Thus, slots' capacities between R'_a and R_a are decreased exactly by the shares assigned to agent a under misreporting. For this Γ'', consider $Q[t]$, the Serial consumption process with delays σ, where $\sigma_a = \sum_{k<R_a} p_{ak}$, while all $\sigma_b = 0$ for all $b \neq a$. It is easy to see that in the final allocation Q agent a will get $Q^a = (0, ..., 0, p_{aR_a}, ..., p_{ak}, ...)$ (and all other agents will get the same assignments as in P).

If her real birth date is R_a then she is indifferent between Q^a and P^a. But by Lemma 6 and Proposition 2, she prefers S^a to Q^a.

(2) Suppose $R'_a > R_a$.

At any moment t, agent a eats at the slot $k^S(a,t)$ in $S[t]$ and at the slot $k^P(a,t)$ in $P[t]$. The $k(a,t)$ is earliest slot after she is born, among those with the switching moment at least t. We need to show that $k^S(a,t) \leq k^P(a,t)$; we only need to consider $k^S(a,t) > R'_a$.

Now, for any $k \geq R'_a$ and $j \leq k$, we have $N^S[j,k] \subset N^P[j,k]$ (if $R_a < j < R'_a$ then in $P[t]$ the set $N^S[j,k]$ would not contain agent a, but $N^P[j,k]$ would).

Hence, $\min_{j\leq k} \frac{Z[j,k]}{n^S[j,k]} \geq \min_{j\leq k} \frac{Z[j,k]}{n^P[j,k]}$, and so

$$\left\{k \geq R'_a : \min_{j\leq k} \frac{Z[j,k]}{n^S[j,k]} \geq t\right\} \supset \left\{k \geq R'_a : \min_{j\leq k} \frac{Z[j,k]}{n^P[j,k]} \geq t\right\}.$$

We then have

$$k^S(a,t) = \min_{k\geq R_a}\left\{k : \min_{j\leq k} \frac{Z[j,k]}{n^S[j,k]} \geq t\right\} = \min_{k\geq R'_a}\left\{k : \min_{j\leq k} \frac{Z[j,k]}{n^S[j,k]} \geq t\right\} \leq$$

$$\leq \min_{k\geq R'_a}\left\{k : \min_{j\leq k} \frac{Z[j,k]}{n^P[j,k]} \geq t\right\} = k^P(a,t). \qquad \square$$

Now we turn to Random Priority rule. It is easy to see that for any ordering of agents the corresponding Priority rule is resource and population monotone, and consistent. Thus, Random Priority rule, being their averaging, is also resource and population monotone. However, it is not consistent.

Proposition 4 *Any Random Priority allocation is envy-free.*

Proof Fix an instance $\Gamma = (A, N, R, z)$, and any two agents a and b. Consider any ordering $\pi : b_1 \succ_\pi ... \succ_\pi b_n$ of the set A of agents, such that $a \succ_\pi b$. Let π' be almost the same ordering as π, except that agents a and b switch positions. We look at Priority allocations corresponding to these orderings: $P = P^\pi$ and $P' = P^{\pi'}$. Let

[13] Feasibility implies $s_{ak} = 0$ for $k < R_a$, and $p_{ak} = 0$ for $k < R'_a$.

$H = \frac{1}{2}P + \frac{1}{2}P'$. It is sufficient to show that in any such allocation H agents a and b do not envy each other. (RP allocation is an averaging of $\frac{n!}{2}$ allocations H, and No Envy property is preserved by linear combinations).

Agents who come before a in π make the same choices under both P and P', and agents who come after b in π do not affect assignments of a or b in both our Priority allocations. Thus, we can assume without loss of generality (simply deleting early agents and subtracting their assignments from z) that agent a is the first and agent b is the last in π. Let $A' = A\backslash\{a, b\}$.

It is enough to show that agent a lexicographically prefers her assignment in P to agent's b assignment in P', and her assignment in P' to agent's b assignment in P (the same fact for agent b is shown in the same way). The first statement is immediate, since in π agent a is the first and gets the best for her feasible assignment in Γ.

Consider the partial allocation $\widetilde{P}$ in P (and $\widetilde{P}'$ in P'), just before the last agent b (or a) picks her assigned slots. Next, reorder these $n-1$ first agents, putting the first agent a (or b) in the last, $(n-1)$-th position, and consider the corresponding Priority allocation S (or S') for those $n-1$ agents. Both those allocations are efficient ones for fist $n-1$ agents in the ordering π (or π'). We know that all efficient allocations use exactly the same total amount of server's capacity at each time slot. Thus, both $\widetilde{P}$ and S leave the same residual vector $\widetilde{z}$ for the remaining agent b to pick time shares from (and both $\widetilde{P}'$ and S' leave the same residual vector $\widetilde{z}'$ for the remaining agent a).

Hence, agent b gets exactly the same assignment in P and in the Priority allocation Q corresponding to the ordering A', a, b; while agent a gets exactly the same assignment in P' and in the Priority allocation Q' corresponding to the ordering A', b, a (here A' is always ordered in the same way—like in π or π').

We are thus left to show that agent a lexicographically prefers her assignment in Q' to the assignment of agent b in Q. Agents in A' make the same choices in Q and Q', and so leave the same residual vector of slot's capacities to agents a and b. Because of that, without loss of generality we can assume that $A' = \emptyset$ and so there exist only agents a and b.

Thus, we only need to show that in any problem with just two agents a and b, agent a prefers her assignment in Priority allocation Q', corresponding to the ordering b, a, to the assignment of agent b in the Priority allocation Q, corresponding to the ordering a, b. Or, in other words, that given any vector z of slots capacities, our agent a prefers the best assignment she can get after b made her pick, to the best assignment she can get after her "clone" (agent born on the same date as a) made her pick. But this is very easy to see:

If $R_b > R_a$ then agent b starts picking slots shares later than the clone of agent a, and so gives to the agent a (lexicographically) better residual vector of capacities.

If $R_b < R_a$ then agent b picks some slots shares before the birth dates of the clone of agent a, and then picks the same slots shares, but finishes earlier; so she also gives to the agent a better residual vector of capacities. □

7 Concluding comments

1. We presented an interesting and important in applications new preference domain for random assignment model, where finally properties of efficiency, strategy-proofness,and envy-freeness become compatible. Both Serial and Random Priority mechanisms turn out to satisfy these axioms, and this is a non-trivial fact.[14] Turning to queueing rules, Uniform also satisfies all three properties, but, surprisingly, FCFS (and FCLS) fail both strategy-proofness and no-envy.

Note that S, RP, and U are different rules, even though they often result in the same allocation.

Example

Let $A = \{a, b, c\}$, $K \geq 5$, and all $z_k = 3/5$. Suppose $R_a = R_b = 1$, $R_c = 2$. It is easy to see that for this instance

$$S = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ a & \frac{3}{10} & \frac{1}{10} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ b & \frac{3}{10} & \frac{1}{10} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ c & 0 & \frac{4}{10} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{array}, \quad RP = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ a & \frac{9}{30} & \frac{5}{30} & \frac{4}{30} & \frac{6}{30} & \frac{6}{30} \\ b & \frac{9}{30} & \frac{5}{30} & \frac{4}{30} & \frac{6}{30} & \frac{6}{30} \\ c & 0 & \frac{8}{30} & \frac{10}{30} & \frac{6}{30} & \frac{6}{30} \end{array},$$

$$U = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ a & \frac{3}{10} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{10} \\ b & \frac{3}{10} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{10} \\ c & 0 & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{4}{10} \end{array}.$$

As an illustration, let us calculate RP_{c2}. In $\frac{1}{3}$ of the 6 orderings c is the first in the ordering and so she gets all $z_2 = 3/5$. In another $\frac{1}{3}$ of the 6 orderings c is the second; so the first in the ordering agent (a or b) gets the whole $z_1 = 3/5$ from slot 1 and then $2/5$ from slot 2. Thus, c gets the remaining $1/5$ from slot 2. Finally, when c is the last she does not get anything from slot 2. We have $RP_{c2} = \frac{1}{3}(3/5 + 1/5 + 0) = \frac{4}{15}$. □

2. One of the concerns in the queueing literature (see also Hougaard et al. (2014) for the random assignment on single-peaked domain) is to minimize "delays". In our model, average delay is the same for all efficient allocations. We can however look at the worst case scenario for the agents who mostly want their jobs to be finalized as soon as possible, and try to minimize the longest possible (total) delay.

Define (total) delay for an agent a, born at R_a and whose job is fully done at slot k, to be $D_a = k - R_a$.

It is known that RP fares as bad as possible with respect to minimizing $\max_{a \in A} D_a$. Surprisingly, S is as bad. Consider $K = n$, all $z_k = 1$, and $R_{a_1} = R_{a_n} = 1$, $R_{a_i} = i$ for all $i \neq 1, n$. Here FCFS minimizes maximal delay (everybody has delay 1). However, both RP and S have maximal delay $n - 1$, since in both rules agents a_1 and a_n get

[14] Note that this is not true on the single-peaked domain, and it is still unknown for the symmetric singe-peaked domain.

positive fraction of slot $K = n$, and so $D_{a_1} = D_{a_n} = n - 1$. (In fact, for this instance RP=S.)

3. All three "good" rules we discuss, S, RP, and U, satisfy "no forecast". Another similar requirement one might think of is "no memory": the exact way the slots before k were distributed does not affect how slot k itself is allocated (though it might affect the demand of agents—in case their jobs were partially processed before k). This one is however less natural and more difficult to define. One way is to assume that at slot k the mechanism can only observe agents' names (this is only if non-anonymous rules are allowed) and and their residual demand.

It is easy to see that Serial[15] and Uniform rules satisfy this property, while FCFS (as well as FCLS) fail it. All Priority rules (which are not symmetric!) have no memory as well. It seems that RP is no memory too.

Moreover, S, and U (and probably RP too), are "stationary", as defined below.

Consider the set of all "one slot distribution functions" $g : [0, 1]^n \times \mathbb{R}_+ \to [0, 1]^n$, $g : (d, z) \longmapsto (s_{a_1}, ..., s_{a_n})$, symmetric with respect to first n variables, which satisfy: $0 \le s_a \le d_a$ for all a; $\sum_{1 \le i \le n} s_{a_i} \le z$; and $\sum_{1 \le i \le n} s_{a_i} < z$ implies all $s_a = 1$. The class Φ of stationary (and ETE and efficient) rules consists of f_g which distribute vector of residual demands at each slot according to the given g.

We have $U, S \in \Phi$. Indeed, U distributes z equally, subject to no one getting more than her d_a. S aims to equalize d_{a_i} as much as possible (by satisfying first the largest demands up to the level of 2nd largest demands, then satisfying agents with those together, etc.).

A natural question would be whether there exist other attractive rules in Φ (in particular, ones which satisfy the properties we discussed above).

References

Abdulkadiroglu, A., Sönmez, T.: Random serial dictatorship and the core from random endowments in house allocation problems. Econometrica **66**, 689–701 (1998)

Abdulkadiroglu, A., Sönmez, T.: Ordinal efficiency and dominated sets. J. Econ. Theory **112**, 157–172 (2003)

Bogomolnaia, A.: Random assignment: redefining the Serial rule. J. Econ. Theory **158**(Part A), 308–318 (2015)

Bogomolnaia, A., Moulin, H.: A new solution to the random assignment problem. J. Econ. Theory **100**(2), 295–328 (2001)

[15] In order to mimic S rule by a no-memory process, server distributes each slot aiming to equalize residual demands as much as possible.

Bogomolnaia, A., Moulin, H.: A simple random assignment problem with a unique solution. Econ. Theory **19**(3), 623–636 (2002). https://doi.org/10.1007/s001990100168
Bogomolnaia, A., Heo, E.J.: Probabilistic assignment of objects: characterizing the serial rule. J. Econ. Theory **147**, 2072–2082 (2012)
Che, Y.-K., Kojima, F.: Asymptotic equivalence of random priority and probabilistic serial mechanisms. Econometrica **78**, 1625–1672 (2010)
Demers, A., Keshav, S., Shenker, S.: Analysis and simulation of a fair queueing algorithm. ACM SIGCOMM Comput. Commun. Rev. **19**(4), 1–12 (1989)
Friedman, E.J., Henderson, S.G.: Fairness and efficiency in web server protocols. ACM SIGMETRICS Perform. Eval. Rev. **31**(1), 229–237 (2003)
Friedman, E., Psomas, C.A., Vardi, S.: Dynamic fair division with minimal disruptions. In: Proceedings of the Sixteenth ACM Conference on Economics and Computation (2015)
Ghosh, S., Long, Y., Mitra, M.: Prior-free online mechanisms for queueing with arrivals. Econ. Theory **72**, 671–700 (2021). https://doi.org/10.1007/s00199-020-01308-7
Hashimoto, T., Hirata, H., Kesten, O., Kurino, M., Ünver, U.M.: Two axiomatic approaches to the probabilistic serial mechanism. Theor. Econ. **9**, 253–277 (2014)
Hougaard, J.L., Moreno-Ternero, J.D., Osterdal, L.P.: Assigning agents to a line. Games Econom. Behav. **87**, 539–553 (2014)
Kasajima, Y.: Probabilistic assignment of indivisible goods with single-peaked preferences. Soc. Choice Welf. **41**, 203–215 (2013)
Katta, A.K., Sethuraman, J.: A solution to the random assignment problem on the full preference domain. J. Econ. Theory **131**, 231–250 (2006)
Mitra, M.: Mechanism design in queueing problems. Econ. Theory **17**(2), 277–305 (2000). https://doi.org/10.1007/PL00004107
Mitra, M., Sen, A.: Efficient allocation of heterogenous commodities with balanced transfers. Soc. Choice Welf. **35**(1), 29–48 (2010)
Moulin, H.: Minimizing the worst slowdown: offline, online. Oper. Res. **55**(5), 876–889 (2007)
Moulin, H., Shenker, S.: Serial cost sharing. Econometrica **60**(5), 1009–1037 (1992)
Moulin, H., Stong, R.: Fair queuing and other probabilistic allocation methods. Math. Oper. Res. **27**, 1 (2002)
Nesterov, A.S.: Fairness and efficiency in strategy-proof object allocation mechanisms. J. Econ. Theory **170**, 145–168 (2017)
Suijs, J.: On incentive compatibility and budget balancedness in public decision making. Econ. Des. **2**(1), 193–209 (1996)
Schulman, L.J., Vazirani, V.V.: Allocation of divisible goods under lexicographic preferences. In: Proceedings of the FSTTCS, pp. 543–559 (2015)