Average-case comparison of random assignment algorithms

Jacky Lo and Mark C. Wilson

Abstract

The problem of *one-sided matching without money* (also known as *house allocation*), namely computing a bijection from a finite set of items to a finite set of agents each of whom has a strict preference order over the items, has been much studied. Symmetry considerations require the use of randomization, yielding the more general notion of *random assignment*. The two most commonly studied algorithms (Random Serial Dictatorship (RP) and Probabilistic Serial Rule (PS)) dominate the literature on random assignments.

One feature of our work is the inclusion of several new algorithms for the problem. We adopt an average-case viewpoint: although these algorithms do not have the axiomatic properties of PS and RP, they are computationally efficient and perform well on random data, at least in the case of sincere preferences. We perform a thorough comparison of the algorithms, using several standard probability distributions on ordinal preferences and measures of fairness, efficiency and social welfare.

We find that there are important differences in performance between the known algorithms. In particular, our lesser-known algorithms yield better overall welfare than PS and RP and better efficiency than RP, with small negative consequences for envy, and are computationally efficient. Thus provided that worst-case and strategic concerns are relatively unimportant, the new algorithms should be seriously considered for use in applications.

1 Introduction

One-sided matching is a fundamental problem from the resource allocation literature. Often called the *house allocation problem* [11], it concerns situations where agents have preferences over items but not vice versa, and each agent must be allocated a unique item (there are variants in which there are too many agents, but we do not consider those in the present article). Although a matching procedure can be an arbitrary mathematical function, we normally consider only those that are algorithmically computable. The one-sided matching problem as stated does not allow monetary transfers or division of items; relaxing each of those conditions leads to a substantial literature. We focus on the simplest case, in which the number n of agents equals the number m of items (the method of analysis used here will extend to the case $n \leq m$ but some axiomatic properties of the algorithms are lost).

Research in this area has mostly focused on axiomatic properties. Various properties such as envy-freeness and ex-post Pareto efficiency have been introduced, and several characterization theorems given using these axioms. While this worst case approach is important, it tends to lead to impossibility results (when too many axioms are imposed) or to several algorithms, each of which satisfies various subsets of the set of axioms, and which are mutually incomparable on the basis of axioms. In order to decide which of these algorithms to choose in a practical situation, it often makes sense to consider measures of performance not based on the worst case. For example, the expected utilitarian welfare for a random preference profile under sincere preferences is an obvious quantity of interest. A further advantage of this approach is that algorithms that are not extreme points in the space of algorithms (in terms of axiomatic worst-case behaviour) and hence had not attracted attention previously, may perform very well on average.

1.1 Our contribution

We study several algorithms for the random assignment problem. In particular we introduce algorithms based on the party game *Yankee Swap* which have not been seriously studied before, and which perform surprisingly well. We discuss specializations of the Boston school assignment mechanism which have been very little studied. We present detailed average-case analysis for the algorithms considered, for three distinct probability distributions on preferences and numerous measures of welfare, efficiency and fairness. The results indicate that serious consideration should be given to the new algorithms at least in settings where strategy and envy are not of major importance. We outline several opportunities for future work.

2 Definitions and terminology

Let $A = \{a_1, \ldots, a_n\}$ be a finite set of *agents* and $O = \{o_1, \ldots, o_n\}$ a finite set of *items*. Let L denote the set of all strict linear orders on O. A preference *profile* is a function $\pi : A \to L$. Given a preference profile as above, a *1-sided matching* or *discrete assignment* is a bijection $f : A \leftrightarrow O$. Let S be the set of all doubly stochastic $n \times n$ matrices. A *random assignment* is a mapping $A \to S$.

A discrete assignment is *Pareto efficient* if no other discrete assignment can award a better item to some agent without awarding a worse one to some other agent.

2.1 Assignment algorithms

All the algorithms under study are *anonymous*, meaning that a permutation of the players leads to the same permutation of the assignment. In other words, only the preferences matter, not the players' identities. This implies in particular that the algorithms are *symmetric*, namely that agents with identical preferences should receive the same assignment. This in turn implies that the algorithms must produce random assignments in general. All algorithms discussed here except Probabilistic Serial achieve this by uniformly choosing a permutation of the agents and applying a deterministic algorithm that produces a discrete assignment. Probabilistic Serial produces a random assignment directly. By a result of Birkhoff and von Neumann, these two approaches are equivalent.

Example 2.1 *(Random Serial Dictatorship)*

Fix an arbitrary ordering on A. The Serial Dictatorship *algorithm with respect to this ordering assigns items to agents as follows: at the ith step, allocate to the ith agent its most preferred item that has not already been allocated to a previous agent. The* Random Serial Dictatorship *(RP) (also called Random Priority) algorithm symmetrizes as described above.*

Example 2.2 *(Probabilistic Serial) The* Probabilistic Serial *rule generates a random assignment as follows. We interpret an assignment of fraction x of item j to agent i to mean that i receives fraction x of j (in other words we pretend that the items are infinitely divisible). All agents simultaneously begin "eating" at unit speed, each agent at each instant eating from its most preferred item. On termination we have a random assignment.*

We now consider some algorithms that have not attracted much attention in the literature.

Example 2.3 *(Yankee Swap) The first class is based on the party game "Yankee Swap" (also known as "White Elephant"). Each of these has a basic outline as follows. For a fixed agent order, play proceeds in rounds. At each iteration, the next player may choose an* *unallocated item, or "steal" its highest preferred available item from another player. Various additional rules on item availability are needed in order to ensure termination. For example, we may require that each player may (temporarily) own each present at most once per round, or that each present may be stolen at most twice per round. As described the Yankee Swap procedure is decentralized. Centralized forms require that players submit their preferences to a central planner, who then simulates the decentralized procedure.*

The next class of algorithms is inspired by the "Boston mechanisms" for the school assignment problem [1], specialised to the case where each school has only a single student place. The algorithms presented here are special cases of those given by Mennle and Seuken [15] who consider the case $m \geq n$.

Example 2.4 *(Boston) The* naive Boston *algorithm proceeds in rounds. Assignments are final — after each item is assigned, that item and agent are no longer available in future rounds. Agents are given a fixed priority order. In the ith round, each remaining item is assigned to the highest priority agent that ranks that item in ith position, if such an agent exists. The* adaptive Boston *algorithm also proceeds in rounds, and at each round each item is assigned to the highest priority agent that ranks that item highest among all remaining items.*

Example 2.5 *(Top Trading Cycle) When each agent is considered to be initially assigned an entire item, the agents may trade amongst themselves as follows. Each agent i points to the agent currently owning the item on the top of i's preference list. By finiteness and since every node has outdegree 1, this directed graph must contain a cycle. Reallocate items according to the arcs in the cycle, and remove these agents and items from further consideration. Repeat (using pointers to the next level preference if necessary) until no items/agents remain. This* TTC algorithm *[16], attributed by Shapley and Scarf to David Gale, always yields a discrete assignment that is Pareto efficient, and the mechanism is strategyproof and individually rational. Furthermore the algorithm runs in polynomial time.*

In our setup we have freedom in the choice of initial assignment. For example, choosing this uniformly at random and running the TTC algorithm is equivalent to RP [2] (a variant of TTC for random assignments yields PS when run on the output of the proportional algorithm [12]). We find it useful to run TTC on the output of our algorithms based on Yankee Swap (algorithms which satisfy ex-post Pareto efficiency gain no benefit from running TTC). The resulting algorithms are ex-post efficient and appear to have considerably better overall performance than the original algorithms.

Example 2.6 *(Benchmarks) There are some basic algorithms we use as benchmarks for the performance of our more interesting algorithms. These are the algorithms that compute the maximum and minimum possible values of the Borda utilitarian welfare (see Section 4.2). Another basic algorithm is that which gives $1/n$ of each item to each agent, which we call the* proportional *algorithm.*

Note that the optimal welfare is achieved when all agents have a different first preference. The lowest possible value for utilitarian welfare occurs when all agents receive their worst item in a situation where they could each exclusively receive their best. The proportional algorithm gives a basic benchmark for algorithm performance. It gives the same output for every input.

2.2 Axioms

When assignments are random, we need to extend preferences over elements of A to preferences over probability distributions on A. The main ways to do this are: expected utility dominance (when agents attach cardinal utilities to items) and stochastic dominance (which requires only ordinal preferences).

A random assignment is *ex-post (Pareto) efficient* if it can be described as a probability distribution over Pareto efficient discrete assignments. Two other commonly used conditions are *SD-efficiency* and *ex-ante efficiency.* A random assignment fails to be SD-efficient if and only if there is some other random assignment in which some agent is better off, and no agents are worse off, in the sense of stochastic dominance (i.e. a Pareto improvement in expectation for all utility functions consistent with the given ordinal preferences). Ex-ante efficiency is not satisfied if and only if a Pareto improvement in expectation is possible for *some* utility functions consistent with the given ordinal preferences. Note that ex-ante efficiency implies SD-efficiency, which implies ex-post efficiency.

A discrete assignment is *envy-free* if each agent prefers her own assignment to that made to any other agent. Envy-freeness is hard to achieve, and can only happen for profiles in which every agent has a different first preference (we term these *anti-unanimous* profiles). For such profiles, an assignment is envy-free if and only if it is efficient. A random assignment is *SD envy-free* if each agent weakly prefers her own assignment to that of any other agent, and *weakly SD envy-free* if for each agent, no other agent has an allocation that i strictly prefers to i's own allocation. Note that the ex-ante versions of these concepts are less interesting.

The axiom of *SD proportionality* states that each agent weakly prefers her own assignment to that given by the proportional algorithm, while *weak SD proportionality* occurs when no agent strictly prefers the proportional allocation to her own. The relation between these concepts and envy-freeness has been clarified by Aziz, Gaspers, Mackenzie and Walsh [5]. In particular SD envy-freeness implies SD-proportionality, but the analogous result is not true for the weak versions.

3 Axiomatic properties of the algorithms

PS is SD-efficient while RP is not [8], and PS is not ex-ante efficient. PS is SD-envy-free; RP is not, but is weakly SD envy-free [8]. RP satisfies SD-proportionality. Both Boston mechanisms are ex-post efficient [15] and neither is SD-efficient [15].

Example 3.1 *(axiomatic failures of Boston algorithms) Consider the case $n = 4$ and a profile for which agents 1 and 2 have preferences $a > b > c > d$ while agents 3 and 4 have preferences $a > c > b > d$ and $b > a > c > d$ respectively. The Naive Boston algorithm yields the random assignment*

$$\begin{bmatrix} 1/3 & 0 & 1/6 & 1/2 \\ 1/3 & 0 & 1/6 & 1/2 \\ 1/3 & 0 & 2/3 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Then agents 1 and 2 are envious of agent 3 because the third row strictly stochastically dominates the first two. Thus the Naive Boston algorithm is not weakly SD envy-free. It is also not SD-proportional: consider a profile in which agents have preferences $a > b > c, a > c > b, b > a > c$ respectively. Under Naive Boston, agent 1 receives assignment $(1/2, 0, 1/2)$ *which does not SD-dominate* $(1/3, 1/3, 1/3)$.

If instead agent 3 had preference the same as agents 1 and 2, then agent 3 would have incentive to switch to $a > c > b > d$. Thus Naive Boston is not strategyproof.

Now consider the case where the first two agents have preference $a > b > c > d$ and the third and fourth agents $a > b > d > c$. Both the Naive and Adaptive Boston algorithms *yield the random assignment*

$$\begin{bmatrix} 1/4 & 1/4 & 5/12 & 1/12 \\ 1/4 & 1/4 & 5/12 & 1/12 \\ 1/4 & 1/4 & 1/12 & 5/12 \\ 1/4 & 1/4 & 1/12 & 5/12 \end{bmatrix}$$

but this is stochastically dominated for all agents by the random assignment

$$\begin{bmatrix} 1/4 & 1/4 & 1/2 & 0 \\ 1/4 & 1/4 & 1/2 & 0 \\ 1/4 & 1/4 & 0 & 1/2 \\ 1/4 & 1/4 & 0 & 1/2 \end{bmatrix}.$$

Thus neither variant is SD-efficient (for the general case $m \geq n$ this is known [15] but it was a priori (barely) conceivable that in the special case $m = n$ each was SD-efficient).

We now show that the Yankee Swap algorithms fail to satisfy any of the basic properties above.

Example 3.2 *(axiomatic failures of Yankee Swap algorithms) Consider the profile where agents 1 and 2 have preferences $a > b > c$ and agent 3 has preference $b > a > c$. Each of the variants of Yankee Swap described above assigns item a to agent 3 with probability 1, and the other two agents are assigned each other item with probability* $1/2$. *This is not an ex post-efficient assignment, because whichever agent receives item b can swap with agent 3, a Pareto improvement. This example also shows that each algorithm is not weakly proportional: in fact from the point of view of agents 1 and 2, the uniform assignment strictly SD-dominates the given assignment.*

If instead all agents had preferences $a > b > c$, then agent 3 would have incentive to switch to $b > a > c$, guaranteeing its second choice. Thus YS is not strategyproof.

We shall always run TTC on the output of a Yankee Swap algorithm, in order to ensure ex-post efficiency.

4 Experimental setup

4.1 Preference distributions

Consider n agents and n items (labelled $\{1, dots, n\}$ for convenience). Let L_n denote the set of strict linear orders on the items. Let P_n denote the set of all profiles. We use three standard probability distributions on profiles: Impartial Culture (IC); Impartial Anonymous Culture (IAC), and Impartial Anonymous Neutral Culture (IANC), all of which have been widely used in the study of voting rules. IC is the uniform distribution on profiles, generated by IID uniform sampling from L_n by each agent. Under IAC, each "anonymous profile" is equally likely — we sample uniformly from equivalence classes corresponding to the action on P_n by the group of permutations of agents. These correspond to Polyà-Eggenberger distributions for values 0 and 1 of the parameter, respectively. IANC was introduced formally more recently [9] but used previously to study voting rules (e.g. [**?**]). In this case we also consider permutations of the items to lead to equivalent situations, and sample uniformly from the resulting equivalence classes.

4.2 Performance measures

We compute expected values, with respect to each of the three probability distributions on P_n listed above, of several random variables. For each axiomatic property, we can use the indicator of that property, or the number of agents that are witness to the property failing. Thus, for example, we report the probability that a randomly chosen profile leads to a random assignment that is SD envy-free, and the expected fraction of ordered pairs of envious agents. Computing the above random variables is accomplished in $\Theta(n^3)$ time by the obvious algorithm.

Efficiency measures are more difficult. The problem of determining whether a random assignment is ex-post efficient is NP-complete [6] even though checking whether a discrete assignment is Pareto efficient is easy (run the TTC algorithm and see whether it changes the assignment). Checking SD-efficiency can be performed by running a cycle detection algorithm such as depth-first search on the digraph described by Bogolmanaia and Moulin [8]. This is an analogue of the digraph in the TTC algorithm. Nodes are agents, and there is an arc from i to j if and only if the random assignment gives nonzero probability to j of i's favourite item.

So far we have dealt with ordinal preferences, which are easier to elicit. However, explicit utility information is helpful in many ways. In order to derive a global welfare measure, we force Borda utilities on the players. In other words, receiving an item ranking ith in ones's list is interpreted as receiving utility of $n + 1 - i$, for $1 \leq i \leq n$. We then compute the expected utility for each agent corresponding to the given random assignment.

When utilities are used, we must still combine them in order to measure social welfare. We consider the *utilitarian* measure (the arithmetic mean of utilities of all agents), the *egalitarian* measure (the minimum utility received by an agent), and the *Nash measure* (the geometric mean of utilities). When using such a global welfare measure μ, we can compare with the values of the function μ_* giving the maximum possible value. We define the *Borda efficiency* of a random assignment to be the fraction $\mu(f)/\mu_*$. Computing μ_* when μ is the Nash or egalitarian measure is computationally hard, so we report only raw values of μ in this case. However computing μ_* is easy for the utilitarian measure: finding an optimal assignment is equivalent to finding a maximum weight matching in a bipartite graph. This is solvable via the *Hungarian algorithm* which runs in time in $\Theta(n^3)$. Thus we report the Borda efficiency in this case. Similarly, we can find an assignment with minimum utilitarian welfare.

4.3 Implementation

We carried out exhaustive experiments for all algorithms and small values of n. In contrast to theoretical worst-case results, we consider average-case analysis with respect to the given distribution on profiles.

For IAC we use the preference distribution representation (the vector whose elements are multiplicities describing the multiset on P_n which represents the profile with order of agents being irrelevant), and enumerated these lexicographically.

We give exhaustive results only for $n = 3, 4$, the cases $n < 3$ being trivial and $n > 4$ computationally infeasible even with all the symmetries employed (except for the IANC distribution where we present results also for $n = 5$). We also did some random sampling for larger values of n, specifically for n from 5 to 50 in increments of 5. For these values we generated 10000 profiles for each value of n under the IC distribution, using the standard Fisher-Yates shuffle algorithm as implemented in Java's Collections framework.

We performed all analysis using programs written in Java. We used a publicly available implementation of the Hungarian algorithm [17] and integer programming software Gurobi [10] for some computations, and our own Java programs (available on request) for the rest.

5 Results

Table 1 gives abbreviations used in the other tables. Table 2–7 shows results for IC, IAC and IANC respectively, for $n = 3$ and $n = 4$. Table 8 presents the case $n = 5$ for IANC. All the above tables give exact results based on exhaustive computation. Table 16 gives sampling results for IC for larger values of n. Note that the reported values for Nash and egalitarian welfare for the optimal utilitarian algorithm ("max") are not reliable for the following reason. The optimum utilitarian welfare over all random assignments always occurs at some a discrete assignment (because of the linearity the objective function). Our Hungarian algorithm finds a specific such optimal discrete assignment. However, the Nash welfare at this assignment will be just one of many possible values that could be reported.

We find clear differences in the Borda welfare performance of the algorithms. In particular Yankee Swap with Top Trading Cycle is better in all cases than the other algorithms (except of course the optimal utilitarian one) under the utilitarian measure and Nash measure, and very competitive under the egalitarian measure. Even for $n = 50$ (under IC) the algorithm achieves over 98% of the optimal utilitarian Borda welfare on average. For egalitarian welfare, RP beats Yankee swap with 3 steals, which beats Yankee Swap, and PS is the best overall in every case.

YS/TTC beats RP and all other algorithms except the SD-efficient PS when we consider the probability of an assignment being SD-efficient.

As far as other fairness criteria go (egalitarian welfare can be considered as a fairness criterion also), Yankee Swap with TTC does create more envy than PS (which is SD envy-free) and RP. The disparity between the fraction of envy-free profiles and the fraction of ordered pairs of agents witnessing envy indicates that very often a small but nonzero number of agents experience envy (in the SD sense). If we weaken this to weak envy-freeness, losses are much smaller, around 2-3% in our exact results. Interestingly, the constrained Yankee Swap with 2 steal limit performs slightly better than the unconstrained version, as it did in the case of egalitarian Borda welfare.

Overall, PS delivers the best results for fairness measures and efficiency, while Yankee Swap is best for (Borda) welfare.

6 Discussion

For situations in which agents are unlikely to act strategically, and envy is not a major issue (for example in artificial multiagent systems applications), the Yankee Swap/TTC algorithm has good computational efficiency and excellent welfare properties, and therefore deserves strong consideration. In particular it outperforms the standard algorithms PS and RP, as well as the Boston algorithms. Theoretical justification for this welfare advantage eludes us at present and this is an interesting topic for future work. Although Yankee Swap/TTC fails to satisfy any of the standard axioms, it deserves further analysis as an algorithm in its own right. For example, how susceptible it is to manipulation in theory and practice is unknown.

Of course, welfare results may be dependent on the utility model assumed. If we use a limiting model (not itself consistent with strict preferences) in which agents care only about their first preference (plurality rather than Borda), the results are different. In this case exhaustive results under IC for $n = 3, 4$ show that the Boston algorithms clearly outperform the others, with PS and RP beating YS. This is not surprising given the first-choice choosing behaviour of the Boston algorithms. We did not compute results in the model in which cardinal utilities are generated randomly from the interval $[0, 1]$ and preference orders inferred from these: experience of the second author in analysis of voting leads us to believe that results would be very similar to those with Borda utilities.

As far as comparison with RP is concerned, our results reiterate the idea that there is a price to pay for strategyproofness [15]. Recent work [14] on hybrid mechanisms (convex combinations of known algorithms for random assignment) shows the benefits to be gained by cooptimizing resistance to manipulation and other desirable properties such as efficiency or envy-freeness. Yankee Swap + TTC is an obvious candidate for hybridisation with RP or other algorithms. Preliminary results (see Tables 7 and 14) show that averaging the output of YS/TTC with RP or PS behaves as one might expect, with welfare and fairness results that are intermediate between those of each individual algorithm. We have not tried to measure susceptibility to strategic manipulation in this paper — this is an obvious topic for future work.

We are aware of very little experimental work on assignment algorithms. Recently, Aziz, Chen, Filos-Ratskias, Mackenzie and Mattei [4] reported experiments with a similar flavour to ours. They generated profiles via the Mallows model with various parameter values and studied the egalitarian welfare for PS and RP (in the case $m \geq n$). Their findings, that PS slightly outperforms RP and that the results are not very sensitive to the underlying distribution, are consistent with ours. Our own results used three probability distributions which, although allowing for correlation in preferences between agents, perhaps cover less ground than [4]. However in view of their results it seems to us likely that the same relationships between algorithms we have found here will persist.

Several authors have discussed worst-case guarantees for algorithms for the assignment problem. Using our preference model and common Borda utilities, Bhagat, Chakrabarty and Khanna [7] proved that the worst case Borda efficiency of RP lies in $[0.526, 2/3]$ while the analogous quantity for PS is $2/3 - o(1)$. Adamczyk, Sankowski and Zhang [3] provide similar analysis for the case where utilities are cardinal and normalised to $[0, 1]$, and provide a lower bound smaller than $1/e$ for the welfare ratio. Our results show that the average case under several preference distributions is very much larger (for several particular small values of n) than these bounds, and very close to the largest possible value, namely 1. This phenomenon of easiness in the average case has become commonplace in computational social choice in recent years, for example in the study of manipulation of voting rules.

For the proportional algorithm, the random assignment is the same for each profile and all agents achieve the same Borda utility $(n+1)/2$. The corresponding values of utilitarian, egalitarian and Nash measures are therefore all $(n+1)/2$. Thus the Borda efficiency of the proportional algorithm is at least $(n+1)/2n$. Our results show that the expected efficiency is not much more than this, implying that the optimal algorithm rather often achieves a Borda welfare close to the maximum possible value n.

It is known [13] that the probability under IC of a profile being such that RP outputs an SD-efficient random assignment is asymptotically zero. We were unable to estimate these probabilities for $n > 4$. Note that already for $n = 5$ under IANC, RP outputs an SD-efficient random assignment less than half the time, suggesting that the convergence will be fast. It seems reasonable to extrapolate that the other algorithms introduced in this paper satisfy the same property, but converge more slowly to zero. Analytic results along these lines will be difficult to obtain, but an interesting challenge.

The numerical results for the new algorithms we have discussed here suggest some axiomatic questions, namely which of the following statements are true for all profiles:

- Naive Boston is weakly SD-proportional.
- Adaptive Boston is SD envy-free.

These are an obvious starting point for future work, and each holds when $n \leq 5$.

References

[1] Atila Abdulkadiroğlu, Parag A. Pathak, Alvin E. Roth, and Tayfun Sönmez. The Boston public school match. *The American Economic Review*, 95(2):368–371, 2005.

[2] Atila Abdulkadiroğlu and Tayfun Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

[3] Marek Adamczyk, Piotr Sankowski, and Qiang Zhang. Efficiency of truthful and symmetric mechanisms in one-sided matching. In *Algorithmic Game Theory*, pages 13–24. Springer Berlin Heidelberg, 2014.

[4] Haris Aziz, Jiashu Chen, Aris Filos-Ratsikas, Simon Mackenzie, and Nicholas Mattei. Egalitarianism of Random Assignment Mechanisms. *arXiv preprint arXiv:1507.06827*, 2015.

[5] Haris Aziz, Serge Gaspers, Simon Mackenzie, and Toby Walsh. Fair assignment of indivisible objects under ordinal preferences. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1305–1312. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[6] Haris Aziz, Simon Mackenzie, Lirong Xia, and Chun Ye. Structure and complexity of ex post efficient random assignments. *CoRR*, abs/1409.6076, 2014.

[7] Anand Bhalgat, Deeparnab Chakrabarty, and Sanjeev Khanna. Social welfare in one-sided matching markets without money. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 87–98. Springer, 2011.

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

[9] Ömer Eğecioğlu and Ayça E. Giritligil. The impartial, anonymous, and neutral culture model: a probability model for sampling public preference structures. *J. Math. Sociol.*, 37(4):203–222, 2013.

[10] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2015.

[11] Aanund Hylland and Richard Zeckhauser. The efficient allocation of individuals to positions. *The Journal of Political Economy*, pages 293–314, 1979.

[12] Onur Kesten. Why do popular mechanisms lack efficiency in random environments? *Journal of Economic Theory*, 144(5):2209–2226, 2009.

[13] Mihai Manea. Asymptotic ordinal inefficiency of random serial dictatorship. *Theoretical Economics*, 4(2):165–197, 2009.

[14] Timo Mennle and Sven Seuken. Hybrid Mechanisms: Trading off Efficiency and Strategyproofness in One-Sided Matching. *ArXiv e-prints*, March 2013.

[15] Timo Mennle and Sven Seuken. The Naive versus the Adaptive Boston Mechanism. *arXiv preprint arXiv:1406.3327*, 2014.

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

[17] Kevin Stern. Hungarianalgorithm.java. `https://github.com/KevinStern/software-and-algorithms/blob/master/src/main/java/blogspot/software_and_algorithms/stern_library/optimization/HungarianAlgorithm.java`

Algorithm	Abbreviation
max	optimal utilitarian Borda
Proportional	Prop
Probabilistic Serial	PS
Random Serial Dictatorship	RP
Adaptive Boston	AB
Naive Boston	NB
Top Trading Cycle	TTC
Yankee Swap	YS
Yankee Swap with TTC	YSG
Yankee Swap with 2 steal limit	YS2
Yankee Swap with 2 steal limit and TTC	YS2G

Table 1: Abbreviations of algorithms used in results

Algo	Util	Egal	Nash	Eff
max	1	2.027778	2.564979	1
prop	0.765212	2	2	0.027778
PS	0.966022	2.430556	2.546195	1
RP	0.967758	2.402778	2.548714	1
AB	0.975198	2.319444	2.55371	1
NB	0.975198	2.319444	2.55371	1
YS	0.955357	2.319444	2.518276	0.75
YSG	0.993056	2.361111	2.604467	1
YS2	0.955357	2.347222	2.520368	0.75
YS2G	0.985119	2.388889	2.590425	1

Table 2: IC3

A Tables of results

A.1 Exact results

Tables of welfare and efficiency results obtained by exact enumeration follow. A title such as "IANC5" means that the preference distribution used was IANC, and $n = 5$. The columns indicate the algorithm used, the expected Borda efficiency, egalitarian and Nash welfare measures, and the probability of SD-efficiency being achieved. The table "IANC5fair" differs in that the columns deal with fairness measures: the probability that an assignment is (weakly) SD envy-free, the fraction of ordered pairs witnessing (weak) SD-envy, and the analogous measures for SD-proportionality.

Note that the maximum possible *a priori* value of the egalitarian and Nash measures is n but this will only be achieved on anti-unanimous profiles. The fraction of profile space consisting of such profiles is small, namely

$$\frac{n!(n-1)!(n-2)(n-2)!\cdots 1!}{(n!)^n} = \frac{n!}{n^n} \sim e^{-n}\sqrt{2\pi n}$$

which approximately equals $22\%, 9\%, 4\%$ in the cases $n = 3, 4, 5$ respectively.

We next present tables of results for fairness criteria.

Algo	Util	Egal	Nash	Eff
max	1	2.679398	3.483575	1
prop	0.705428	2.5	2.5	0.000072
PS	0.954562	3.153272	3.393107	1
RP	0.952864	3.089229	3.384829	0.792101
AB	0.962292	2.883723	3.392333	0.959418
NB	0.963679	2.851605	3.394204	0.959418
YS	0.939804	3.016529	3.342877	0.547743
YSG	0.983932	3.076871	3.486269	0.969184
YS2	0.940554	3.090706	3.348613	0.486545
YS2G	0.972934	3.142729	3.45697	0.924913
RP+PS	0.953713	3.121558	3.38921	0.792101
RP+YSG	0.968398	3.208237	3.442732	0.792101
PS+YSG	0.969247	3.226958	3.446477	0.967448

Table 3: IC4

Algo	Util	Egal	Nash	Eff
max	1	1.875	2.448681	1
prop	0.79932	2	2	0.107143
PS	0.973055	2.357143	2.46239	1
RP	0.974171	2.339286	2.464009	1
AB	0.981505	2.267857	2.469153	1
NB	0.981505	2.267857	2.469153	1
YS	0.955995	2.232143	2.419205	0.785714
YSG	0.995536	2.285714	2.508817	1
YS2	0.955995	2.267857	2.421895	0.785714
YS2G	0.985332	2.321429	2.490762	1

Table 4: IAC3

Algo	Util	Egal	Nash	Eff
max	1	2.614074	3.425454	1
prop	0.716974	2.5	2.5	0.001368
PS	0.957256	3.118993	3.350918	1
RP	0.955587	3.065285	3.34322	0.813675
AB	0.964963	2.873732	3.350404	0.966154
NB	0.966455	2.841254	3.352333	0.966154
YS	0.938964	2.966809	3.288838	0.560684
YSG	0.984336	3.037179	3.434886	0.974359
YS2	0.939813	3.044103	3.295488	0.508034
YS2G	0.972275	3.10339	3.402916	0.928547
RP+PS	0.956422	3.092481	3.347282	0.813675
RP+YSG	0.969962	3.166132	3.396028	0.813675
PS+YSG	0.970796	3.181986	3.399562	0.971624

Table 5: IAC4

Algo	Util	Egal	Nash	Eff
max	1	1.9	2.474968	1
prop	0.790476	2	2	0.1
PS	0.974851	2.4	2.498231	1
RP	0.975893	2.383333	2.499742	1
AB	0.982738	2.316667	2.504543	1
NB	0.982738	2.316667	2.504543	1
YS	0.958929	2.283333	2.457925	0.8
YSG	0.995833	2.333333	2.541563	1
YS2	0.958929	2.316667	2.460435	0.8
YS2G	0.98631	2.366667	2.524711	1

Table 6: IANC3

Algo	Util	Egal	Nash	Eff
max	1	2.551181	3.412235	1
prop	0.717335	2.5	2.5	0.001312
PS	0.958375	3.127151	3.353907	1
RP	0.956138	3.073163	3.344427	0.799213
AB	0.965433	2.887139	3.351865	0.956693
NB	0.966903	2.855479	3.353783	0.956693
YS	0.939112	2.976269	3.288933	0.564304
YSG	0.984518	3.047572	3.434785	0.968504
YS2	0.940116	3.051837	3.295962	0.509186
YS2G	0.972735	3.111658	3.403662	0.917323
(RP+PS)/2	0.957256	3.100485	3.349373	0.799213
(RP+YSG)/2	0.970328	3.172107	3.396462	0.799213
(PS+YSG)/2	0.971446	3.188709	3.400906	0.965879

Table 7: IANC4

Algo	Util	Egal	Nash	Eff
max	1	3.338428	4.407144	1
prop	0.67051	3	3	0.000001
PS	0.950258	3.909104	4.256146	1
RP	0.945863	3.817688	4.23375	0.428321
AB	0.955603	3.489908	4.242348	0.884335
NB	0.958148	3.400126	4.24679	0.891502
YS	0.93235	3.739734	4.177761	0.387172
YSG	0.980143	3.8332	4.379954	0.906478
YS2	0.933121	3.858568	4.185488	0.258149
YS2G	0.966787	3.951573	4.331847	0.710622

Table 8: IANC5

Algo	%EF	#EF	w%EF	w#EF	%Prop	#Prop	w%Prop	w#Prop
max	0.222222	0.819444	0.222222	0.819444	0.222222	0.703704	0.805556	0.935185
prop	1	1	1	1	1	1	1	1
PS	1	1	1	1	1	1	1	1
RP	0.666667	0.944444	1	1	1	1	1	1
AB	0.5	0.888889	1	1	0.5	0.777778	1	1
NB	0.5	0.888889	1	1	0.5	0.777778	1	1
YS	0.416667	0.805556	0.916667	0.972222	0.583333	0.805556	0.916667	0.944444
YSG	0.416667	0.861111	0.833333	0.972222	0.583333	0.833333	1	1
YS2	0.5	0.833333	0.916667	0.944444	0.666667	0.833333	0.916667	0.916667
YS2G	0.583333	0.916667	0.833333	0.972222	0.75	0.916667	1	1

Table 9: IC3fair

Algo	%EF	#EF	w%EF	w#EF	%Prop	#Prop	w%Prop	w#Prop
max	0.09375	0.853769	0.09375	0.853769	0.09375	0.667444	0.947627	0.986907
prop	1	1	1	1	1	1	1	1
PS	1	1	1	1	1	1	1	1
RP	0.282118	0.886285	1	1	1	1	1	1
AB	0.266782	0.869358	1	1	0.541088	0.846354	1	1
NB	0.215567	0.851563	0.862847	0.985677	0.506366	0.842448	1	1
YS	0.189525	0.754774	0.752604	0.96101	0.349248	0.725839	0.967014	0.98235
YSG	0.190394	0.830584	0.652778	0.963469	0.41522	0.819227	0.989583	0.99537
YS2	0.209852	0.780816	0.810619	0.96969	0.469401	0.782697	0.97338	0.990234
YS2G	0.237703	0.85178	0.771412	0.979818	0.557147	0.868056	0.989294	0.997106
RP+PS	0.282118	0.886285	1	1	1	1	1	1
RP+YSG	0.21412	0.797743	0.951389	0.994936	0.718461	0.91088	0.996528	0.998409
PS+YSG	0.253183	0.858724	0.924479	0.992405	0.780093	0.933232	0.996528	0.998409

Table 10: IC4fair

Algo	%EF	#EF	w%EF	w#EF	%Prop	#Prop	w%Prop	w#Prop
max	0.142857	0.767857	0.142857	0.767857	0.142857	0.625	0.732143	0.910714
prop	1	1	1	1	1	1	1	1
PS	1	1	1	1	1	1	1	1
RP	0.785714	0.964286	1	1	1	1	1	1
AB	0.571429	0.892857	1	1	0.571429	0.785714	1	1
NB	0.571429	0.892857	1	1	0.571429	0.785714	1	1
YS	0.464286	0.803571	0.892857	0.964286	0.571429	0.785714	0.892857	0.928571
YSG	0.464286	0.857143	0.892857	0.982143	0.571429	0.821429	1	1
YS2	0.571429	0.839286	0.892857	0.928571	0.678571	0.821429	0.892857	0.892857
YS2G	0.678571	0.928571	0.892857	0.982143	0.785714	0.928571	1	1

Table 11: IAC3fair

Algo	%EF	#EF	w%EF	w#EF	%Prop	#Prop	w%Prop	w#Prop
max	0.073846	0.836125	0.073846	0.836125	0.073846	0.628689	0.936467	0.984117
prop	1	1	1	1	1	1	1	1
PS	1	1	1	1	1	1	1	1
RP	0.342222	0.89641	1	1	1	1	1	1
AB	0.286154	0.868718	1	1	0.551453	0.832479	1	1
NB	0.236923	0.849573	0.871453	0.985641	0.513162	0.828718	1	1
YS	0.195897	0.748376	0.737436	0.956125	0.336752	0.709402	0.948718	0.970598
YSG	0.197949	0.824843	0.682735	0.965812	0.413333	0.811624	0.981538	0.991453
YS2	0.228376	0.778234	0.807521	0.96547	0.467692	0.773675	0.961026	0.98359
YS2G	0.268376	0.851738	0.796923	0.981311	0.564444	0.86735	0.985641	0.996239
RP+PS	0.342222	0.89641	1	1	1	1	1	1
RP+YSG	0.233504	0.801368	0.952137	0.994302	0.69641	0.899145	0.992479	0.996239
PS+YSG	0.275214	0.856182	0.92547	0.992023	0.749744	0.918974	0.992479	0.996239

Table 12: IAC4fair

Algo	%EF	#EF	w%EF	w#EF	%Prop	#Prop	w%Prop	w#Prop
max	0.2	0.783333	0.2	0.783333	0.2	0.666667	0.7	0.9
prop	1	1	1	1	1	1	1	1
PS	1	1	1	1	1	1	1	1
RP	0.8	0.966667	1	1	1	1	1	1
AB	0.6	0.9	1	1	0.6	0.8	1	1
NB	0.6	0.9	1	1	0.6	0.8	1	1
YS	0.5	0.816667	0.9	0.966667	0.6	0.8	0.9	0.933333
YSG	0.5	0.866667	0.9	0.983333	0.6	0.833333	1	1
YS2	0.6	0.85	0.9	0.933333	0.7	0.833333	0.9	0.9
YS2G	0.7	0.933333	0.9	0.983333	0.8	0.933333	1	1

Table 13: IANC3fair

Algo	%EF	#EF	w%EF	w#EF	%Prop	#Prop	w%Prop	w#Prop
max	0.07874	0.835739	0.07874	0.835739	0.07874	0.644685	0.906824	0.976706
prop	1	1	1	1	1	1	1	1
PS	1	1	1	1	1	1	1	1
RP	0.356955	0.898622	1	1	1	1	1	1
AB	0.305774	0.871829	1	1	0.565617	0.83727	1	1
NB	0.254593	0.852581	0.875328	0.986002	0.527559	0.833005	1	1
YS	0.212598	0.750328	0.740157	0.954724	0.349081	0.711942	0.947507	0.96916
YSG	0.216535	0.828412	0.690289	0.966207	0.430446	0.815945	0.981627	0.99147
YS2	0.244094	0.779418	0.805774	0.963473	0.476378	0.773622	0.959318	0.981955
YS2G	0.286089	0.854878	0.80315	0.981737	0.576115	0.869751	0.985564	0.996063
(RP+PS)/2	0.356955	0.898622	1	1	1	1	1	1
(RP+YSG)/2	0.250656	0.805337	0.951444	0.994094	0.7021	0.899934	0.992126	0.996063
(PS+YSG)/2	0.292651	0.859361	0.925197	0.991798	0.757218	0.920932	0.992126	0.996063

Table 14: IANC4fair

Algo	%EF	#EF	w%EF	w#EF	%Prop	#Prop	w%Prop	w#Prop
max	0.035368	0.872021	0.035368	0.872021	0.035368	0.639326	0.982507	0.996501
prop	1	1	1	1	1	1	1	1
PS	1	1	1	1	1	1	1	1
RP	0.09188	0.839838	1	1	1	1	1	1
AB	0.079572	0.843468	1	1	0.28183	0.776017	1	1
NB	0.07738	0.828047	0.730876	0.982336	0.252679	0.77765	1	1
YS	0.079396	0.732629	0.71494	0.974659	0.214598	0.690822	0.985415	0.99534
YSG	0.079592	0.818195	0.620468	0.97517	0.288428	0.806849	0.997751	0.999378
YS2	0.086319	0.75577	0.846747	0.988231	0.404032	0.806293	0.995173	0.998956
YS2G	0.093631	0.810675	0.812399	0.990019	0.515033	0.8834	0.999651	0.99993

Table 15: IANC5fair

A.2 Sampling results

Computational resources required us to compute only the utilitarian Borda welfare. For each size we generated 10000 samples from the IC distribution. All algorithms except PS arise as symmetrisations of deterministic algorithms based on a fixed priority order of agents, so their output cannot be computed efficiently for large n. Awe omit any results essentially involving the random assignment and not a realisation of it, such as whether it is SD-efficient.

Algo	size 10	size 15	size 20	size 25	size 30	size 35	size 40	size 45	size 50
max	1	1	1	1	1	1	1	1	1
prop	0.587938	0.559432	0.54491	0.536123	0.530123	0.525866	0.522668	0.520204	0.518157
PS	0.952354	0.959046	0.964443	0.968452	0.971761	0.974331	0.976622	0.978373	0.979948
RP	0.937452	0.94096	0.9457	0.949558	0.953082	0.956125	0.958885	0.961365	0.96338
AB	0.945345	0.947892	0.951003	0.954211	0.95722	0.960145	0.962257	0.964374	0.966178
NB	0.949237	0.951489	0.954223	0.957468	0.960098	0.962631	0.96464	0.966588	0.968174
YS	0.930863	0.937091	0.943781	0.948886	0.953072	0.957043	0.960185	0.963094	0.965367
YSG	0.976724	0.978726	0.98113	0.982761	0.984343	0.985649	0.98673	0.987679	0.988545
YS2	0.928983	0.934866	0.940909	0.94501	0.94964	0.952918	0.956128	0.958505	0.960834
YS2G	0.959705	0.961158	0.964629	0.967146	0.969676	0.971714	0.9737	0.975211	0.976653

Table 16: Borda utilitarian efficiency for larger n