# Deploying Fair and Efficient Course Allocation Mechanisms


Paula Navarrete

Cyrus Cousins

Fair & Explainable Decision-Making (FED) Lab

UMassAmherst | College of Information & Computer Sciences

Yair Zick

George Bissias

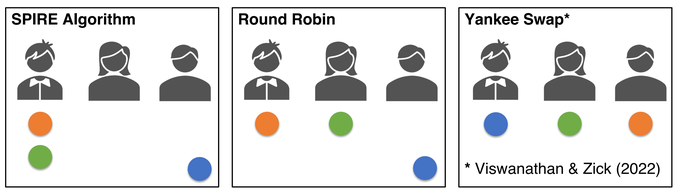

## Motivation

Consider the large-scale allocation problem of assigning seats to students in a large public university. We need practical algorithms that are easy to deploy and ensure

- Efficiency
- A clean allocation
- Fairness


### Allocation Algorithms

Students with binary submodular preferences:


## Our Contribution

### Yankee Swap with Item Multiplicity

We propose a modified version of the Yankee Swap algorithm that allows multiplicity of items by carefully updating the allocation and the exchange graph after every iteration.

```
Set N of n agents, m items
All items initially unassigned, in A_0;
Let U ← N
while U ≠ ∅ do:
    Pick lowest utility agent i ∈ U
if there is some transfer path from i to A_0:
    update allocation and exchange graph
else
    Remove i from U
return (A_1, ..., A_n)
```

**Original algorithm**

$\mathcal{O}\big((n+q)q^2(n+\tau)\big)$

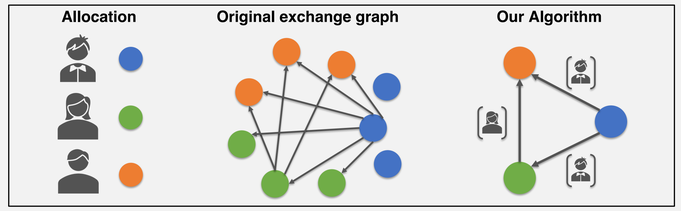
**Our algorithm**

$\mathcal{O}\big((n+q)(\ln n + m^2 + p\gamma c_{max}(q_{max}+\tau)\big)$

$q$: number of copies
$\tau$: valuation function
$p$: maximum path length
$\gamma$: number of items
$q_{max}$: Course capacity
$c_{max}$: Student capacity

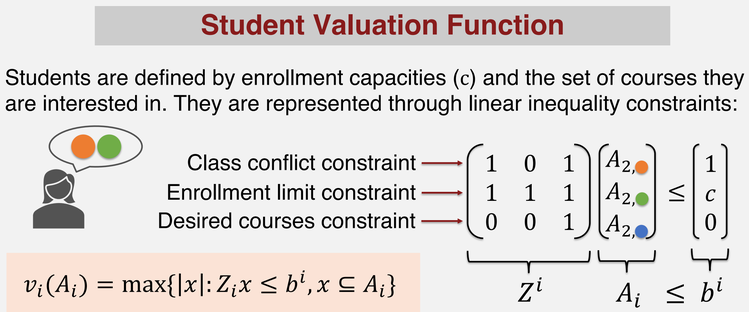
### Exchange Graph

The original Yankee Swap algorithm considers an inefficient representation of the exchange graph when items have multiple copies:




### Student Valuation Function

Students are defined by enrollment capacities (c) and the set of courses they are interested in. They are represented through linear inequality constraints:




$$v_i(A_i) = \max\{|x| : Z_i x \le b^i, x \subseteq A_i\}$$

$$\underbrace{\begin{pmatrix}1&0&1\\1&1&1\\0&0&1\end{pmatrix}}_{Z^i}\underbrace{\begin{pmatrix}A_{2,\bullet}\\A_{2,\bullet}\\A_{2,\bullet}\end{pmatrix}}_{A_i} \le \underbrace{\begin{pmatrix}1\\c\\0\end{pmatrix}}_{b^i}$$

### Implementation

We built a framework that, from a simple course schedule, generates random students, and implemented four allocation algorithms:

- SPIRE Algorithm
- Round Robin
- Yankee Swap
- Integer Linear Program

**Maximize** $\sum_{i \in N} v_i(A_i)$

**Subject to** $\begin{pmatrix}Z^1 & & \\ & \ddots & \\ & & Z^n\end{pmatrix}\begin{pmatrix}A_1\\ \vdots \\ A_n\end{pmatrix} \le \begin{pmatrix}b_1\\ \vdots \\ b_n\end{pmatrix}$

## Experiments

### Setup

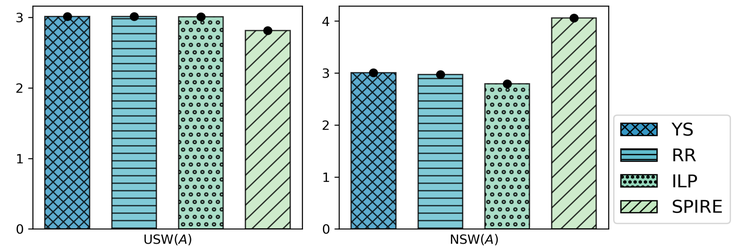
- Courses: Fall 2024 UMass Amherst Computer Science course schedule
  - 98 courses with time slot, day(s) of week and section
- Students: Randomly generated considering the following

| | Undergraduate | MS | PhD |
|---|---|---|---|
| Number of Students | 1,693 | 613 | 148 |
| Enrollment Capacity | 6 | 4 | 4 |
| Min number of liked courses | 1 | 1 | 1 |
| Max number of liked courses | 20 | 15 | 10 |
| Preferred Categories | {UGRAD, 500} | {500, 600} | {500, 600} |

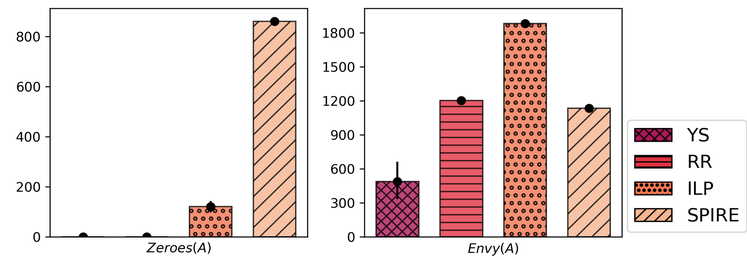
### Results

We generated 100 instances of randomly sampled students, ran the four allocation algorithms and assessed their performance:

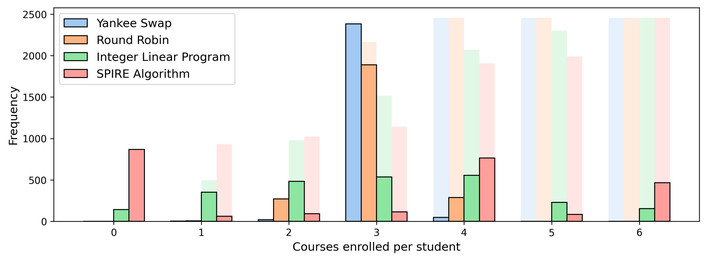
**Maximize:** USW($A$): Sum of all students' utilities (total allocated seats)
NSW($A$): Product of all student's utilities with non-empty bundles




**Minimize:** Zeroes($A$): Number of students with empty bundles
Envy($A$): Number of students who prefer another agent's bundle




## Discussion




### Open Questions and Future Work

- What if we conduct experiments with more competition among students?
- Model realistic student preferences based on:
  - Past data
  - Fall 2024 preference survey data
- Teaching Assistants (TAs) allocation