Distribution of courses without envy

Summary report

Lecturer: Erel David Segal-Halevi

Presenters:

Ofir Shitrit 324249150

Erga Bar Ilan 207829813

Renana Turgeman 322998287

Article summary

Introduction

The article focuses on finding a fair way to distribute resources, such as courses or jobs, among a group of people. It presents a solution called A-CEEI that tries to find an equilibrium where everyone is satisfied with their allocation.
This problem is very common among universities, it helps to create equality between students so that there are no students who are dropped for technical reasons for example.
A previous solution to the problem is CEEI, the problem with this algorithm is that it will not always work in cases where there are complex preferences. For this reason, Budish developed the A-CEEI algorithm, which is essentially an approximate CEEI for the problem, and in which weight equality always holds.

Definitions

A-CEEI, Approximate Competitive Equilibrium from Equal Incomes

Is an equilibrium-based solution for the fair distribution of discrete items to agents with combinatorial demands.
Briefly, this term refers to an algorithm or method used to distribute items among people when each has certain requirements of items, and the goal is to achieve a fair and proper distribution in terms of representation and equality.

EF-TB with respect to budget b_0

Envy-Free-but-for-Tie-Breaking
is a concept used in problems of fair distribution or allocation. It is based on the idea of non-envy, which means that no participant in a division or allocation envies what another participant received.
An allocation is EF-TB relative to the budget b_0 if no student with a higher initial budget envies a student with a lower initial budget in the final allocation. This criterion ensures that

students with a higher initial budget will receive at least good allocations than those received by students with a lower initial budget.
for each student $i, j \in [n]$ such that $b_{0,i} > b_{0,j}$ and we have $u_i(a_i(u, p, b)) > u_i(S)$ for each bundle $S \subseteq a_j(u, p, b)$.

Contested EF-TB with respect to b_0 and p

Reinforcement of the EF-TB definition, in that we now also refer to courses whose price is 0.

SP-L

Strategyproof-in-the-Large
In mechanism design, the goal is to design rules or mechanisms for allocating resources, goods, or making collective decisions in a way that encourages participants to reveal their true preferences or strategies honestly, rather than trying to manipulate the system in their favor.
A "strategy-proof" mechanism is a mechanism where it is best for each player to act according to his true preferences, and not try to manipulate the mechanism.
SP-L takes this idea a step further by examining large-scale settings. It refers to the question of whether a mechanism that is not strategy-proof becomes strategy-proof as the number of participants or the scope of the problem becomes very large. In other words, SP-L examines whether a mechanism that is not strategy-proof in small-scale scenarios becomes strategy-proof when applied to larger and more complex situations.

Randomized mechanism

A random mechanism for assigning courses M described by the function
$f_M: (u, c, r) \mapsto (a_i)_{(i=1)}^{n}$ which receives as input the course assignments (u, c) and random r
and outputs as an assignment output $(a_i)_{(i=1)}^{n} = (M_i(u, c, r))_{(i=1)}^{n}$ so $M_i(u, c, r) \subseteq [m]$
denotes the courses that the mechanism assigned to student i.

Profitable manipulation

For a random mechanism M, a preference function u_i of student i and some manipulation u_i' we say that the manipulation from u_i to u_i' is profitable.

- under resampled randomness (u_{-i}, c, R):

  profits under the market $([u_i, u_{-i}], c)$ and random r so that $r \sim R$ that is $E_{r\sim R}[u_i(M_i([u_i', u_{-i}], c, r))] > E_{r\sim R}[u_i(M_i([u_i, u_{-i}], c, r))]$ which means that the expected value of the function u_i will be greater after we made the change in the weight of one course compared to the original function before the change.
- under resampled population (U_{-i}, c, R):

  profits under the market $([u_i, u_{-i}], c)$ and random r so that $r \sim R$ and $u_{-i} \sim U_{-i}$ that is $E_{u_{-i}\sim U_{-i}, r\sim R}[u_i(M_i([u_i', u_{-i}], c, r))] > E_{u_{-i}\sim U_{-i}, r\sim R}[u_i(M_i([u_i, u_{-i}], c, r))]$.

  Given a known market (u, c), for each student we let U_{-i} be the distribution that resamples $n - 1$ other students, independently and with replacement, from the true population u_{-i} of other students.

Course allocation market

The course allocation market is a system in which students have preferences over courses and courses have a limit on the amount of students they contain, and the goal is to allocate the courses to students in a way that maximizes the students' total benefit, subject to the capacity limits of the courses.

$$a_i(u, p, b) = arg\,max_{x\in 2^{[m]}, p\cdot x\le b_i} u_i(x)$$

Excess demand function

Referring to the difference between the number of students who want to enroll in a course and the number of available places in that course.

$$z_j(u, c, p, b) = \sum_{i=1}^{n} a_{ij}(u, p, b) - c_j$$

Clipped excess demand function

Is a modified version of the excess demand function that takes into account the possibility of negative prices. It is defined as the maximum of zero and the excess demand function. When the price of the course is 0 we round the negative demand function to be 0.

$$\tilde{z}(u,c,p,b) = \begin{cases} z_j(u,c,p,b) & if\ p_j > 0, \\ \max\{0, z_j(u,c,p,b)\} & if\ p_j = 0. \end{cases}$$

Market-clearing error

Market-clearing refers to the price at which the supply of seats (that is, how many seats there are) equals the demand for seats in the course allocation market. In other words, it is the price at which the excess demand function equals zero. At the market-clearing price , all available places in courses are assigned to students who wish to enroll in them, and there are no more students who want to enroll in courses than available places.

$$||\bar{z}(u,c,p,b)||_2$$

(α, β)-CEEI

Refers to a mechanism for assigning courses that aims to find an approximate competitive equilibrium from equal revenues with a small market-clearing error and a small disturbance in the budget. In this mechanism, each student is given an equal budget, and course prices are iteratively adjusted until the market-clearing price is reached.
The (α, β)-CEEI mechanism is designed to provide two important fairness properties: non-envy up to budget (EF-UB) and proportionality up to one product (P1). EF-UB says that no student will prefer the package of courses assigned to another student, given their budget. P1 means that each student receives a bundle of courses at least as valuable as any other bundle they could get by giving up one course. The (α, β)-CEEI mechanism is an improvement over previous mechanisms because it achieves small market clearing error and small budget disturbance while satisfying important fairness properties.
is a variant of the A-CEEI mechanism.

Budgets

In the context of the (α, β)-CEEI mechanism for assigning courses, budgets are used to represent the amount of money each student should spend on courses.

Utility functions

Utility functions are mathematical functions that assign a numerical value to each possible outcome of a decision or choice. In the context of A-CEEI, each student has a utility function that describes his preferences over packages of courses. The utility function takes as input a bundle of courses and outputs a numerical value representing the satisfaction or happiness of the student from that bundle.

The utility function is used by the A-CEEI algorithm to assign courses to students in a way that maximizes overall student satisfaction.

Each auxiliary function u is described by a tuple $(w \in R^m, valid, req: 2^{[m]} \to N)$, where:

$\boldsymbol{w}$: is a vector of weights representing the importance of each course to the student.

$valid$: is a binary function that indicates whether a course package is valid or not (that is, whether it meets the student's hard constraints, such as scheduling and curriculum conflicts).

req**:** is a binary function indicating whether a course bundle is required or not (i.e. whether it meets the student's soft constraints, such as preferences over electives, number of electives required).

For each possible package $\boldsymbol{x} \in 2^{[m]}$, the utility function $u(\boldsymbol{x})$ is defined as follows:

- If x satisfies the validity and demand constraints $(valid\ (\boldsymbol{x}) = 1,\ req(\boldsymbol{x}) = 1)$, then $u(x) = w \cdot x + B$.

  B The level of joy of a student that he gets all the courses he needs. A number that is greater than the sum of all values of a student in all courses. We will choose to represent it as the sum of all the values in the vector $w + 1$.

- If x satisfies only the validity constraint $(valid\ (\boldsymbol{x}) = 1,\ req(\boldsymbol{x}) = 0)$, then $u(x) = w \cdot x$

- If x does not satisfy the validity constraint $(valid\ (\boldsymbol{x}) = 0)$, then $u(x) =$ -∞

$$u(x) = \begin{cases} w \cdot x + b & valid(x) = 1, req(x) = 1 \\ w \cdot x & valid(x) = 1, req(x) = 0 \\ -\infty & valid(x) = 0 \end{cases}$$

Neighborhood function N(p)

The neighborhood function is a function used in the Tabu search algorithm (Algorithm 3) to generate a set of candidate solutions that are close to the current solution. In other words, it defines a set of neighboring solutions that can be explored in search of an optimal solution
The neighborhood function $N(p)$ consists of two types of neighbors: G gradient neighbors and individual price matching neighbors. Gradient neighbors adjust prices relative to excess demand, while single price adjustment neighbors adjust the price vector in a small number of entries only.
In the neighbors of Grandiants, they change the price according to the priority of demand - add to the fixed price a delta twice the excess demand,
In individual neighbors, we increase the price until the demand decreases (there is excess demand and we want to lower it) by 1, we increase the price until there is a student for whom the price is too high and he will already give up, if there is a lack of demand, we either lower the price to 0 or decrease it by a certain constant

Binary relation ~p for prices

The binary relation $\sim_p$ to prices is a relation used in the Tabu search algorithm (Algorithm 3) to compare prices and determine which prices are equal to it in the sense that they have the same excess demand so that it is enough to check a part.
If p and q are two price vectors, then $p \sim q$ means that p is just like q in terms of clearing error. They have the same demand. The binary relation $\sim_p$ is used in the tabu search algorithm to determine which neighboring solutions to explore in the search for an optimal solution.

The Algorithms

Algorithm 1 – Find an A-CEEI with (contested) EF-TB property

The purpose of the algorithm: to find a distribution of courses for students that meets the contested EF-TB.

Input: preference functions $\boldsymbol{u}$, capacity of the courses $\boldsymbol{c}$, initial budget b_0.

Output: final prices p* and final budgets b*.

Parameters: step size δ, budget disturbance ε, type of constraint of the EF-TB denoted t. (0-no constraint, 1- constraint of EF-TB 2- constraint of contested EF-TB)

The algorithm:

1) **Let's initialize the price vector p to be 0.**

2) **budget disturbance ε:**

We will divide the segment $[b_0 \pm \varepsilon]$ into equal segments. We can divide at most into $k_i \leq \frac{2\varepsilon}{\delta} + 1$ segments.

Then for each student i and for each segment l we calculate a_{il} - that is, the group of courses that maximizes the benefit of student i but that meets the current budget b_l.

Next, we will use a linear program to find the optimal budget disturbance that will cause the excess demand function $||\bar{z}(u,c,p,b)||_1$ to be minimal.

The calculation:

Calculation of $\bar{z}(u,c,p,b)$:

For each course j calculate the excess demand function:

$$z_j(u,c,p,b) = \sum_{i=1}^{n} a_{ij}(u,p,b) - c_j$$

For each student i we consider the set of courses a_i that maximizes his utility while maintaining his budget: $a_i(u,p,b) = arg\,max_{x\in 2^{[m]}, p\cdot x\leq b_i} u_i(x)$

Then we calculate the sum $\sum_{i=1}^{n} a_{ij}(u,p,b)$ which represents the number of students who want the j course.

After that, we will subtract the above amount from the capacity of course j.
We will do this for each course j and then the vector z will be constructed as follows:

$z = (z_j)_{j=1}^{m}$ so that z_j is the subtraction between the demand and the supply.

Finally, we will update z to be $\bar{z}$ according to the formula:

$$\tilde{z}(u,c,p,b) = \begin{cases} z_j(u,c,p,b) & if\ p_j > 0, \\ \max\{0, z_j(u,c,p,b)\} & if\ p_j = 0. \end{cases}$$

Calculation of $\|\tilde{z}(u,c,p,b)\|_2$:

That is, each member of the vector z will be raised to the power of 2, then the new members will be added.

Then the amount will be raised to the power of $\frac{1}{2}$.

That is, we would like to find budgets b so that the above function is minimal under the following constraints:

a) If there is no constraint of EF-TB:

Constraint 1:

$$\sum_{i \in [n]} \sum_{l \in [k_i]} x_{il} \cdot a_{ilj} = c_j + z_j \qquad \forall j \in [m],\ p_j > 0$$

x_{il} Did student i receive basket l (0 or 1)

a_{ilj} in basket l, student i received course j (0 or 1)

Therefore, the left wing is actually the demand for course j.
The meaning of the constraint is that for courses whose price is greater than 0, the demand for course j should be equal to the sum of capacity and excess demand.

Constraint 2:

$$\sum_{i \in [n]} \sum_{l \in [k_i]} x_{il} \cdot a_{ilj} \leq c_j + z_j \qquad \forall j \in [m],\ p_j = 0$$

Similar to constraint 1, only this time it is about courses whose price is equal to 0.

Constraint 3:

$$\sum_{l \in [k_i]} x_{il} = 1 \qquad \forall i \in [n]$$

That means each student gets exactly one basket.

b) If there is an EF-TB constraint:

For every two students $i,\ i' \in [n]$ so that $b_{0,i} > b_{0,i'}$ for every two baskets $l \in [k_i]$ and $l' \in [k_{i'}]$ we will check whether (contested) EF-TB Violated when basket l is assigned to student i, and basket l' is also assigned to student i'.

When this situation happens, in order to prevent it we will add the following constraint:

$x_{il} + x_{i'l'} \leq 1$

The meaning of the constraint is that if there is (contested) EF-TB, student i cannot receive basket l at the same time that student i' received basket l'. (make sure there is no jealousy)

The above linear program will give us the optimal perturbations that must be made to the student budgets to minimize the excess demand function. In addition, we will also get the minimum $||\bar{z}(u, c, p, b)||_1$.

3) **We will check whether $||\bar{z}(u, c, p, b)||_2$ is equal to zero:**

After we have executed the linear program from the previous section and received the allocations and the error, we can check if the error is equal to 0, if so we will return $p\ *= p, b\ *= b$.

Otherwise we will go to step 4.

4) **We will update the price vector p as follows:** $p \leftarrow p + \delta\bar{z}\,(u, c, p, b)$

That is, if the demand is positive then the price will go up, otherwise the price will go down.

Then return to step 2.

Runtime improvements:

- When δ is small compared to ε then the algorithm can reach a clearing error of 0, however the courses whose excess demand is large, the price will slowly increase from the initial price 0 to the final price. Therefore for these courses the algorithm will require at least $\frac{1}{\delta}$ steps to converge
- On the other hand, if the δ is large, then allocating a number of different budgets to each student will not be effective enough to reset the clearing error. Even if we take ε=0, the algorithm can reach a price close enough as it would give for a small δ. In this situation the algorithm will require much less steps due to the large step size.

According to these distinctions, our algorithm will combine discrete tatonnement and algorithm 1 to improve the running time. First we will run the algorithm with a large δ_0 for $\frac{1+\beta}{\delta_0}$ steps, then we will change to a small δ.

When we use the small step size δ, it may take many iterations of the algorithm to update some of the prices, for many of those iterations the possible demand remains constant for all students for many iterations in a row. Whenever this is the case, we can save time by doing a binary search of the next iteration where the excess demand changes.
A smarter optimization checks how the excess demand can be transformed into another vector, and to find it efficiently we can do a binary search.

Algorithm 2 – Find a profitable manipulation for a student

The purpose of the algorithm: to determine whether students have an incentive to deviate from the truth. The algorithm simulates the student's process for selecting their bids and tests whether bid manipulations will improve the student's service. That is, checks if the algorithm is immune to manipulation.

Input: distribution mechanism M (from Algorithm 1), student i and its preference function u_i, v_0 the best previous manipulation (in the first run of this algorithm u), a criterion for profitable manipulation that can be or a random sample denoted (u_{-i}, c, R) or they are given in the input and marked (U_{-i}, c, R)

Output: more profitable manipulation u_i'

Parameters: a local update coefficient η - the coefficient by which one course is updated at each step.

The algorithm:

1) We will update $v_0 \leftarrow u$,

u is taken as input and is the preferences the students reported. (or be the most profitable manipulation we found in a previous iteration with a different η).

2) We denote v_0 as $(\omega, valid, req)$

The function consists of the following parts:

A. A numerical value for each course - denoted as a vector ω, so that $\omega = (\omega_j)_{J\in[M]}$

B. Constraints on the courses a student **may** take, for example prerequisites, or overlap between courses - marked as a valid function. A boolean function that receives a bundle of courses, and returns true if the bundle is correct (meets all requirements) and false if the bundle is not correct.

C. Constraints on the courses a student **must** take, for example: compulsory courses, or at least 6 elective courses. Denoted as function req. A boolean function that receives a bundle of courses, and returns true if the bundle includes the required amount of courses, and false if not. The bundle can be correct even if it does not contain the required amount of courses, but its value is lower.

3) Try increasing or decreasing the weight w_j for each course j in v_0 to arrive at new misreports $V = \{v_j \pm 1\}_{j\in[m]}$

The group V is the group that has the exchange values at the end.

Every experiment $v_{j,k}$ does can be described by $(w', valid, req)$.

So the new weight $w'_{j'} = \begin{cases} w_j & j' \neq j \\ \eta^k w_j & j' = j \end{cases}$ for $j \in [m],\ k \in \{\pm 1\}$

That is, at this stage we go through all the courses and for each course separately we define a utility function and increase or decrease the weight,

In total, 2M options are tried because for each course 2 options are tried - to decrease or increase the weight

4) let it be

$$v^* = \begin{cases} \arg max_{v \in V \cup \{v_0\}} E_{r \sim R}\left[u_i\left(M_i\left(\left[v_j, u_{-i}\right], c, r\right)\right)\right] & resampled\ randomness \\ \arg max_{v \in V \cup \{v_0\}} E_{u_{-i} \sim U, r \sim R}\left[u_i\left(M_i\left(\left[v_j, u_{-i}\right], c, r\right)\right)\right] & resampledpopulation \end{cases}$$

To calculate this step we run the first algorithm for the vector of courses we received from step 3. And run one of the options resampled randomness, resampled population according to the criterion for profitable manipulation we received in the input.

We want to get the result that the function ui will have a larger expected value after we changed the weight of one course compared to the original function before the change.

In resampled randomness even if they have perfect knowledge of their friends' preferences, the students will never be able to anticipate the randomness of the first algorithm, what is random in it is the budget disturbance.

The input for running the first algorithm is the course assignments (u, c) and random r,

Here, as the input for u, we enter $[v_j, u_{-i}]$ which means that all the students' preferences remain the same except for student i whose preference is v_j.

The input of member preferences here is constant for all iterations, so a student will be able to manipulate and change their preferences. (For example, in the event that a student knows about a certain course that the other students almost do not want, and he gave it a high preference, he will be able to lower the preference for this course)

The input to the initial budget b_0 (in Algorithm 1) is determined randomly by r. r is a parameter chosen in a uniform distribution in the domain $[1 + \frac{\beta}{4},\ 1 + \frac{3\beta}{4}]$ where β is a parameter.

In the resampled population there is additional uncertainty for the students regarding the preferences of their friends.

We randomly choose both r as explained above and the values of the other players.
The values of the other players are selected from a database of students who registered on the site and filled in their values randomly and with repetitions.
In each iteration of the algorithm, samples are resampled from the database, so that the preferences of the other students change in each iteration and the student cannot manipulate and rely on them.

The output is an assignment $(a_i)_{(i=1)}^{n} = (M_i(u,c,r))_{(i=1)}^{n}$ so $M_i(u,c,r) \subseteq [m]$ marks the courses that the mechanism assigned to student i.

We will mark the above output as X and run the utility function of the ith student $u_i(x)$ on it,
The function is calculated as follows:

$$u(x) = \begin{cases} w \cdot x + b & valid(x) = 1, req(x) = 1 \\ w \cdot x & valid(x) = 1, req(x) = 0 \\ -\infty & valid(x) = 0 \end{cases}$$

where B represents a student's level of happiness when he gets all the courses he needs. B is a number greater than the sum of all values of a student in all courses. We will choose to represent it as the sum of all the values in the vector $w + 1$.
For each wrong report v we received we will run the calculation $u_i(M_i([v_j, u_{-i}], c, r))$ twice (for example 10) and for all the results we will calculate the expected value..
After we performed these calculations for each wrong report v we chose v^* to be $argmax_{v \in V \cup \{v_0\}}$

5) If $v^* = v_0$ the algorithm ends and v_0 is the best manipulation.
If $v_0 = u$, we return failed because there was no manipulation that gave us something better. That is v_0 is the best manipulation.

6) Otherwise, we will update $v_0 \leftarrow v^*$, and return to step 2.

The algorithm changes one course each time and each time it chooses the course that will result in the greatest improvement and you know this according to the argmax.

Algorithm 3 – Tabu search

The purpose of the algorithm: to find a distribution of courses for students.

Input: capacity of the courses $\boldsymbol{c}$, preference functions $\boldsymbol{u}$, initial budget $b_0 \in [1,\ 1\ +\ \beta]^n$

(where β is an accepted parameter, and each student has a different initial budget)

Output: final prices $p\ *$

Parameters: neighborhood function $N(p)$, binary relation to prices $\sim_p$

The algorithm:

1) **We initialize a random price vector $p \leftarrow U(1, 1\ +\ \beta)^m$ (where β is the obtained parameter), and initialize the price history group $H \leftarrow \phi$**
2) **We will check whether $||z(u, c, p, b)||_2\ =\ 0$**

a) Calculate $z(u, c, p, b)$:

- For each student i, consider the set of courses a_i that maximizes his utility under the given constraints, while maintaining the budget.

  $a_i\ (u, p, b_0\ )\ =\ arg\ max_{(x \in 2\text{^}[m]\ , p \cdot x \leq b\_0\ )}\ u_i\ (x)$

- For each course j, we calculate the excess demand function, which is the difference between the number of students interested in the course and the capacity of the course: $z_j(u, c, p, b)\ =\ \sum_{i=1}^{n} a_{ij}(u, p, b)\ -\ c_j$

  where the expression $a_{ij}\ (u, p, b)$ will be equal to 1 if student i is interested in course j, and 0 otherwise.

  Finally we will get the vector $z\ =\ (z_j)_{j=1}^{m}$ which represents the market clearing error of the given courses.

b) Calculate $||z(u, c, p, b)||_2$:

The calculation of the norm of the vector will be performed as follows:

$$||z(u, c, p, b)||_2\ =\ \sqrt{(z_1)^2 + (z_2)^2 +\ \cdots\ +\ (z_m)^2}$$

If we got $||z(u,c,p,b)||_2 = 0$ **we will update** $p^* = p$ **and finish.**

3) Otherwise,

- **We will include all prices equivalent to p in history:** $H \leftarrow H + \{p' : p' \sim_p p\}$

  A price vector p' will be called "equivalent" to a price vector p if it holds that for each student, the allocation function $a = (a_i)_{i=1}^n$ given p will be equal to the allocation given p': $\forall\, student\ i \in [n],\ a_i(u,p,b_0) = a_i(u,p',b_0)$.

- **We will update** $p \leftarrow arg\ min_{p' \in N(p)-H} \|z(u,c,p,b_0)\|_2$

  The neighborhood function $N(p)$ consists of the following two types of neighbors:

  1) Neighbors on the slope (Gradient):

  We will adjust the prices in relation to the excess demand. We will increase the prices of the courses for which there is a high demand, and we will decrease the prices of the courses for which the demand is low. The change will be made in relation to the values we get in the excess demand function $z(u,\ c,\ p,\ b)$ for a time-varying set $\Delta \subseteq [0,1]$. The slope neighbors will consist of the following prices:

  $N^{gradient}(p,\ \Delta) = \{p + \delta \cdot z(u,c,p,b) : \delta \in \Delta\}$

  # Although it is not specified in the article - we will use $\bar{z}(u,c,p,b)$, when there is a price equal to 0. In addition, we will make sure that the price does not become negative - in such a situation, we will update it to 0.

  2) Neighbors with individual price adjustments:

  In this type of neighborhood, the price vector will change each time in relation to only one course. For courses with excess demand, we would like to increase their price so that the demand decreases by one. That is, we will increase the price of a course until the budget of one of the students no longer allows him to take the course. For courses for which the demand is less than the capacity, we would like to set their price to 0, or reduce their price by a fixed number. In order to prevent the algorithm from checking too many neighbors of this type, it will be limited to checking 35 different price matches.

After finding the neighborhood function $N(p)$, we will check for each price its market clearing error $z(u,c,p,b)$ and calculate the norm $\|z(u,c,p,\ b_0)\|_2$. The price that brings the lowest result in the norm will be updated to be the new p.
We will return to step 2) in the algorithm.

Experiments

Algorithm 1 – Find an A-CEEI with (contested) EF-TB property

The paper describes the experiments conducted to evaluate the performance of the algorithm.
Below are the key points related to experiments, comparisons and algorithm performance:

1. Benchmarking

The proposed FAST algorithm (in subsection 3.2) is compared against a benchmark algorithm described as the advanced (previous) commercial algorithm.
Two intermediate algorithms are also considered: Vanilla tatonnement (without optimizations such as tabu search, individual price adjustments or optimal budget perturbations) and Basic (the algorithm described in subsection 3.1, adding optimal budget perturbations to Vanilla tatonnement but without additional optimizations).

2. Selection of parameters

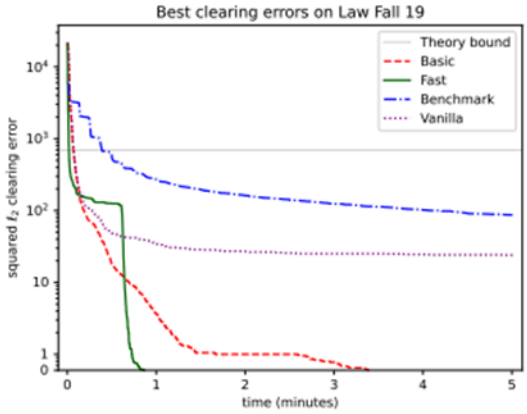
Random and optimal budget perturbations are applied to student budgets in different algorithms.
Parameters such as the size of the total budget disruptions are kept consistent for a fair comparison.

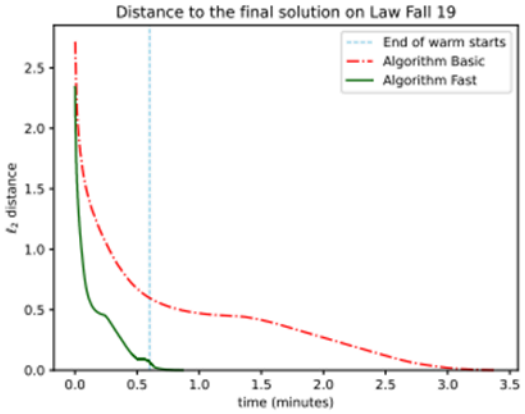
3. Cases

Experiments are conducted using large shows of different types of schools (law school, ivy league business school with varying sizes and challenging constraints).
Student schedules in these cases usually include about 5 courses.

4. Findings

Figure 2 is shown, showing the average best clearing error found by the four algorithms over time.





Observations are made based on the findings.
For example, optimal budget perturbations have been observed to significantly improve clearing errors.

5. Distinctions on algorithm performance

Distinction 1: Optimal budget perturbations significantly improve error cleanup, outperforming algorithms without budget perturbations.
Distinction 2: Individual price adjustments do not seem to help that much, contrary to previous findings. The time to iterate the comparison algorithm is a factor.
Distinction 3: The optimal algorithm initially has larger cleaning errors than the basic algorithm, but this is explained by the transition from a warm start (the starting point of the solution) to the second step.

6. Comparative analysis

The article discusses the surprising finding that individual price adjustments, while theoretically useful, slowed down the benchmark algorithm compared to vanilla.

Algorithm 2 – Find a profitable manipulation for a student

Experiments conducted

1. The manipulation-finding algorithm

An algorithm is developed to investigate whether students using the A-CEEI algorithm have an incentive to deviate from truth-telling.
The algorithm uses a hill-climbing approach, and iteratively searches for course bid manipulations that increase the student's utility.
The algorithm is based on a course allocation mechanism that can be manipulated.
The algorithm was used at Harvard Business School (HBS).

2. Random mechanism and A-CEEI algorithm:

The A-CEEI algorithm is considered as a random mechanism with fixed parameters.
A random mechanism is commonly defined for course assignment problems, and the HBS mechanism is used as an example.
Parameters such as base budgets are randomly determined.

3. Creating models of students' uncertainty:

Two uncertainty models are considered: random resampled and population resampled.
Resampled randomness assumes that students have perfect knowledge of peers' preferences but are uncertain about the randomness of the algorithm.
A resampled population adds uncertainty about the preferences of peers.

4. Statistical findings:

The results are presented in Table 1, where it can be seen that they tested this on three different schools, for each school they used two uncertainty models, and the mechanism for choosing the courses with the constraint type of the EF-TB.

Instance	Uncertainty	Mechanism							
		No EF-TB		Classic EF-TB		Contested EF-TB		HBS	
		#	Gain	#	Gain	#	Gain	#	Gain
Small	Randomness	1	0.04%	0	-	1	0.04%	66	8.0 ± 2.1%
	Population	0	-	0	-	0	-	63	9.0 ± 2.1%
Ivy Small	Randomness	21	13.5 ± 3.6%	15	8.5 ± 2.2%	1	0.02%	107	23.5 ± 3.6%
	Population	20	7.4 ± 1.4%	11	4.7 ± 1.0%	0	-	117	20.7 ± 3.0%
Biz	Randomness	0	-	0	-	0	-	87	26.3 ± 4.0%
	Population	1	0.7%	0	-	0	-	64	21.6 ± 2.8%

Table 1. Number of statistically significant manipulations found by our algorithm and their average relative gain (± standard error) in utility.

The manipulation experiments are performed on cases with different characteristics, such as Small, Ivy Small and Biz, which represent business schools of varying sizes.
The A-CEEI algorithm is compared against the manipulable HBS mechanism, and the number of profitable manipulations and their average profits are reported.
It should be noted that the algorithm finds few statistically significant manipulations for the Small and Biz cases, indicating lower manipulation ability compared to Ivy Small.
When contested EF-TB was used, there was almost no ability to manipulate.

Performance evaluation

The performance of the algorithm is evaluated based on its ability to identify profitable manipulations in different scenarios.
The comparison with the HBS mechanism serves as a benchmark, and the success of the algorithm in finding manipulations indicates its effectiveness.
In summary, the experiments include developing an algorithm for detecting manipulations, creating models of the students' uncertainty, performing practical optimizations and analyzing manipulations on different cases. The performance of the algorithm is evaluated by comparing its findings to a known mechanism that can be manipulated. The results indicate that the A-CEEI algorithm exhibits lower manipulability in some cases compared to the benchmark mechanism used at Harvard Business School.

Examples of manipulations in Algorithm 2

An example from the article for profitable manipulation without EFTB constraints: (5.1)
Alice and Bob want only the last seat of CS161, due to demand from other students ECON101 is full but has a low price.
With true reports - CS161 will go to either Alice or Bob, it depends on the random starting budget.
Bob can manipulate and report his preferences so that he wants ECON101 as his second choice.
Under the manipulated preferences, since ECON101 is cheap, the only way the course would not be assigned to Bob is if Bob spent his entire budget on CS161.
Even if Alice's initial budget is higher, the optimal and unbounded budget disturbance will ensure that Bob's final initial budget is higher than Alice's, in this case - Bob gets CS161 and Alice gets nothing - equilibrium.

For this manipulation to work, Bob would have to know that at equilibrium prices ECON101 is already exactly full by other students – knowledge he is unlikely to have in realistic bids. Indeed, suppose that ECON101 demand was higher this semester, the algorithm raised its price, and now it lacks exactly one student: in this case the optimal budget disruption would have to ensure that Bob does enter ECON101, which can be achieved by budgeting disruption against Bob and letting Alice take the last seat in CS161.
However, this manipulation is quite strong if the price of ECON101 is always very low ("ECON101 tends to be low price" is a general fact that a student can reasonably study historical proposals). In this case, even if ECON101 lacks a student, Bob's budget may be larger than Alice's by enough of a margin to afford both CS161 and ECON101. Therefore, when ECON101 is not full, the budget disturbance can go either way, but it always goes in favor of strategic Bob when ECON101 is overfull.

An example from the article for profitable manipulation with EFTB constraints: (5.2)
Alice and Bob both want the last seat on the popular CS161 course. But Alice wants even more to take independent research units with her advisor, in this course there is unlimited space, so the price is always zero,
She would like to take both.
Bob's second choice is ECON101, due to demand from other students, ECON101 is full but has a low price.
In the situation where students report true preferences, since ECON101 is cheap, the only way it won't be assigned to Bob is if Bob exhausts his budget on CS161. So even if Alice ranks higher, the optimal budget constraint sets her budget lower than Bob. In this case Alice always gets independent study units (only), and Bob always gets CS161 (only) - equilibrium.
If Alice misreports her preferences to rank independent study units lower than CS161, she will be jealous of Bob every time he gets the last seat in CS161.
Whenever her initial budget is higher, the optimal budget disruption increases Alice's chances of getting the last seat at CS161.
We'll notice that the lower undisputed course rating (independent study units) never hurts Alice. Thus this manipulation is profitable in anticipation of Alice even if she only has very vague information about the rank and demand of other students.
(Interestingly, this manipulation works because of the EF-TB constraints we introduce to prevent manipulations!)

An example from the article for profitable manipulation with contested EF-TB constraints: (5.3)

Many students, including Bob, like to take CS161 and ECON101 together, but they rank ECON101 above CS161.

Alice already took ECON101 last semester, so she only wants the last seat in CS161. Due to demand from other students, ECON101 is full but has a very low price.

With real preferences, if Bob's budget is higher than Alice's by a margin greater than the price of ECON101, he can afford both CS161 and ECON101, leaving Alice with nothing.

Alice could have manipulated her preferences to report that she wanted ECON101 as her second course.

Bob always gets ECON101, because it's cheap and he makes it his priority.

Whenever Alice doesn't get CS161, she has to get ECON101 (because its price is cheap).

If both Bob and Alice take ECON101, there are too many people enrolled in the course, which will cause a clearing error.

Therefore the optimal budget constraint will ensure that Bob's budget is low enough compared to Alice's that he cannot afford both courses: he will take ECON101, and Alice will take CS161 - equilibrium.

As in Example 2, this manipulation does pose some risk – if ECON101 is not full, the optimal budget disruption may reduce Alice's budget below the price of CS161 so that she has to take ECON101. However, if ECON101 is very cheap, there is also always the small disruption that increases Alice's budget so that she can afford both ECON101 and CS161 (while Bob only funds ECON101).

Therefore, because of the asymmetry in the prices of CS161 and ECON101, if ECON101 is overfilled, this manipulation can increase Alice's chances of entering CS161, but if ECON101 is underfilled, Alice's chances are not affected much.

Algorithm 3 – Tabu search

Evaluation metrics

1. Utilitarian Social Welfare (USW):

Average USW is calculated for each algorithm, taking into account the base budget of each student.
USW is defined as the sum of the products of base and benefit budgets for each student.
Reflects how satisfied the students are.

2. Nash Social Welfare (NSW):

NSW is a geometric mean (if all budgets are equal it is the root of the product) compared between the proposed A-CEEI algorithm and the entire base algorithm.
NSW is defined as the product of the benefits in holding the base budget for each student divided by the sum of the base budgets.
Here, too, this index reflects how satisfied the students are while maintaining the balance of NASH.
Results (Table 2):

Instance	Avg. NSW			Avg. USW		
	Fast	Benchmark	Imp.	Fast	Benchmark	Imp.
Ivy Huge Fall 21	470.44	-	$+\infty$	552.92	541.5	2.11%
Law Fall 19	846.66	837.29	1.12%	1045.25	1036.76	0.82%
Law Fall 20	646.16	572.66	12.83%	848.04	764.39	10.94%
Law Fall 21	168.93	150.85	11.99%	194.15	176.24	10.16%
Ivy Small Fall 19	444.32	445.86	−0.35%	567.31	564.43	0.51%
Ivy Small Fall 20	421.54	415.86	1.37%	501.29	496.28	1.01%
Ivy Small Fall 21	437.49	444.79	−1.64%	535.47	540.59	−0.95%
Ivy Small Fall 22	462.49	462.77	−0.06%	521.4	522.94	−0.29%
Biz Fall 18	203.69	202.75	0.46%	232.53	231.77	0.33%
Biz Fall 19	191.53	189.0	1.34%	222.21	219.88	1.06%
Biz Fall 20	209.41	191.97	9.08%	227.47	212.09	7.25%
Biz Fall 21	234.95	234.62	0.14%	240.77	240.51	0.11%

Table 2. Comparing the (nash) social welfare.

The table summarizes the comparison of Nash social welfare between different cases.
The "Imp" - a column indicating the percentage of improvement of the A-CEEI algorithm compared to the index.

The results show that at a high level, the A-CEEI algorithm performs better in both measures in most cases.

Distinctions

The A-CEEI algorithm results in a situation where everyone is satisfied in many cases.
The percentages of improvement vary between cases, indicating that the proposed algorithm is not universally superior but performs competitively or better on average.

Running examples

Algorithm 1 – Running examples

Example run 1

$n = 2$ *# Number of students*

$m = 3$ *# Number of courses*

parameters:

$\epsilon = 0.5$

$\delta = 0.5$

$t = 0$ *# The EF − TB constraint*

$capacities = \{"x": 1, "y": 1, "z": 2\}$

$utilities = \{"ami": \{"x": 1, "y": 2, "z": 4\}, "tami": \{"x": 2, "y": 3, "z": 1\}\}$

$initial\ budgets = \{"ami": 2, "tami": 3\}$

$required = \{"ami": 2, "tami": 2\}$ *# The number of courses each student should take*

Running the algorithm:

1) $prices = \{"x": 0, "y": 0, "z": 0\}$
2) $\epsilon - budget\ perturbation$:

   # For each student the range initial budget ± ε

   $range_{ami} = [1.5, 2.5]$

   $range_{tami} = [2.5, 3.5]$

   # The number of bundles in range initial budget ± ε

   $k = \{"ami": 1, "tami": 1\}$

   # The represents of all budgets from the range initial budget ± ε

   $b_{ami} = [2]$

   $b_{tami} = [3]$

   # The bundles that gives maximum utility for each student

   $a_{ami} = [0, 1, 1] = [y, z]$

$a_{tami} = [1,\ 1,\ 0]\ =\ [x, y]$

Results of the linear program:

 # The market clearing error – excess demand function

$z = \{"x":\ 0,\ "y":\ 1,\ "z":\ 0\}$

 # The budgets that will give the minimum of the market clearing error

$b = \{"ami":\ 2,\ "tami":\ 3\}$

3) $||z||_2 = 1 \neq 0$

4) $update\ p = p + \delta \cdot z$

$= [0,\ 0,\ 0] + 0.5 \cdot [0,\ 1,\ 0]$

$= [0,\ 0.5,\ 0] \rightarrow \{"x":\ 0,\ "y":\ 0.5,\ "z":\ 0\}$

Iteration 2:

2) ϵ – *budget perturbation*:

$range_{ami} = [1.5, 2.5]$

$range_{tami} = [2.5, 3.5]$

$k = \{"ami":\ 1,\ "tami":\ 1\}$

$b_{ami} = [2]$

$b_{tami} = [3]$

$a_{ami} = [0,\ 1,\ 1]\ =\ [y, z]$

$a_{tami} = [1,\ 1,\ 0]\ =\ [x, y]$

Results of the linear program:

$z = \{"x":\ 0,\ "y":\ 1,\ "z":\ 0\}$

$b = \{"ami":\ 2,\ "tami":\ 3\}$

3) $||z||_2 = 1 \neq 0$

4) $update\ p = p + \delta \cdot z$

$= [0,\ 0.5,\ 0] + 0.5 \cdot [0,\ 1,\ 0]$

$= [0,\ 1,\ 0] \rightarrow \{"x":\ 0,\ "y":\ 1,\ "z":\ 0\}$

Iteration 3:

2) ϵ – *budget perturbation*:

$range_{ami} = [1.5, 2.5]$

$range_{tami} = [2.5, 3.5]$

$k = \{"ami" : 1, "tami" : 1\}$

$b_{ami} = [2]$

$b_{tami} = [3]$

$a_{ami} = [0, 1, 1] = [y, z]$

$a_{tami} = [1, 1, 0] = [x, y]$

Results of the linear program:

$z = \{"x": 0, "y": 1, "z": 0\}$

$b = \{"ami" : 2, "tami" : 3\}$

3) $||z||_2 = 1 \neq 0$

4) $update\ p = p + \delta \cdot z$

$= [0, 1, 0] + 0.5 \cdot [0, 1, 0]$

$= [0, 1.5, 0] \rightarrow \{"x": 0, "y": 1.5, "z": 0\}$

Iteration 4:

2) $\epsilon -$ *budget perturbation*:

$range_{ami} = [1.5, 2.5]$

$range_{tami} = [2.5, 3.5]$

$k = \{"ami" : 1, "tami" : 1\}$

$b_{ami} = [2]$

$b_{tami} = [3]$

$a_{ami} = [0, 1, 1] = [y, z]$

$a_{tami} = [1, 1, 0] = [x, y]$

Results of the linear program:

$z = \{"x": 0, "y": 1, "z": 0\}$

$b = \{"ami" : 2, "tami" : 3\}$

3) $||z||_2 = 1 \neq 0$

4) $update\ p = p + \delta \cdot z$

$= [0, 1.5, 0] + 0.5 \cdot [0, 1, 0]$

$= [0, 2, 0] \rightarrow \{"x": 0, "y": 2, "z": 0\}$

Iteration 5:

2) $\epsilon - budget\ perturbation$:

$range_{ami} = [1.5, 2.5]$

$range_{tami} = [2.5, 3.5]$

$k = \{"ami" : 2, "tami" : 1\}$

$b_{ami} = [1.5, 2]$

$b_{tami} = [3]$

$a_{ami} = [1, 0, 1] = [x, z]$, $a_{ami} = [0, 1, 1] = [y, z]$

$a_{tami} = [1, 1, 0] = [x, y]$

Results of the linear program:

$z = \{"x": 1, "y": 0, "z": 0\}$

$b = \{"ami" : 1.5, "tami" : 3\}$

3) $||z||_2 = 1 \neq 0$

4) $update\ p = p + \delta \cdot z$

$= [0, 2, 0] + 0.5 \cdot [1, 0, 0]$

$= [0.5, 2, 0] \rightarrow \{"x": 0.5, "y": 2, "z": 0\}$

Iteration 6:

$range_{ami} = [1.5, 2.5]$

$range_{tami} = [2.5, 3.5]$

$k = \{"ami" : 2, "tami" : 1\}$

$b_{ami} = [1.5, 2]$

$b_{tami} = [3]$

$a_{ami} = [1, 0, 1] = [x, z]$, $a_{ami} = [0, 1, 1] = [y, z]$

$a_{tami} = [1, 1, 0] = [x, y]$

Results of the linear program:

$z = \{"x": 1, "y": 0, "z": 0\}$

$b = \{"ami" : 1.5, "tami" : 3\}$

3) $||z||_2 = 1 \neq 0$

4) $update\ p = p + \delta \cdot z$

$= [0.5, 2, 0] + 0.5 \cdot [1, 0, 0]$

$$= [1, 2, 0] \rightarrow \{"x": 1, "y": 2, "z": 0\}$$

Iteration 7:

$range_{ami} = [1.5, 2.5]$

$range_{tami} = [2.5, 3.5]$

$k = \{"ami": 2, "tami": 2\}$

$b_{ami} = [1.5, 2]$

$b_{tami} = [2.5, 3]$

$a_{ami} = [1, 0, 1] = [x, z]$, $a_{ami} = [0, 1, 1] = [y, z]$

$a_{tami} = [0, 1, 1] = [y, z]$, $a_{tami} = [1, 1, 0] = [x, y]$

Results of the linear program:

$z = \{"x": 0, "y": 0, "z": 0\}$

$b = \{"ami": 1.5, "tami": 2.5\}$

3) $||z||_2 = 0$

4) $update\ p^* = p = \{"x": 0.5, "y": 2, "z": 0\}$

$b^* = b = \{"ami": 1.5, "tami": 2.5\}$

Algorithm results:

Final course prices: $p^* = \{"x": 0.5, "y": 2, "z": 0\}$

Final budgets : $b^* = \{"ami": 1.5, "tami": 2.5\}$

Final distribution of courses: $ami: [1, 0, 1] = [x, z]$

$tami: [0, 1, 1] = [y, z]$

Example run 2

$n = 2$ # Number of students

$m = 3$ # Number of courses

parameters:

$\epsilon = 1$

$\delta = 0.5$

$t = 1$

$capacities = [1, 1, 2]$

$utilities = [[5, 2, 1], [4, 1, 3]]$

$b_0 = [3, 4]$

$required = [2, 2]$ # The number of courses each student should take

Running the algorithm:

1) $prices = [0, 0, 0]$
2) $\epsilon - budget\ perturbation$:

   $k = [1, 1]$

   $b_{00} = [3]$

   $b_{01} = [4]$

   $a_{00} = [1, 1, 0]$, $a_{10} = [1, 0, 1]$

   $\forall j \in m, p_j = 0 \rightarrow No\ jealousy$

   $Results\ of\ the\ linear\ program$:

   $z = [1, 0, 0]$

   $b = [3, 4]$
3) $||z||_2 = 1 \neq 0$
4) $update\ p = p + \delta \cdot z$

   $= [0, 0, 0] + 0.5 \cdot [1, 0, 0]$

   $= [0.5, 0, 0]$

Iteration 2:

2)$\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [3]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 0,\ 1]$

 # No jealousy

Results of the linear program:

$z = [1,\ 0,\ 0]$

$b = [3, 4]$

3) $||z||_2 = 1 \neq 0$

4) update $p = p + \delta \cdot z$

$= [0.5,\ 0,\ 0] + 0.5 \cdot [1,\ 0,\ 0]$

$= [1,\ 0,\ 0]$

Iteration 3:

2)ϵ $-$ *budget perturbation*:

$k = [1, 1]$

$b_{00} = [3]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 0,\ 1]$

 # No jealousy

Results of the linear program:

$z = [1,\ 0,\ 0]$

$b = [3, 4]$

3) $||z||_2 = 1 \neq 0$

4) *update* $p = p + \delta \cdot z$

$= [1,\ 0,\ 0] + 0.5 \cdot [1,\ 0,\ 0]$

$= [1.5,\ 0,\ 0]$

Iteration 4:

2) ϵ $-$ *budget perturbation*:

$k = [1, 1]$

$b_{00} = [3]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 0,\ 1]$

 # No jealousy

Results of the linear program:

$z = [1, 0, 0]$

$b = [3, 4]$

3) $||z||_2 = 1 \neq 0$

4) $update\ p = p + \delta \cdot z$

$= [1.5, 0, 0] + 0.5 \cdot [1, 0, 0]$

$= [2, 0, 0]$

Iteration 5:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [3]$

$b_{01} = [4]$

$a_{00} = [1, 1, 0]\,,\ a_{10} = [1, 0, 1]$

 # No jealousy

Results of the linear program:

$z = [1, 0, 0]$

$b = [3, 4]$

3) $||z||_2 = 1 \neq 0$

4) $update\ p = p + \delta \cdot z$

$= [2, 0, 0] + 0.5 \cdot [1, 0, 0]$

$= [2.5, 0, 0]$

Iteration 6:

2)$\epsilon - budget\ perturbation$:

$k = [2, 1]$

$b_{00} = [2, 3]$

$b_{01} = [4]$

$a_{00} = [0, 1, 1]\,, a_{01} = [1, 1, 0],\ a_{10} = [1, 0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 0, 0]$

$b = [2, 4]$

3) $||z||_2 = 0$

4) $update: p^* = p = [2.5, 0, 0]$

$b^* = b = [2, 4]$

Algorithm results:

Final course prices: $p^* = [2.5, 0, 0]$

Final budgets : $b^* = [2, 4]$

Final distribution of courses: A: $[0, 1, 1]$

B: $[1, 0, 1]$

Example run 3

$n = 2$ # Number of students

$m = 3$ # Number of courses

parameters:

$\epsilon = 2$

$\delta = 0.5$

$t = 1$

$capacities = [1, 2, 2]$

$utilities = [[5, 5, 1], [4, 6, 4]]$

$b_0 = [5, 4]$

$required = [2, 2]$ # The number of courses each student should take

Running the algorithm:

1) $prices = [0, 0, 0]$
2) $\epsilon - budget\ perturbation$:

   $k = [1, 1]$

   $b_{00} = [5]$

   $b_{01} = [4]$

   $a_{00} = [1, 1, 0]$, $a_{10} = [1, 1, 0]$

   $\forall j \in m, p_j = 0 \rightarrow No\ jealousy$

   $Results\ of\ the\ linear\ program$:

   $z = [1, 0, 0]$

   $b = [5, 4]$
3) $||z||_2 = 1 \neq 0$
4) $update: p = p + \delta \cdot z$

   $= [0, 0, 0] + 0.5 \cdot [1, 0, 0]$

   $= [0.5, 0, 0]$

Iteration 2:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [5]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 1,\ 0]$

 # No jealousy

Results of the linear program:

$z = [1,\ 0,\ 0]$

$b = [5, 4]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.5,\ 0,\ 0] + 0.5 \cdot [1, 0,\ 0]$

$= [1,\ 0,\ 0]$

Iteration 3:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [5]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 1,\ 0]$

 # No jealousy

Results of the linear program:

$z = [1,\ 0,\ 0]$

$b = [5, 4]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [1,\ 0,\ 0] + 0.5 \cdot [1, 0,\ 0]$

$= [1.5,\ 0,\ 0]$

Iteration 4:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [5]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 1,\ 0]$

 # No jealousy

Results of the linear program:

$z = [1,\ 0,\ 0]$

$b = [5, 4]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [1.5,\ 0,\ 0] + 0.5 \cdot [1, 0,\ 0]$

$= [2,\ 0,\ 0]$

Iteration 5:

2)ϵ $-$ *budget perturbation*:

$k = [1, 1]$

$b_{00} = [5]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 1,\ 0]$

 # No jealousy

Results of the linear program:

$z = [1,\ 0,\ 0]$

$b = [5, 4]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [1.5,\ 0,\ 0] + 0.5 \cdot [1, 0,\ 0]$

$= [2,\ 0,\ 0]$

Iteration 6:

2) ϵ $-$ *budget perturbation*:

$k = [1, 1]$

$b_{00} = [5]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 1,\ 0]$

 # No jealousy

Results of the linear program:

$z = [1,\ 0,\ 0]$

$b = [5, 4]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [2,\ 0,\ 0] + 0.5 \cdot [1, 0,\ 0]$

$= [2.5,\ 0,\ 0]$

Iteration 7:

2) $\epsilon - budget\ perturbation:$

$k = [1, 2]$

$b_{00} = [5]$

$b_{01} = [2, 4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [0,\ 1,\ 1]\,, a_{11} = [1,\ 1,\ 0]$

 # No jealousy

Results of the linear program:

$z = [0,\ 0,\ 0]$

$b = [5, 2]$

3) $||z||_2 = 0$

4) $update: p^* = p = [2.5, 0, 0]$

$b^* = b = [5, 2]$

Algorithm results:

Final course prices: $p^* = [2.5, 0, 0]$

Final budgets : $b^* = [5, 2]$

Final distribution of courses: A: $[1,\ 1,\ 0]$

B: $[0,\ 1,\ 1]$

Example run 4

$n = 2$ # Number of students

$m = 2$ # Number of courses

parameters:

$\epsilon = 0.2$

$\delta = 0.1$

$t = 1$

$capacities = [1, 1]$

$utilities = [[10, 20], [10, 20]]$

$b_0 = [1.1, 1]$

$required = [1, 1]$ # The number of courses each student should take

Running the algorithm:

1) $prices = [0, 0]$
2) $\epsilon - budget\ perturbation$:

   $k = [1, 1]$

   $b_{00} = [1.1]$

   $b_{01} = [1]$

   $a_{00} = [0, 1],\ a_{10} = [0, 1]$

   $\forall j \in m,\ p_j = 0 \rightarrow No\ jealousy$

   $Results\ of\ the\ linear\ program$:

   $z = [0, 1]$

   $b = [1.1,\ 1]$
3) $||z||_2 = 1 \neq 0$
4) $update: p = p + \delta \cdot z$

   $= [0, 0] + 0.1 \cdot [0, 1]$

   $= [0, 0.1]$

Iteration 2:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [0, 1],\ a_{10} = [0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 1]$

$b = [1.1,\ 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0,\ 0.1] + 0.1 \cdot [0, 1]$

$= [0, 0.2]$

Iteration 3:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [0, 1],\ a_{10} = [0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 1]$

$b = [1.1,\ 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0,\ 0.2] + 0.1 \cdot [0, 1]$

$= [0, 0.3]$

Iteration 4:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [0, 1],\ a_{10} = [0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 1]$

$b = [1.1,\ 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0,\ 0.3] + 0.1 \cdot [0, 1]$

$= [0, 0.4]$

Iteration 5:

2) $\epsilon - budget\ perturbation:$

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [0, 1],\ a_{10} = [0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 1]$

$b = [1.1,\ 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0,\ 0.4] + 0.1 \cdot [0, 1]$

$= [0, 0.5]$

Iteration 6:

2) $\epsilon - budget\ perturbation:$

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [0, 1],\ a_{10} = [0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 1]$

$b = [1.1,\ 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0,\ 0.5] + 0.1 \cdot [0, 1]$

$= [0, 0.6]$

Iteration 7:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [0, 1],\ a_{10} = [0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 1]$

$b = [1.1,\ 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0,\ 0.6] + 0.1 \cdot [0, 1]$

$= [0, 0.7]$

Iteration 8:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [0, 1],\ a_{10} = [0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 1]$

$b = [1.1,\ 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0,\ 0.7] + 0.1 \cdot [0, 1]$

$= [0, 0.8]$

Iteration 9:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [0, 1],\ a_{10} = [0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 1]$

$b = [1.1,\ 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0,\ 0.8] + 0.1 \cdot [0, 1]$

$= [0, 0.9]$

Iteration 10:

2) $\epsilon - budget\ perturbation$:

$k = [1, 2]$

$b_{00} = [1.1]$

$b_{01} = [0.8,\ 1]$

$a_{00} = [0, 1],\ a_{10} = [1, 0]\ ,\ a_{11} = [0, 1]$

 # No jealousy

Results of the linear program:

$z = [0, 0]$

$b = [1.1,\ 0.8]$

3) $||z||_2 = 0$

4) $update$: $p^* = p = [0,\ 0.9]$

$b^* = b = [1.1,\ 0.8]$

Algorithm results:

Final course prices: $p^* = [0,\ 0.9]$

Final budgets : $b^* = [1.1,\ 0.8]$

Final distribution of courses: A: $[0, 1]$

B: $[1, 0]$

Example run 5

$n = 2$ # Number of students

$m = 1$ # Number of courses

parameters:

$\epsilon = 0.2$

$\delta = 0.1$

$t = 1$

$capacities = [1]$

$utilities = [[2], [3]]$

$b_0 = [1.1, 1]$

$required = [1, 1]$ # The number of courses each student should take

Running the algorithm:

1) $prices = [0]$
2) $\epsilon - budget\ perturbation$:

   $k = [1, 1]$

   $b_{00} = [1.1]$

   $b_{01} = [1]$

   $a_{00} = [1]\ ,\ a_{10} = [1]$

   $\forall j \in m,\ p_j = 0 \rightarrow No\ jealousy$

   $Results\ of\ the\ linear\ program$:

   $z = [1]$

   $b = [1.1, 1]$
3) $||z||_2 = 1 \neq 0$
4) $update: p = p + \delta \cdot z$

   $= [0] + 0.1 \cdot [1]$

   $= [0.1]$

Iteration 2:

2)$\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [1]\,,\ a_{10} = [1]$

 # No jealousy

Results of the linear program:

$z = [1]$

$b = [1.1, 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.1] + 0.1 \cdot [1]$

$= [0.2]$

Iteration 3:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [1]\,,\ a_{10} = [1]$

 # No jealousy

Results of the linear program:

$z = [1]$

$b = [1.1, 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.2] + 0.1 \cdot [1]$

$= [0.3]$

Iteration 4:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [1]\,,\ a_{10} = [1]$

 # No jealousy

Results of the linear program:

$z = [1]$

$b = [1.1, 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.3] + 0.1 \cdot [1]$

$= [0.4]$

Iteration 5:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [1]\,,\ a_{10} = [1]$

 # No jealousy

Results of the linear program:

$z = [1]$

$b = [1.1, 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.4] + 0.1 \cdot [1]$

$= [0.5]$

Iteration 6:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [1]\,,\ a_{10} = [1]$

 # No jealousy

Results of the linear program:

$z = [1]$

$b = [1.1, 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.5] + 0.1 \cdot [1]$

$= [0.6]$

Iteration 7:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [1]\,,\ a_{10} = [1]$

 # No jealousy

Results of the linear program:

$z = [1]$

$b = [1.1, 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.6] + 0.1 \cdot [1]$

$= [0.7]$

Iteration 8:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [1]\,,\ a_{10} = [1]$

 # No jealousy

Results of the linear program:

$z = [1]$

$b = [1.1, 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.7] + 0.1 \cdot [1]$

$= [0.8]$

Iteration 9:

2)ϵ $-$ *budget perturbation*:

$k = [1, 1]$

$b_{00} = [1.1]$

$b_{01} = [1]$

$a_{00} = [1]$, $a_{10} = [1]$

 # No jealousy

Results of the linear program:

$z = [1]$

$b = [1.1, 1]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.8] + 0.1 \cdot [1]$

$= [0.9]$

Iteration 10:

2)ϵ $-$ *budget perturbation*:

$k = [1, 2]$

$b_{00} = [1.1]$

$b_{01} = [0.8, 1]$

$a_{00} = [1]$, $a_{10} = [0]$, $a_{11} = [1]$

 # No jealousy

Results of the linear program:

$z = [0]$

$b = [1.1, 0.8]$

3) $||z||_2 = 0$

4) $update$: $p^* = p = [0.9]$

$b^* = b = [1.1,\ 0.8]$

Algorithm results:

Final course prices: $p^* = [0.9]$

Final budgets : $b^* = [1.1,\ 0.8]$

Final distribution of courses: A: $[1]$

B: $[0]$

Example run 6

$n = 2$ # Number of students

$m = 3$ # Number of courses

parameters:

$\epsilon = 2$

$\delta = 0.5$

$t = 2$

$capacities = [1, 1, 2]$

$utilities = [[5, 4, 1], [4, 6, 3]]$

$b_0 = [5, 4]$

$required = [2, 2]$ # The number of courses each student should take

Running the algorithm:

1) $prices = [0, 0, 0]$
2) $\epsilon - budget\ perturbation$:

   $k = [1, 1]$

   $b_{00} = [5]$

   $b_{01} = [4]$

   $a_{00} = [1, 1, 0]$, $a_{10} = [1, 1, 0]$

   # No jealousy

   Results of the linear program:

   $z = [1, 1, 0]$

   $b = [5, 4]$
3) $||z||_2 = \sqrt{2} \neq 0$
4) $update: p = p + \delta \cdot z$

   $= [0, 0, 0] + 0.5 \cdot [1, 1, 0]$

   $= [0.5, 0.5, 0]$

Iteration 2:

2) $\epsilon - budget\ perturbation$:

   $k = [1, 1]$

$b_{00} = [5]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 1,\ 0]$

 # No jealousy

Results of the linear program:

$z = [1, 1, 0]$

$b = [5, 4]$

3) $||z||_2 = \sqrt{2} \neq 0$

4) $update: p = p + \delta \cdot z$

$= [0.5,\ 0.5,\ 0] + 0.5 \cdot [1, 1, 0]$

$= [1,\ 1,\ 0]$

Iteration 3:

2) $\epsilon - budget\ perturbation$:

$k = [1, 1]$

$b_{00} = [5]$

$b_{01} = [4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [1,\ 1,\ 0]$

 # No jealousy

Results of the linear program:

$z = [1, 1, 0]$

$b = [5, 4]$

3) $||z||_2 = \sqrt{2} \neq 0$

4) $update: p = p + \delta \cdot z$

$= [1,\ 1,\ 0] + 0.5 \cdot [1, 1, 0]$

$= [1.5,\ 1.5,\ 0]$

Iteration 4:

2)$\epsilon - budget\ perturbation$:

$k = [1, 2]$

$b_{00} = [5]$

$b_{01} = [2, 4]$

$a_{00} = [1,\ 1,\ 0]\,,\ a_{10} = [0,\ 1,\ 1], a_{11} = [1,\ 1,\ 0]$

$\#\ No\ jealousy$

$Results\ of\ the\ linear\ program:$

$z = [0, 1, 0]$

$b = [5, 2]$

3) $||z||_2 = 1 \neq 0$

4) $update: p = p + \delta \cdot z$

$= [1.5,\ 1.5,\ 0] + 0.5 \cdot [0, 1, 0]$

$= [1.5,\ 2,\ 0]$

Iteration 5:

2)$\epsilon - budget\ perturbation:$

$k = [2, 2]$

$b_{00} = [3, 5]$

$b_{01} = [2, 3.5]$

$a_{00} = [1,\ 0,\ 1], a_{01} = [1,\ 1,\ 0], a_{10} = [0,\ 1,\ 1], a_{11} = [1,\ 1,\ 0]$

$Jealousy\ exists:\ u_0(a_{00}) < u_0(a_{11})$

$Results\ of\ the\ linear\ program:$

$z = [0, 0, 0]$

$b = [3, 2]$

3) $||z||_2 = 0$

4) $update: p^* = p = [1.5,\ 2,\ 0]$

$b^* = b = [3, 2]$

Algorithm results:

Final course prices: $p^* = [1.5,\ 2,\ 0]$

Final budgets : $b^* = [3, 2]$

Final distribution of courses: A: $[1,\ 0,\ 1]$

B: $[0,\ 1,\ 1]$

Example run 7

obvious cases:

 # Case 1:

$n = 100$ # Number of students

$m = 500$ # Number of courses

$capacities = [200, 200, \dots, 200]$

$\Rightarrow$ *Each student will get all the courses*

 # Case 2:

$n = 100$ # Number of students

$m = 100$ # Number of courses

$capacities = [1, 1, 1, \dots, 1]$

$\forall student\ i \in [n]$: $utilities = only\ course\ i$

$\Rightarrow$ *Each student i will get course i*

 # Case 3:

$n = 100$ # Number of students

$m = 100$ # Number of courses

$capacities = [1, 1, 1, \dots, 1]$

$\forall student\ i \in [n]$: $u_i = [100, 99, 98, \dots, 1]$

$b_0 = [100, 99, 98, \dots, 1]$

$\Rightarrow$ *Each student i will get course i, because student i have the highest i budget.*

 # Case 4:

$n = 100$ # Number of students

$m = 300$ # Number of courses

$capacities = [200, 200, \dots, 200]$

$required = 3$ # The number of courses each student should take

$\Rightarrow$ *Each student will get his* 3 *favorite courses*

Algorithm 2 – Running examples

Example run 1

$M = algo\,1$

$criteria\ for\ profitable\ manipulation = resampled\ randomness\ (\boldsymbol{u}_{-i}, \boldsymbol{c}, R)$

$u_{-i} = \{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\}\}$

$u_i = \{moti: \{x: 1,\ y: 2,\ z: 4\}\}$

$c = \{x: 1,\ y: 2,\ z: 3\}$ $\quad$ $\#\ item\ capacity$

$agent\_capacities = 2$

$\varepsilon = 0.5$

$\delta = 0.5$

$t = 0$ $\quad$ $\rightarrow NO\_EF\_TB$

$\eta = 2$

$\beta = 4$

Running the algorithm:

1) $v_0 = \{x: 1,\ y: 2,\ z: 4\}$

2) $v_0 = (w = \{x: 1,\ y: 2,\ z: 4\},\ valid = 1,\ req = 1)$

3) $\eta = 2$

$V = \{\{x: 2,\ y: 2,\ z: 4\}, \{x: \frac{1}{2},\ y: 2,\ z: 4\},$

$\{x: 1,\ y: 4,\ z: 4\}, \{x: 1,\ y: 1,\ z: 4\},$

$\{x: 1,\ y: 2,\ z: 8\}, \{x: 1,\ y: 2,\ z: 2\}\}$

4) $\beta = 4$

$r \sim U(2,\ 4)^3$

$\#\ the\ original\ utility$

run for $\{x: 1,\ y: 2,\ z: 4\}$

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: $avi: \{x, z\},\ beni: \{y, z\},\ moti: \{y, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: $avi: \{y, z\},\ beni: \{x, y\},\ moti: \{z\}$

$u_{moti} = 4$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$u_{moti} = 6$

$E[u_{moti}(M_1($

$\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 2,\ z: 4\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$

run for $\{x: \frac{1}{2},\ y: 2,\ z: 4\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: $avi: \{x, z\},\ beni: \{y, z\},\ moti: \{y, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: $avi: \{y, z\},\ beni: \{x, y\},\ moti: \{z\}$

$u_{moti} = 4$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$$E[u_{moti}(M_1(\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 2,\ y: 2,\ z: 4\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$$

run for $\{x: 1,\ y: 4,\ z: 4\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: $avi: \{x, z\}$, $beni: \{y, z\}$, $moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: $avi: \{y, z\}$, $beni: \{x, y\}$, $moti: \{z\}$

$$u_{moti} = 4$$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\}$, $beni: \{x, z\}$, $moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: $avi: \{y, z\}$, $beni: \{x, z\}$, $moti: \{y, z\}$

$$u_{moti} = 6$$

$$E[u_{moti}(M_1(\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 4,\ z: 4\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$$

run for $\{x: 1,\ y: 1,\ z: 4\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: $avi: \{y, z\}$, $beni: \{y, z\}$, $moti: \{x, z\}$

$$u_{moti} = 5$$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: $avi: \{y, z\}$, $beni: \{x, y\}$, $moti: \{z\}$

$$u_{moti} = 4$$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$$u_{moti} = 5$$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$$u_{moti} = 5$$

$E[u_{moti}(M_1($
$\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 1,\ z: 4\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 4.75$

run for $\{x: 1,\ y: 2,\ z: 8\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: $avi: \{x, z\},\ beni: \{y, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: $avi: \{y, z\},\ beni: \{x, y\},\ moti: \{z\}$

$$u_{moti} = 4$$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$E[u_{moti}(M_1($

$\{avi: \{x: 3, y: 5, z: 1\}, beni: \{x: 2, y: 3, z: 1\}, moti: \{x: 1, y: 2, z: 2\}\}, c = \{x: 1, y: 2, z: 3\}, r))] = 5.5$

$\rightarrow arg\ max_{v \in V \cup \{v_0\}} = \{x: 1, y: 2, z: 4\}$

5) $v^* = v_0$ *# terminate with v_0 as the best manipulation found*

Example run 2

$M = algo\,1$

$criteria\ for\ profitable\ manipulation = resampled\ randomness\ (U_{-i}, \boldsymbol{c}, R)$

$u_{-i} = \{avi: \{x: 3, y: 5, z: 1\}, beni: \{x: 2, y: 3, z: 1\}\}$

$u_i = \{moti: \{x: 1, y: 2, z: 4\}\}$

$c = \{x: 1, y: 2, z: 3\}$ $\quad$ $\#\ item\ capacity$

$agent\_capacities = 2$

$\varepsilon = 0.5$

$\delta = 0.5$

$t = 1$ $\quad$ $\rightarrow EF - TB$

Running the algorithm:

1) $v_0 = \{x: 1, y: 2, z: 4\}$

2) $v_0 = (w = \{x: 1, y: 2, z: 4\}, valid = 1, req = 1)$

3) $\eta = 2$

$V = \{\{x: 2, y: 2, z: 4\}, \{x: \frac{1}{2}, y: 2, z: 4\},$

$\{x: 1, y: 4, z: 4\}, \{x: 1, y: 1, z: 4\},$

$\{x: 1, y: 2, z: 8\}, \{x: 1, y: 2, z: 2\}\}$

4) $\beta = 4$

$r \sim U(2, 4)^3$

run for $\{x: 1, y: 2, z: 4\}$**:** $\#\ the\ original$

$b_0 = \{avi: 2, beni: 3, moti: 4\}$

answer 1: $avi: \{x, z\}, beni: \{y, z\}, moti: \{y, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 3, beni: 4, moti: 2\}$

answer 2: $avi: \{y, z\}, beni: \{x, y\}, moti: \{z\}$

$$u_{moti} = 4$$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$\Xi[u_{moti}(M_1($

$\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 2,\ z: 4\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$

---

run for $\{x: \frac{1}{2},\ y: 2,\ z: 4\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: $avi: \{x, z\},\ beni: \{y, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: $avi: \{y, z\},\ beni: \{x, y\},\ moti: \{z\}$

$$u_{moti} = 4$$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: avi: $\{y, z\}$, $beni$: $\{x, z\}$, $moti$: $\{y, z\}$

$$u_{moti} = 6$$

$$\Xi[u_{moti}(M_1($$

$$\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 2,\ y: 2,\ z: 4\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$$

run for $\{x: 1,\ y: 4,\ z: 4\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: avi: $\{x, z\}$, $beni$: $\{y, z\}$, $moti$: $\{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: avi: $\{y, z\}$, $beni$: $\{x, y\}$, $moti$: $\{z\}$

$$u_{moti} = 4$$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: avi: $\{y, z\}$, $beni$: $\{x, z\}$, $moti$: $\{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: avi: $\{y, z\}$, $beni$: $\{x, z\}$, $moti$: $\{y, z\}$

$$u_{moti} = 6$$

$$\Xi[u_{moti}(M_1($$

$$\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 4,\ z: 4\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$$

---

run for $\{x: 1,\ y: 1,\ z: 4\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: avi: $\{y, z\}$, $beni$: $\{y, z\}$, $moti$: $\{x, z\}$

$$u_{moti} = 5$$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: $avi: \{y, z\},\ beni: \{x, y\},\ moti: \{z\}$

$$u_{moti} = 4$$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$$u_{moti} = 5$$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$$u_{moti} = 5$$

$\Xi[u_{moti}(M_1($

$\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 1,\ z: 4\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 4.75$

---

run for $\{x: 1,\ y: 2,\ z: 8\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: $avi: \{x, z\},\ beni: \{y, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: $avi: \{y, z\},\ beni: \{x, y\},\ moti: \{z\}$

$$u_{moti} = 4$$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$\Xi[u_{moti}(M_1($

$\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 2,\ z: 8\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$

---

run for $\{x: 1,\ y: 2,\ z: 2\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 4\}$

answer 1: $avi: \{x, z\},\ beni: \{y, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 3,\ beni: 4,\ moti: 2\}$

answer 2: $avi: \{y, z\},\ beni: \{x, y\},\ moti: \{z\}$

$$u_{moti} = 4$$

$b_0 = \{avi: 5,\ beni: 3,\ moti: 4\}$

answer 3: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 4,\ beni: 3,\ moti: 5\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{y, z\}$

$$u_{moti} = 6$$

$\Xi[u_{moti}(M_1($

$\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 2,\ z: 2\}\},\ c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$

$\rightarrow arg\ max_{v \in V \cup \{v_0\}} = \{x: 1,\ y: 2,\ z: 4\}$

5) $v^* = v_0$ *terminate with* v_0 *as the best manipulation found*

Example run 3 – Article 5.2

$M = algo\,1$

$criteria\ for\ profitable\ manipulation = resampled\ population\ (U_{-i}, \boldsymbol{c}, R)$

$u_{-i} = \{bob\colon \{CS161\colon 3,\ ECON101\colon 2,\ IR\colon 0\},\ eve(1-10)\colon \{CS161\colon 1,\ ECON101\colon 10,\ IR\colon 0\}\}$

$u_i = \{alice\colon \{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 6\}\}$

$c = \{CS161\colon 1,\ ECON101\colon 10,\ IR\colon \infty\}$ $\quad$ $\#\ item\ capacity$

$agent\_capacities = 2$

$initial\_budget = \{alice\colon 2,\ bob\colon 5,\ eve\colon [6,\ 1,\ 1..]\}$ $\quad$ $\#\ \beta = 16 \rightarrow r \sim U(5,\ 13)^3$

$\varepsilon = 0.5$

$\delta = 0.5$

$t = 1$ $\quad$ $\rightarrow EF - TB$

Running the algorithm:

1) $v_0 = \{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 6\}$
2) $v_0 = (w = \{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 6\})$
3) $\eta = 2$

   $V = \{\{CS161\colon 2\frac{1}{2},\ ECON101\colon 3,\ IR\colon 6\}, \{CS161\colon 10,\ ECON101\colon 3,\ IR\colon 6\},$

   $\{CS161\colon 5,\ ECON101\colon 1\frac{1}{2},\ IR\colon 6\}, \{CS161\colon 5,\ ECON101\colon 6,\ IR\colon 6\},$

   $\{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 3\}, \{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 12\}\}$

4)

run for $\{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 6\}$**:** *# the original*

answer: $alice\colon \{IR\},\ bob\colon \{CS161,\ IR\},\ eve\colon \{ECON101,\ IR\}$

$u_{alice} = 6$

$E[u_{alice}(M_1(\{alice: \{CS161: 5,\ ECON101: 3,\ IR: 6\},$

$bob: \{CS161: 3,\ ECON101: 2,\ IR: 0\},$

$eve: \{CS161: 1,\ ECON101: 10,\ IR: 0\}\},$

$c = \{CS161: 1,\ ECON101: 1,\ IR: \infty\},\ r))] = 6$

Misreports:

run for $\{CS161: 2\frac{1}{2},\ ECON101: 3,\ IR: 6\}$**:**

answer: $alice: \{IR\}$, $bob: \{CS161,\ IR\}$, $eve: \{ECON101,\ IR\}$

$u_{alice} = 6$

$E[u_{alice}(M_1(\{alice: \{CS161: 2\frac{1}{2},\ ECON101: 3,\ IR: 6\},$

$bob: \{CS161: 3,\ ECON101: 5,\ IR: 1\},$

$eve: \{CS161: 1,\ ECON101: 10,\ IR: 5\}\},$

$c = \{CS161: 1,\ ECON101: 1,\ IR: \infty\},\ r))] = 6$

---

run for $\{CS161: 10,\ ECON101: 3,\ IR: 6\}$**:**

answer: $alice: \{IR\}$, $bob: \{CS161,\ IR\}$, $eve: \{ECON101,\ IR\}$

$u_{alice} = 6$

$E[u_{alice}(M_1(\{alice: \{CS161: 10,\ ECON101: 3,\ IR: 6\},$

$bob: \{CS161: 3,\ ECON101: 5,\ IR: 1\},$

$eve: \{CS161: 1,\ ECON101: 10,\ IR: 5\}\},$

$c = \{CS161: 1,\ ECON101: 1,\ IR: \infty\},\ r))] = 6$

---

run for $\{CS161: 5,\ ECON101: 1\frac{1}{2},\ IR: 6\}$**:**

answer: $alice: \{IR\}$, $bob: \{CS161,\ IR\}$, $eve: \{ECON101,\ IR\}$

$u_{alice} = 6$

$$E[u_{alice}(M_1(\{alice\colon \{CS161\colon 5,\ ECON101\colon 1\tfrac{1}{2},\ IR\colon 6\},$$
$$bob\colon \{CS161\colon 3,\ ECON101\colon 5,\ IR\colon 1\},$$
$$eve\colon \{CS161\colon 1,\ ECON101\colon 10,\ IR\colon 5\}\},$$
$$c = \{CS161\colon 1,\ ECON101\colon 1,\ IR\colon \infty\},\ r))] = 6$$

---

run for $\{CS161\colon 5,\ ECON101\colon 6,\ IR\colon 6\}$**:**

answer: $alice\colon \{IR\}$, $bob\colon \{CS161,\ IR\}$, $eve\colon \{ECON101,\ IR\}$

$$u_{alice} = 6$$

$$E[u_{alice}(M_1(\{alice\colon \{CS161\colon 5,\ ECON101\colon 6,\ IR\colon 6\},$$
$$bob\colon \{CS161\colon 3,\ ECON101\colon 5,\ IR\colon 1\},$$
$$eve\colon \{CS161\colon 1,\ ECON101\colon 10,\ IR\colon 5\}\},$$
$$c = \{CS161\colon 1,\ ECON101\colon 1,\ IR\colon \infty\},\ r))] = 6$$

---

run for $\{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 3\}$**:**

answer: $alice\colon \{CS161,\ IR\}$, $bob\colon \{IR\}$, $eve\colon \{ECON101,\ IR\}$

$$u_{alice} = 11$$

$$E[u_{alice}(M_1(\{alice\colon \{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 3\},$$
$$bob\colon \{CS161\colon 3,\ ECON101\colon 5,\ IR\colon 1\},$$
$$eve\colon \{CS161\colon 1,\ ECON101\colon 10,\ IR\colon 5\}\},$$
$$c = \{CS161\colon 1,\ ECON101\colon 1,\ IR\colon \infty\},\ r))] = 11$$

---

run for $\{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 12\}$**:**

answer: $alice\colon \{IR\}$, $bob\colon \{IR\}$, $eve\colon \{ECON101,\ CS161\}$

$$u_{alice} = 6$$

$$E[u_{alice}(M_1(\{alice\colon \{CS161\colon 5,\ ECON101\colon 3,\ IR\colon 12\},$$
$$bob\colon \{CS161\colon 3,\ ECON101\colon 5,\ IR\colon 1\},$$
$$eve\colon \{CS161\colon 1,\ ECON101\colon 10,\ IR\colon 5\}\},$$
$$c = \{CS161\colon 1,\ ECON101\colon 1,\ IR\colon \infty\},\ r))] = 6$$

$\rightarrow arg\,max_{v \in V \cup \{v_0\}} = \{CS161: 10,\ ECON101: 3,\ IR: 6\}$

5) $v\ ^* \neq v_0$

6) $v_0 = \{CS161: 10,\ ECON101: 3,\ IR: 6\}$

Iteration 2:

2) $v_0 = (w = \{CS161: 10,\ ECON101: 3,\ IR: 6\})$

3) $\eta = 2$

$V = \{\{CS161: 20,\ ECON101: 3,\ IR: 6\}, \{CS161: 5,\ ECON101: 3,\ IR: 6\},$

$\{CS161: 10,\ ECON101: 1\frac{1}{2},\ IR: 6\}, \{CS161: 10,\ ECON101: 6,\ IR: 6\},$

$\{CS161: 10,\ ECON101: 3,\ IR: 3\}, \{CS161: 10,\ ECON101: 3,\ IR: 12\}\}$

4)

run for $\{CS161: 5,\ ECON101: 3,\ IR: 6\}$**:** *# the original*

answer: $alice: \{IR\}$, $bob: \{CS161,\ IR\}$, $eve: \{ECON101,\ IR\}$

$u_{alice} = 6$

$E[u_{alice}(M_1(\{alice: \{CS161: 5,\ ECON101: 3,\ IR: 6\},$

$bob: \{CS161: 3,\ ECON101: 2,\ IR: 0\},$

$eve: \{CS161: 1,\ ECON101: 10,\ IR: 0\}\},$

$c = \{CS161: 1,\ ECON101: 1,\ IR: \infty\},\ r))] = 6$

Misreports:

run for $\{CS161: 2\frac{1}{2},\ ECON101: 3,\ IR: 6\}$**:**

answer: $alice: \{IR\}$, $bob: \{CS161,\ IR\}$, $eve: \{ECON101,\ IR\}$

$u_{alice} = 6$

$E[u_{alice}(M_1(\{alice: \{CS161: 2\frac{1}{2}, ECON101: 3, IR: 6\},$

$bob: \{CS161: 3, ECON101: 5, IR: 1\},$

$eve: \{CS161: 1, ECON101: 10, IR: 5\}\},$

$c = \{CS161: 1, ECON101: 1, IR: \infty\}, r))] = 6$

---

run for $\{CS161: 10, ECON101: 3, IR: 6\}$**:**

answer: $alice: \{CS161, IR\}$, $bob: \{IR\}$, $eve: \{ECON101, IR\}$

$u_{alice} = 11$

$E[u_{alice}(M_1(\{alice: \{CS161: 10, ECON101: 3, IR: 6\},$

$bob: \{CS161: 3, ECON101: 5, IR: 1\},$

$eve: \{CS161: 1, ECON101: 10, IR: 5\}\},$

$c = \{CS161: 1, ECON101: 1, IR: \infty\}, r))] = 11$

---

run for $\{CS161: 5, ECON101: 1\frac{1}{2}, IR: 6\}$**:**

answer: $alice: \{CS161, IR\}$, $bob: \{IR\}$, $eve: \{ECON101, IR\}$

$u_{alice} = 11$

$E[u_{alice}(M_1(\{alice: \{CS161: 5, ECON101: 1\frac{1}{2}, IR: 6\},$

$bob: \{CS161: 3, ECON101: 5, IR: 1\},$

$eve: \{CS161: 1, ECON101: 10, IR: 5\}\},$

$c = \{CS161: 1, ECON101: 1, IR: \infty\}, r))] = 11$

---

run for $\{CS161: 5, ECON101: 6, IR: 6\}$**:**

answer: $alice: \{IR\}$, $bob: \{CS161, IR\}$, $eve: \{ECON101, IR\}$

$u_{alice} = 6$

$E[u_{alice}(M_1(\{alice: \{CS161: 5, ECON101: 6, IR: 6\},$

$bob: \{CS161: 3, ECON101: 5, IR: 1\},$

$eve: \{CS161: 1,\ ECON101: 10,\ IR: 5\}\},$

$c = \{CS161: 1,\ ECON101: 1,\ IR: \infty\},\ r))] = 6$

---

run for $\{CS161: 5,\ ECON101: 3,\ IR: 3\}$**:**

answer: $alice: \{CS161,\ IR\}$, $bob: \{IR\}$, $eve: \{ECON101,\ IR\}$

$u_{alice} = 11$

$E[u_{alice}(M_1(\{alice: \{CS161: 5,\ ECON101: 3,\ IR: 3\},$

$bob: \{CS161: 3,\ ECON101: 5,\ IR: 1\},$

$eve: \{CS161: 1,\ ECON101: 10,\ IR: 5\}\},$

$c = \{CS161: 1,\ ECON101: 1,\ IR: \infty\},\ r))] = 11$

---

run for $\{CS161: 5,\ ECON101: 3,\ IR: 12\}$**:**

answer: $alice: \{IR\}$, $bob: \{IR\}$, $eve: \{ECON101,\ CS161\}$

$u_{alice} = 6$

$E[u_{alice}(M_1(\{alice: \{CS161: 5,\ ECON101: 3,\ IR: 12\},$

$bob: \{CS161: 3,\ ECON101: 5,\ IR: 1\},$

$eve: \{CS161: 1,\ ECON101: 10,\ IR: 5\}\},$

$c = \{CS161: 1,\ ECON101: 1,\ IR: \infty\},\ r))] = 6$

$\rightarrow arg\,max_{v \in V \cup \{v_0\}} = \{CS161: 10,\ ECON101: 3,\ IR: 6\}$

5) $v\ ^* \neq v_0$

6) $v_0 = \{CS161: 10,\ ECON101: 3,\ IR: 6\}$

Example run 4

$M = algo\,1$

$criteria\ for\ profitable\ manipulation = resampled\ randomness\ (\boldsymbol{u}_{-i}, \boldsymbol{c}, R)$

$u_{-i} = \{avi: \{x: 5, y: 3\}\}$

$u_i = \{moti: \{x: 6, y: 2\}\}$

$c = \{x: 1, y: 2\}$ $\quad$ $\#\ item\ capacity$

$agent\_capacities = 2$

$\varepsilon = 0.5$

$\delta = 0.5$

$t = 0$ $\quad$ $\rightarrow NO\_EF\_TB$

Running the algorithm:

1) $v_0 = \{x: 6, y: 2\}$

2) $v_0 = (w = \{x: 6, y: 2\})$

3) $\eta = 2$

$V = \{\{x: 3, y: 2\}, \{x: 12, y: 2\},$
$\{x: 6, y: 1\}, \{x: 6, y: 4\}\}$

4) $\beta = 4$

$r \sim U(2, 4)^3$

$\#\ the\ original\ utility$

run for $\{x: 6, y: 2\}$

$b_0 = \{avi: 2, moti: 1\}$

answer 1: $avi: \{x, y\},\ moti: \{y\}$

$u_{moti} = 2$

$b_0 = \{avi: 2, moti: 3\}$

answer 2: $avi: \{y\},\ moti: \{x, y\}$

$u_{moti} = 8$

$b_0 = \{avi: 2, moti: 4\}$

answer 3: $avi: \{y\}, moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 3, moti: 2\}$

answer 4: $avi: \{x, y\}, moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 3, moti: 7\}$

answer 5: $avi: \{y\}, moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 3, moti: 4\}$

answer 6: $avi: \{y\}, moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 4, moti: 2\}$

answer 7: $avi: \{x, y\}, moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 4, moti: 1\}$

answer 8: $avi: \{x, y\}, moti: \{y\}$

$$u_{moti} = 2$$

$E[u_{alice}(M_1(\{avi: \{x: 5, y: 3\}, moti: \{x: 6, y: 2\}\}, r))] = 5$

---

run for $\{x: 3, y: 2\}$

$b_0 = \{avi: 2, moti: 1\}$

answer 1: $avi: \{x, y\}, moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 2, moti: 3\}$

answer 2: $avi: \{y\}$, $moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 2, moti: 4\}$

answer 3: $avi: \{y\}$, $moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 3,\ moti: 2\}$

answer 4: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 3,\ moti: 7\}$

answer 5: $avi: \{y\},\ moti: \{x,\ y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 3,\ moti: 4\}$

answer 6: $avi: \{y\}$, $moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 4,\ moti: 2\}$

answer 7: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 4,\ moti: 1\}$

answer 8: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$E[u_{alice}(M_1(\{avi: \{x: 5,\ y: 3\},\ moti: \{x: 3,\ y: 2\}\},\ r))] = 5$

---

run for $\{x: 12,\ y: 2\}$

$b_0 = \{avi: 2, moti: 1\}$

answer 1: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 2, moti: 3\}$

answer 2: $avi: \{y\},\ moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 2, moti: 4\}$

answer 3: $avi: \{y\},\ moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 3,\ moti: 2\}$

answer 4: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 3,\ moti: 7\}$

answer 5: $avi: \{y\},\ moti: \{x,\ y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 3,\ moti: 4\}$

answer 6: $avi: \{y\},\ moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 4,\ moti: 2\}$

answer 7: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 4,\ moti: 1\}$

answer 8: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$E[u_{alice}(M_1(\{avi:\ \{x:5,\ y:3\},\ moti:\ \{x:12,\ y:2\}\},\ r))] = 5$

---

run for $\{x:6,\ y:4\}$

$b_0 = \{avi:2, moti:1\}$

answer 1: $avi:\{x,y\},\ moti:\{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi:2, moti:3\}$

answer 2: $avi:\{\ y\}\ ,\ moti:\{x,y\}$

$$u_{moti} = 8$$

$b_0 = \{avi:2, moti:4\}$

answer 3: $avi:\{\ y\}\ ,\ moti:\{x,y\}$

$$u_{moti} = 8$$

$b_0 = \{avi:3,\ moti:2\}$

answer 4: $avi:\{x,y\},\ moti:\{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi:3,\ moti:7\}$

answer 5: $avi:\{y\},\ moti:\{x,\ y\}$

$$u_{moti} = 8$$

$b_0 = \{avi:3,\ moti:4\}$

answer 6: $avi:\{\ y\}\ ,\ moti:\{x,y\}$

$$u_{moti} = 8$$

$b_0 = \{avi:4,\ moti:2\}$

answer 7: $avi:\{x,y\},\ moti:\{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi:4,\ moti:1\}$

answer 8: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$E[u_{alice}(M_1(\{avi: \{x: 5,\ y: 3\},\ moti: \{x: 6,\ y: 4\}\},\ r))] = 5$

---

run for $\{x: 6,\ y: 1\}$

$b_0 = \{avi: 2, moti: 1\}$

answer 1: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 2, moti: 3\}$

answer 2: $avi: \{y\},\ moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 2, moti: 4\}$

answer 3: $avi: \{y\},\ moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 3,\ moti: 2\}$

answer 4: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 3,\ moti: 7\}$

answer 5: $avi: \{y\},\ moti: \{x,\ y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 3,\ moti: 4\}$

answer 6: $avi: \{y\},\ moti: \{x, y\}$

$$u_{moti} = 8$$

$b_0 = \{avi: 4,\ moti: 2\}$

answer 7: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$b_0 = \{avi: 4,\ moti: 1\}$

answer 8: $avi: \{x, y\},\ moti: \{y\}$

$$u_{moti} = 2$$

$E[u_{alice}(M_1(\{avi: \{x: 5,\ y: 3\},\ moti: \{x: 6,\ y: 1\}\},\ r))] = 5$

$\rightarrow arg\ max_{v \in V \cup \{v_0\}} = \{x: 6,\ y: 2\}$

5) $v\ ^* = v_0$ *terminate with v_0 as the best manipulation found*

Example run 5

$M = algo\,1$

$criteria\ for\ profitable\ manipulation = population\ (U_{-i}, \boldsymbol{c}, R)$

$u_i = \{moti: \{x: 1, y: 2, z: 5\}\}$

$c = \{x: 1, y: 2, z: 3\}$ $\quad$ $\#\ item\ capacity$

$agent\_capacities = 2$

$\varepsilon = 0.5$

$\delta = 0.5$

$t = 0$ $\quad$ $\rightarrow NO\_EF\_TB$

Running the algorithm:

1) $v_0 = \{x: 1, y: 2, z: 5\}$

2) $v_0 = (w = \{x: 1, y: 2, z: 5\}, valid = 1, req = 1)$

3) $\eta = 2$

$V = \{\{x: 2, y: 2, z: 5\}, \{x: \frac{1}{2}, y: 2, z: 5\},$

$\{x: 1, y: 4, z: 5\}, \{x: 1, y: 1, z: 5\},$

$\{x: 1, y: 2, z: 10\}, \{x: 1, y: 2, z: 2.5\}\}$

4) $\beta = 4$

$r \sim U(2, 4)^3$

 # the original utility

run for $\{x: 1, y: 2, z: 5\}$

$b_0 = \{avi: 2, beni: 3, moti: 1\}$

$u_{-i} = \{avi: \{x: 3, y: 6, z: 4\}, beni: \{x: 2, y: 4, z: 1\}\}$

answer 1: $avi: \{y, z\}, beni: \{x, z\}, moti: \{z\}$

$u_{moti} = 5$

$b_0 = \{avi: 6, beni: 5, moti: 2\}$

$u_{-i} = \{avi: \{x: 2, y: 6, z: 3\}, beni: \{x: 2, y: 7, z: 4\}\}$

answer 2: $avi: \{y, z\}, beni: \{y, z\}, moti: \{x, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 3,\ beni: 2,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 4,\ y: 5,\ z: 6\},\ beni: \{x: 5,\ y: 6,\ z: 7\}\}$

answer 3: $avi: \{y, z\}$, $beni: \{y, z\}$, $moti: \{x, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 5,\ beni: 7,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 3,\ y: 3,\ z: 1\},\ beni: \{x: 4,\ y: 5,\ z: 7\}\}$

answer 4: $avi: \{y, z\}$, $beni: \{x, z\}$, $moti: \{z\}$

$$u_{moti} = 5$$

$E[u_{moti}(M_1(\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 2,\ z: 4\}\},$

$c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$

---

run for $\{x: 2,\ y: 2,\ z: 5\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 3,\ y: 6,\ z: 4\},\ beni: \{x: 2,\ y: 4,\ z: 1\}\}$

answer 1: $avi: \{y, z\}$, $beni: \{x, y\}$, $moti: \{z\}$

$$u_{moti} = 5$$

$b_0 = \{avi: 6,\ beni: 5,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 2,\ y: 6,\ z: 3\},\ beni: \{x: 2,\ y: 7,\ z: 4\}\}$

answer 2: $avi: \{y, z\}$, $beni: \{y, z\}$, $moti: \{x, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 3,\ beni: 2,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 4,\ y: 5,\ z: 6\},\ beni: \{x: 5,\ y: 6,\ z: 7\}\}$

answer 3: $avi: \{y, z\}$, $beni: \{y, z\}$, $moti: \{x, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 5,\ beni: 7,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 3,\ y: 3,\ z: 1\},\ beni: \{x: 4,\ y: 5,\ z: 7\}\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{z\}$

$u_{moti} = 5$

$E[u_{moti}(M_1(\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 2,\ y: 2,\ z: 4\}\},$

$c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$

---

run for $\{x: \frac{1}{2},\ y: 2,\ z: 5\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 3,\ y: 6,\ z: 4\},\ beni: \{x: 2,\ y: 4,\ z: 1\}\}$

answer 1: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{z\}$

$u_{moti} = 5$

$b_0 = \{avi: 6,\ beni: 5,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 2,\ y: 6,\ z: 3\},\ beni: \{x: 2,\ y: 7,\ z: 4\}\}$

answer 2: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 3,\ beni: 2,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 4,\ y: 5,\ z: 6\},\ beni: \{x: 5,\ y: 6,\ z: 7\}\}$

answer 3: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 5,\ beni: 7,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 3,\ y: 3,\ z: 1\},\ beni: \{x: 4,\ y: 5,\ z: 7\}\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{z\}$

$u_{moti} = 5$

$E[u_{moti}(M_1(\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 4,\ z: 4\}\},$

$c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$

---

run for $\{x\colon 1,\ y\colon 1,\ z\colon 5\}$**:**

$b_0 = \{avi\colon 2,\ beni\colon 3,\ moti\colon 1\}$

$u_{-i} = \{avi\colon \{x\colon 3,\ y\colon 6,\ z\colon 4\},\ beni\colon \{x\colon 2,\ y\colon 4,\ z\colon 1\}\}$

answer 1: $avi\colon \{y, z\},\ beni\colon \{x, z\},\ moti\colon \{z\}$

$u_{moti} = 5$

$b_0 = \{avi\colon 6,\ beni\colon 5,\ moti\colon 2\}$

$u_{-i} = \{avi\colon \{x\colon 2,\ y\colon 6,\ z\colon 3\},\ beni\colon \{x\colon 2,\ y\colon 7,\ z\colon 4\}\}$

answer 2: $avi\colon \{y, z\},\ beni\colon \{y, z\},\ moti\colon \{x, z\}$

$u_{moti} = 6$

$b_0 = \{avi\colon 3,\ beni\colon 2,\ moti\colon 1\}$

$u_{-i} = \{avi\colon \{x\colon 4,\ y\colon 5,\ z\colon 6\},\ beni\colon \{x\colon 5,\ y\colon 6,\ z\colon 7\}\}$

answer 3: $avi\colon \{y, z\},\ beni\colon \{y, z\},\ moti\colon \{x, z\}$

$u_{moti} = 6$

$b_0 = \{avi\colon 5,\ beni\colon 7,\ moti\colon 2\}$

$u_{-i} = \{avi\colon \{x\colon 3,\ y\colon 3,\ z\colon 1\},\ beni\colon \{x\colon 4,\ y\colon 5,\ z\colon 7\}\}$

answer 4: $avi\colon \{y, z\},\ beni\colon \{x, z\},\ moti\colon \{z\}$

$u_{moti} = 5$

$E[u_{moti}(M_1(\{avi\colon \{x\colon 3,\ y\colon 5,\ z\colon 1\},\ beni\colon \{x\colon 2,\ y\colon 3,\ z\colon 1\},\ moti\colon \{x\colon 1,\ y\colon 1,\ z\colon 4\}\},$

$c = \{x\colon 1,\ y\colon 2,\ z\colon 3\},\ r))] = 5.5$

---

run for $\{x\colon 1,\ y\colon 4,\ z\colon 5\}$**:**

$b_0 = \{avi\colon 2,\ beni\colon 3,\ moti\colon 1\}$

$u_{-i} = \{avi\colon \{x\colon 3,\ y\colon 6,\ z\colon 4\},\ beni\colon \{x\colon 2,\ y\colon 4,\ z\colon 1\}\}$

answer 1: $avi\colon \{y, z\},\ beni\colon \{x, z\},\ moti\colon \{z\}$

$u_{moti} = 5$

$b_0 = \{avi\colon 6,\ beni\colon 5,\ moti\colon 2\}$

$u_{-i} = \{avi: \{x: 2,\ y: 6,\ z: 3\},\ beni: \{x: 2,\ y: 7,\ z: 4\}\}$

answer 2: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 3,\ beni: 2,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 4,\ y: 5,\ z: 6\},\ beni: \{x: 5,\ y: 6,\ z: 7\}\}$

answer 3: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 5,\ beni: 7,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 3,\ y: 3,\ z: 1\},\ beni: \{x: 4,\ y: 5,\ z: 7\}\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{z\}$

$u_{moti} = 5$

$E[u_{moti}(M_1(\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 2,\ z: 8\}\},$

$c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$

---

run for $\{x: 1,\ y: 2,\ z: 2.5\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 3,\ y: 6,\ z: 4\},\ beni: \{x: 2,\ y: 4,\ z: 1\}\}$

answer 1: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{z\}$

$u_{moti} = 5$

$b_0 = \{avi: 6,\ beni: 5,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 2,\ y: 6,\ z: 3\},\ beni: \{x: 2,\ y: 7,\ z: 4\}\}$

answer 2: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$u_{moti} = 6$

$b_0 = \{avi: 3,\ beni: 2,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 4,\ y: 5,\ z: 6\},\ beni: \{x: 5,\ y: 6,\ z: 7\}\}$

answer 3: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 5,\ beni: 7,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 3,\ y: 3,\ z: 1\},\ beni: \{x: 4,\ y: 5,\ z: 7\}\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{z\}$

$$u_{moti} = 5$$

$E[u_{moti}(M_1(\{avi: \{x: 3,\ y: 5,\ z: 1\},\ beni: \{x: 2,\ y: 3,\ z: 1\},\ moti: \{x: 1,\ y: 2,\ z: 2\}\},$
$c = \{x: 1,\ y: 2,\ z: 3\},\ r))] = 5.5$

---

run for $\{x: 1,\ y: 2,\ z: 10\}$**:**

$b_0 = \{avi: 2,\ beni: 3,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 3,\ y: 6,\ z: 4\},\ beni: \{x: 2,\ y: 4,\ z: 1\}\}$

answer 1: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{z\}$

$$u_{moti} = 5$$

$b_0 = \{avi: 6,\ beni: 5,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 2,\ y: 6,\ z: 3\},\ beni: \{x: 2,\ y: 7,\ z: 4\}\}$

answer 2: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 3,\ beni: 2,\ moti: 1\}$

$u_{-i} = \{avi: \{x: 4,\ y: 5,\ z: 6\},\ beni: \{x: 5,\ y: 6,\ z: 7\}\}$

answer 3: $avi: \{y, z\},\ beni: \{y, z\},\ moti: \{x, z\}$

$$u_{moti} = 6$$

$b_0 = \{avi: 5,\ beni: 7,\ moti: 2\}$

$u_{-i} = \{avi: \{x: 3,\ y: 3,\ z: 1\},\ beni: \{x: 4,\ y: 5,\ z: 7\}\}$

answer 4: $avi: \{y, z\},\ beni: \{x, z\},\ moti: \{z\}$

$$u_{moti} = 5$$

$E[u_{moti}(M_1(\{avi: \{x: 3, y: 5, z: 1\}, beni: \{x: 2, y: 3, z: 1\}, moti: \{x: 1, y: 2, z: 2\}\},$
$c = \{x: 1, y: 2, z: 3\}, r))] = 5.5$

$\rightarrow arg\ max_{v \in V \cup \{v_0\}} = \{x: 1, y: 2, z: 5\}$

5) $v\ ^* = v_0$ *# terminate with v_0 as the best manipulation found*

Algorithm 3 – Running examples

Example run 1

$n = 3$ # Number of students

$m = 3$ # Number of courses

parameter:

$\beta = 4,\ b_0 \in [1, 5]$

$capacities = \{"x": 2, "y": 1, "z": 3\}$

$utilities = \{"ami": \{"x": 3, "y": 4, "z": 2\}, "tami": \{"x": 4, "y": 3, "z": 2\}, "rami": \{"x": 2, "y": 4, "z": 3\}\}$

$initial\ budgets = \{"ami": 5, "tami": 4, "rami": 3\}$

$required = \{"ami": 2, "tami": 2, "rami": 2\}$ # The number of courses each student should take

Running the algorithm:

1) $p \leftarrow uniform(1, 5)^3$

   $p = \{"x": 1, "y": 2, "z": 1\}$

   $H \leftarrow \phi$

2) $z_j(u, c, p, b) = \sum_{i=1}^{n} a_{ij}(u, p, b) - c_j\ ,\ \ a_i(u, p, b) = arg\ max\ u_i(x)$

   $ami([3, 4, 2],\ [1, 2, 1],\ 5) \rightarrow max[1, 1, 0] \rightarrow [x, y]$

   $tami([4, 3, 2],\ [1, 2, 1],\ 4) \rightarrow max[1, 1, 0] \rightarrow [x, y]$

   $rami([2, 4, 3],\ [1, 2, 1],\ 3) \rightarrow max[0, 1, 1] \rightarrow [y, z]$

   $z = \{"x": 0, "y": 2, "z": -2\},\quad ||z||_2 = \sqrt{8}$

3) $H = \{[1, 2, 1],\ [0, 0, 0],\ [1, 0, 0],\ [0, 1, 0],\ [0, 0, 1],\ [1, 1, 0],\ [1, 0, 1],\ [0, 1, 1],\ [1, 1, 1],$
   $[0, 1, 2],\ [0, 2, 1],\ [1, 0, 2],\ [1, 2, 0],\ [2, 0, 1],\ [2, 1, 0],\ [2, 2, 0],\ [2, 2, 1],\ [1, 1, 2],$
   $[2, 1, 1],\ [3, 0, 0],\ [3, 1, 0],\ [3, 0, 1],\ [3, 1, 1],\ [0, 3, 0],\ [1, 3, 0],\ [0, 0, 3],\ [1, 0, 3],$
   $[2, 0, 3],\ [3, 0, 3],\ [4, 0, 3]\}$

   $\#\ p_0 + p_1 \leq 5,\ p_0 + p_1 \leq 4 = b_0(B),\ p_1 + p_2 \leq 3 = b_0(C)$

$N^G(p, \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \Delta \subseteq [0, 1]$

$= \{[1, 2, 1] + \delta \cdot [0, 2, -2]\}$

$= [1, 4, -1] \qquad (\delta = 1)$

$\approx \{"x": 1, "y": 4, "z": 0\}$ # *When the price is negative we will update it to* 0

$N(p) = \{\{"x": 1, "y": 3, "z": 1\}\}$ # *Student C will be forced to change his choice*

Examination:

$p = \{"x": 1, "y": 4, "z": 0\}$

$ami([3, 4, 2], [1, 4, 0], 5) \rightarrow max[1, 1, 0] \rightarrow [x, y]$

$tami([4, 3, 2], [1, 4, 0], 4) \rightarrow max[1, 0, 1] \rightarrow [x, z]$

$rami([2, 4, 3], [1, 4, 0], 3) \rightarrow max[1, 0, 1] \rightarrow [x, z]$

$z = \{"x": 1, "y": 0, "z": -1\}, \quad ||z||_2 = \sqrt{2} \qquad \rightarrow min$

$p = \{"x": 1, "y": 3, "z": 1\}$

$ami([3, 4, 2], [1, 3, 1], 5) \rightarrow max[1, 1, 0] \rightarrow [x, y]$

$tami([4, 3, 2], [1, 3, 1], 4) \rightarrow max[1, 1, 0] \rightarrow [x, y]$

$rami([2, 4, 3], [1, 3, 1], 3) \rightarrow max[1, 0, 1] \rightarrow [x, z]$

$z = \{"x": 1, "y": 1, "z": -2\}, \quad ||z||_2 = \sqrt{6}$

$update: p = \{"x": 1, "y": 4, "z": 0\}$

$update: \bar{z} = \{"x": 1, "y": 0, "z": 0\}$

Iteration 2:

2) $||z||_2 = \sqrt{2} \neq 0$

3) $H = \{[1, 2, 1], [0, 0, 0], [1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 1, 0], [1, 0, 1], [0, 1, 1], [1, 1, 1],$
$[0, 1, 2], [0, 2, 1], [1, 0, 2], [1, 2, 0], [2, 0, 1], [2, 1, 0], [2, 2, 0], [2, 2, 1], [1, 1, 2],$
$[2, 1, 1], [3, 0, 0], [3, 1, 0], [3, 0, 1], [3, 1, 1], [0, 3, 0], [1, 3, 0], [0, 0, 3], [1, 0, 3],$
$[2, 0, 3], [3, 0, 3], [4, 0, 3],$
$[1, 4, 0], [2, 3, 1], [0, 5, 0]\}$
$\# p_0 + p_1 = 5 = b_0(A), p_0 + p_2 \leq 3 = b_0(C)$

$N^G(p, \Delta) = \{p + \delta \cdot z : \delta \in \Delta\}, \quad \Delta \subseteq [0, 1]$

$$= \{[1,4,0] + \delta \cdot [1,0,0]\}$$

$$= \{"x": 2, "y": 4, "z": 0\} \qquad (\delta = 1)$$

$N(p) = \{\{"x": 2, "y": 4, "z": 0\}, \{"x": 3, "y": 4, "z": 0\}\}$ # *Student A will be forced forced to change his choice*

Examination:

$p = \{"x": 2, "y": 4, "z": 0\}$

$ami([3,4,2], [2,4,0], 5) \rightarrow max[0,1,1] \rightarrow [y,z]$

$tami([4,3,2], [2,4,0], 4) \rightarrow max[1,0,1] \rightarrow [x,z]$

$rami([2,4,3], [2,4,0], 3) \rightarrow max[1,0,1] \rightarrow [x,z]$

$z = \{"x": 0, "y": 0, "z": 0\}, \quad ||z||_2 = 0 \qquad \rightarrow min$

$p = \{"x": 3, "y": 4, "z": 0\}$

$ami([3,4,2], [3,4,0], 5) \rightarrow max[0,1,1] \rightarrow [z,y]$

$tami([4,3,2], [3,4,0], 4) \rightarrow max[1,0,1] \rightarrow [x,z]$

$rami([2,4,3], [3,4,0], 3) \rightarrow max[1,0,1] \rightarrow [x,z]$

$z = \{"x": 0, "y": 0, "z": 0\}, \quad ||z||_2 = 0$

$update: p = \{"x": 2, "y": 4, "z": 0\}$

Iteration 4:

2) $||z||_2 = 0 \Rightarrow p^* = p = \{"x": 2, "y": 4, "z": 0\}$

Algorithm results:

Final course prices: $p^* = \{"x": 2, "y": 4, "z": 0\}$

Final distribution of courses: $ami: [0,1,1] \rightarrow [z,y]$

$tami: [1,0,1] \rightarrow [x,z]$

$rami: [1,0,1] \rightarrow [x,z]$

Example run 2

$n = 2$ # Number of students

$m = 4$ # Number of courses

parameter:

$\beta = 9,\ b_0 \in [1, 10]$

$capacities = [1,\ 2,\ 1,\ 2]$

$utilities = [[5, 4, 3, 2],\ [5, 2, 4, 3]]$ [A, B]

$b_0 = [8, 6]$

$required = [3, 3]$ # The number of courses each student should take

Running the algorithm:

1) $p \leftarrow uniform(1, 10)^4$

$p = [1, 2, 3, 4]$

$H \leftarrow \phi$

2) $z_j(u, c, p, b) = \sum_{i=1}^{n} a_{ij}(u, p, b) - c_j\ ,\ \ a_i(u, p, b) = arg\ max\ u_i(x)$

$A([5, 4, 3, 2],\ [1, 2, 3, 4],\ 8) \rightarrow max[1, 1, 1, 0]$

$B([5, 2, 4, 3],\ [1, 2, 3, 4],\ 6) \rightarrow max[1, 1, 1, 0]$

$z = [1, 0, 1, -2]\ ,\qquad ||z||_2 = \sqrt{6}$

3) $H = \{[1, 2, 3, 4],\ [1, 3, 2, 4],\ [2, 1, 3, 4],\ [2, 3, 1, 4],\ [3, 1, 2, 4],\ [3, 2, 1, 4],\ [2, 2, 1, 4],$
$[2, 1, 2, 4],\ [1, 2, 2, 4],\ [2, 2, 2, 4],\ [0, 1, 5, 2],\ [1, 2, 3, 3],\ [0, 2, 3, 4],\ [3, 2, 0, 4],$
$[2, 1, 3, 3],\ [2, 1, 3, 2],\ [1, 1, 3, 4],\ [0, 1, 3, 4],\ [2, 1, 2, 4],\ [2, 1, 1, 4],\ [1, 1, 3, 3],$
$[1, 1, 2, 4],\ [2, 1, 2, 3],\ [2, 0, 3, 4],\ [0, 0, 3, 4],\ [2, 0, 1, 4],\ [2, 0, 3, 2],\ [1, 0, 2, 4],$
$[1, 0, 3, 3],\ [2, 0, 2, 3],\ \ldots\}$

$\#\ p_0 + p_1 + p_2 \leq 6 = b_0(B),\ p_0 + p_2 + p_3 > 6 = b_0(B)$

$N^G(p,\ \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \Delta \subseteq [0,\ 1]$

$= \{[1, 2, 3, 4] + \delta \cdot [1, 0, 1, -2]\}$

$= [2, 2, 4, 2] \qquad (\delta = 1)$

$N(p) = \{[1, 2, 3, 0]\}$ *# For under-demanded courses, we consider setting them to zero*

Examination:

$p = [2, 2, 4, 2]$

$A([5, 4, 3, 2],\ [2, 2, 4, 2],\ 8) \rightarrow max[1, 1, 1, 0]$

$B([5, 2, 4, 3],\ [2, 2, 4, 2],\ 6) \rightarrow max[1, 1, 0, 1]$

$z = [1, 0, 0, -1]$, $||z||_2 = \sqrt{2}$ $\rightarrow min$

$p = [1, 2, 3, 0]$

$A([5, 4, 3, 2],\ [1, 2, 3, 0],\ 8) \rightarrow max[1, 1, 1, 0]$

$B([5, 2, 4, 3],\ [1, 2, 3, 0],\ 6) \rightarrow max[1, 0, 1, 1]$

$z = [1, -1, 1, -1]$, $||z||_2 = 2$

$update$: $p = [2, 2, 4, 2]$

Iteration 2:

2) $||z||_2 = \sqrt{2} \neq 0$

3) $H = \{[1, 2, 3, 4],\ [1, 3, 2, 4],\ [2, 1, 3, 4],\ [2, 3, 1, 4],\ [3, 1, 2, 4],\ [3, 2, 1, 4],\ [2, 2, 1, 4],$
$[2, 1, 2, 4],\ [1, 2, 2, 4],\ [2, 2, 2, 4],\ [0, 1, 5, 2],\ [1, 2, 3, 3],\ [0, 2, 3, 4],\ [3, 2, 0, 4],$
$[2, 1, 3, 3],\ [2, 1, 3, 2],\ [1, 1, 3, 4],\ [0, 1, 3, 4],\ [2, 1, 2, 4],\ [2, 1, 1, 4],\ [1, 1, 3, 3],$
$[1, 1, 2, 4],\ [2, 1, 2, 3],\ [2, 0, 3, 4],\ [0, 0, 3, 4],\ [2, 0, 1, 4],\ [2, 0, 3, 2],\ [1, 0, 2, 4],$
$[1, 0, 3, 3],\ [2, 0, 2, 3],\ ...,$
$[1, 3, 4, 2],\ [3, 1, 4, 2],\ [2, 1, 4, 3],\ [1, 2, 4, 3],\ [1, 1, 5, 4],\ [2, 1, 5, 3],\ [1, 1, 6, 1],$
$[1, 1, 6, 2],\ [1, 1, 6, 3],\ [1, 1, 6, 4],\ ...\}$

$\#\ p_0 + p_1 + p_2 \leq 6 = b_0(B),\ p_0 + p_2 + p_3 > 6 = b_0(B)$

$\#\ 6 < p_0 + p_1 + p_2 \leq 8,\ p_0 + p_1 + p_3 \leq 6$

$N^G(p, \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \Delta \subseteq [0, 1]$

$= \{[2, 2, 4, 2] + \delta \cdot [1, 0, 0, -1]\}$

$= [3, 2, 4, 1]$ $(\delta = 1)$

$N(p) = \{[2, 2, 4, 0]\}$ *# For under-demanded courses, we consider setting them to zero*

Examination:

$p = [3, 2, 4, 1]$

$A([5,4,3,2],\ [3,2,4,1],\ 8) \to max[1,1,0,1]$

$B([5,2,4,3],\ [3,2,4,1],\ 6) \to max[1,1,0,1]$

$z = [1,0,-1,0]\,, \qquad ||z||_2 = \sqrt{2} \qquad \to min$

$p = [2,2,4,0]$

$A([5,4,3,2],\ [2,2,4,0],\ 8) \to max[1,1,1,0]$

$B([5,2,4,3],\ [2,2,4,0],\ 6) \to max[1,0,1,1]$

$z = [1,-1,1,-1]\,, \qquad ||z||_2 = 2$

$update\colon\ p = [3,2,4,1]$

Iteration 3:

2) $||z||_2 = \sqrt{2} \neq 0$

3) $H = \{[1,2,3,4],\ [1,3,2,4],\ [2,1,3,4],\ [2,3,1,4],\ [3,1,2,4],\ [3,2,1,4],\ [2,2,1,4],$
$[2,1,2,4],\ [1,2,2,4],\ [2,2,2,4],\ [0,1,5,2],\ [1,2,3,3],\ [0,2,3,4],\ [3,2,0,4],$
$[2,1,3,3],\ [2,1,3,2],\ [1,1,3,4],\ [0,1,3,4],\ [2,1,2,4],\ [2,1,1,4],\ [1,1,3,3],$
$[1,1,2,4],\ [2,1,2,3],\ [2,0,3,4],\ [0,0,3,4],\ [2,0,1,4],\ [2,0,3,2],\ [1,0,2,4],$
$[1,0,3,3],\ [2,0,2,3],\ \dots\ ,$
$[1,3,4,2],\ [3,1,4,2],\ [2,1,4,3],\ [1,2,4,3],\ [1,1,5,4],\ [2,1,5,3],\ [1,1,6,1],$
$[1,1,6,2],\ [1,1,6,3],\ [1,1,6,4],\ \dots\ ,$
$[2,3,4,1],\ [1,1,7,1],\ [2,2,5,2],\ [1,2,7,1],\ [1,3,5,1],\ \dots\ \}$

$\#\ p_0 + p_1 + p_2 \leq 6 = b_0(B),\ p_0 + p_2 + p_3 > 6 = b_0(B)$

$\#\ 6 < p_0 + p_1 + p_2 \leq 8,\ p_0 + p_1 + p_3 \leq 6$

$\#\ p_0 + p_1 + p_2 > 8,\ p_0 + p_1 + p_3 \leq 6$

$N^G(p,\ \Delta) = \{p + \delta \cdot z\colon \delta \in \Delta\},\ \Delta \subseteq [0,\ 1]$

$= \{[3,2,4,1] + \delta \cdot [1,0,-1,0]\}$

$= [4,2,3,1] \qquad (\delta = 1)$

$N(p) = \{\,[3,2,3,1],$ *# Student A will be forced to change his choice*

$[3,2,0,1]\,\}$ *# For under-demanded courses, we consider setting them to zero*

Examination:

$p = [4,2,3,1]$

$A([5,4,3,2],\ [4,2,3,1],\ 8) \to max[1,1,0,1]$

$B([5,2,4,3],\ [4,2,3,1],\ 6) \rightarrow max[0,1,1,1]$

$z = [0,0,0,0]\,, \qquad ||z||_2 = 0 \qquad \rightarrow min$

$p = [3,2,3,1]$

$A([5,4,3,2],\ [3,2,3,1],\ 8) \rightarrow max[1,1,1,0]$

$B([5,2,4,3],\ [3,2,3,1],\ 6) \rightarrow max[1,1,0,1]$

$z = [1,0,0,-1]\,, \qquad ||z||_2 = \sqrt{2}$

$p = [3,2,0,1]$

$A([5,4,3,2],\ [3,2,0,1],\ 8) \rightarrow max[1,1,1,0]$

$B([5,2,4,3],\ [3,2,0,1],\ 6) \rightarrow max[1,0,1,1]$

$z = [1,-1,1,-1]\,, \qquad ||z||_2 = 2$

$update\colon\ p = [4,2,3,1]$

Iteration 5:

2) $||z||_2 = 0 \Rightarrow p^* = p = [4,2,3,1]$

Algorithm results:

Final course prices: $p^* = [4,2,3,1]$

Final distribution of courses: $A\colon\ [1,1,0,1]$

$B\colon\ [0,1,1,1]$

Example run 3

$n = 3$ # Number of students

$m = 4$ # Number of courses

parameter:

$\beta = 5,\ b_0 \in [1,\ 6]$

$capacities = [1,\ 2,\ 2,\ 1]$

$utilities = [[3,\ 3,\ 3,\ 3],\ [3,\ 3,\ 3,\ 3],\ [4,\ 4,\ 4,\ 4]]$ [A, B, C]

$b_0 = [4,\ 5,\ 2]$

$required = [2,\ 2,\ 2,\ 2]$ # The number of courses each student should take

Running the algorithm:

1) $p \leftarrow uniform(1, 6)^4$

$p = [1,\ 1,\ 1,\ 1]$

$H \leftarrow \phi$

2) $z_j(u, c, p, b) = \sum_{i=1}^{n} a_{ij}(u, p, b) - c_j\ ,\ \ a_i(u, p, b) = arg\ max\ u_i(x)$

$A([3, 3, 3, 3],\ [1, 1, 1, 1],\ 4) \rightarrow max[1, 1, 0, 0]$

$B([3, 3, 3, 3],\ [1, 1, 1, 1],\ 5) \rightarrow max[1, 1, 0, 0]$

$C([4, 4, 4, 4],\ [1, 1, 1, 1],\ 2) \rightarrow max[1, 1, 0, 0]$

$z = [2,\ 1, -2, -1]\ ,\qquad ||z||_2 = \sqrt{10}$

3) $H = \{[1, 1, 1, 1], [0, 0, 0, 0], [1, 0, 0, 0], [0, 1, 0, 0],\ [1, 1, x, x], [2, 0, x, x],$
$[0, 2, x, x]\ \}\ \ x \in N$

$\#\ p_0 + p_1 \leq 2 = b_0(C)$

$N^G(p,\ \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \ \Delta \subseteq [0,\ 1]$

$= \{[1, 1, 1, 1] + \delta \cdot [2, 1, -2, -1]\}$

$= [3, 2, -1, 0] \qquad (\delta = 1)$

$\approx [3, 2, 0, 0]$ *# When the price is negative we will update it to 0*

$N(p) = \{\ [1, 2, 1, 1],\ [2, 1, 1, 1]\ \}$ *# Student C will be forced to change his choice*

Examination:

$p = [3, 2, 0, 0]$

$A([3, 3, 3, 3], [3, 2, 0, 0], 4) \rightarrow max[1, 0, 1, 0]$

$B([3, 3, 3, 3], [3, 2, 0, 0], 5) \rightarrow max[1, 1, 0, 0]$

$C([4, 4, 4, 4], [3, 2, 0, 0], 2) \rightarrow max[0, 1, 1, 0]$

$z = [1, 0, 0, -1]$, $||z||_2 = \sqrt{2}$ $\rightarrow min$

$p = [1, 2, 1, 1]$

$A([3, 3, 3, 3], [1, 2, 1, 1], 4) \rightarrow max[1, 1, 0, 0]$

$B([3, 3, 3, 3], [1, 2, 1, 1], 5) \rightarrow max[1, 1, 0, 0]$

$C([4, 4, 4, 4], [1, 2, 1, 1], 2) \rightarrow max[1, 0, 1, 0]$

$z = [2, 0, -1, -1]$, $||z||_2 = 2$

$p = [2, 1, 1, 1]$

$A([3, 3, 3, 3], [2, 1, 1, 1], 4) \rightarrow max[1, 1, 0, 0]$

$B([3, 3, 3, 3], [2, 1, 1, 1], 5) \rightarrow max[1, 1, 0, 0]$

$C([4, 4, 4, 4], [2, 1, 1, 1], 2) \rightarrow max[0, 1, 1, 0]$

$z = [1, 1, -1, -1]$, $||z||_2 = 2$

$update$: $p = [3, 2, 0, 0]$

$update$: $\bar{z} = [1, 0, 0, 0]$

Iteration 2:

2) $||z||_2 = \sqrt{2} \neq 0$

3) $H = \{[1, 1, 1, 1], [0, 0, 0, 0], [1, 0, 0, 0], [0, 1, 0, 0], [1, 1, x, x], [2, 0, x, x],$
$[0, 2, x, x], [4, 1, 0, x] \}$ $x \in N$

$\# p_0 + p_1 \leq 2 = b_0(C)$

$\# p_0 + p_1 = 5 = b_0(B), \; p_0 + p_2 = 4 = b_0(A), \; p_1 + p_2 = 2 = b_0(C)$

$N^G(p, \Delta) = \{p + \delta \cdot z : \delta \in \Delta\}, \; \Delta \subseteq [0, 1]$

$= \{[3, 2, 0, 0] + \delta \cdot [1, 0, 0, 0]\}$

$= [4, 2, 0, 0]$ $(\delta = 1)$

$N(p) = \phi$

Examination:

$p = [4, 2, 0, 0]$

$A([3, 3, 3, 3],\ [4, 2, 0, 0],\ 4) \rightarrow max[1, 0, 1, 0]$

$B([3, 3, 3, 3],\ [4, 2, 0, 0],\ 5) \rightarrow max[1, 0, 1, 0]$

$C([4, 4, 4, 4],\ [4, 2, 0, 0],\ 2) \rightarrow max[0, 1, 1, 0]$

$z = [1, -1, 1, -1]\ ,\qquad ||z||_2 = 2 \qquad \rightarrow min$

$update\colon\ p = [4, 2, 0, 0]$

$update\colon\ \bar{z} = [1, -1, 1, 0]$

Iteration 3:

2) $||z||_2 = 2 \neq 0$

3) $H = \{[1, 1, 1, 1], [0, 0, 0, 0], [1, 0, 0, 0], [0, 1, 0, 0],\ [1, 1, x, x], [2, 0, x, x],$
$[0, 2, x, x],\ [4, 1, 0, x],\ [4, 2, 0, x]\ \}\ \ x \in N$

$\#\ p_0 + p_1 \leq 2 = b_0(C)$

$\#\ p_0 + p_1 = 5 = b_0(B),\ \ p_0 + p_2 = 4 = b_0(A),\ \ p_1 + p_2 = 2 = b_0(C)$

$\#\ p_0 = 4 = b_0(A),\ \ p_0 + p_2 = 4 = b_0(A),\ \ p_1 + p_2 = 2 = b_0(C)$

$N^G(p,\ \Delta) = \{p + \delta \cdot z\colon \delta \in \Delta\},\ \ \Delta \subseteq [0,\ 1]$

$= \{[4, 2, 0, 0] + \delta \cdot [1, -1, 1, 0]\}$

$= [5, 1, 0, 0] \qquad (\delta = 1)$

$N(p) = \{\ [5, 2, 0, 0],$ *# Student A will be forced to change his choice*

$[4, 0, 0, 0]\ \}$ *# For under-demanded courses, we consider setting them to zero*

Examination:

$p = [5, 1, 0, 0]$

$A([3, 3, 3, 3],\ [5, 1, 0, 0],\ 4) \rightarrow max[0, 1, 1, 0]$

$B([3, 3, 3, 3],\ [5, 1, 0, 0],\ 5) \rightarrow max[1, 0, 1, 0]$

$C([4, 4, 4, 4],\ [5, 1, 0, 0],\ 2) \rightarrow max[0, 1, 1, 0]$

$z = [0, 0, 1, -1]\ ,\qquad ||z||_2 = \sqrt{2} \qquad \rightarrow min$

$p = [5, 2, 0, 0]$

$A([3, 3, 3, 3],\ [5, 2, 0, 0],\ 4) \rightarrow max[0, 1, 1, 0]$

$B([3, 3, 3, 3],\ [5, 2, 0, 0],\ 5) \rightarrow max[1, 0, 1, 0]$

$C([4, 4, 4, 4],\ [5, 2, 0, 0],\ 2) \rightarrow max[0, 1, 1, 0]$

$z = [0, 0, 1, -1]$, $||z||_2 = \sqrt{2}$

$p = [4, 0, 0, 0]$

$A([3, 3, 3, 3],\ [4, 0, 0, 0],\ 4) \rightarrow max[1, 1, 0, 0]$

$B([3, 3, 3, 3],\ [4, 0, 0, 0],\ 5) \rightarrow max[1, 1, 0, 0]$

$C([4, 4, 4, 4],\ [4, 0, 0, 0],\ 2) \rightarrow max[0, 1, 1, 0]$

$z = [1, 1, -1, -1]$, $||z||_2 = 2$

$update$: $p = [5, 1, 0, 0]$

$update$: $\bar{z} = [0, 0, 1, 0]$

Iteration 4:

2) $||z||_2 = \sqrt{2} \neq 0$

3) $H = \{[1, 1, 1, 1], [0, 0, 0, 0], [1, 0, 0, 0], [0, 1, 0, 0],\ [1, 1, x, x], [2, 0, x, x],$
$[0, 2, x, x],\ [4, 1, 0, x],\ [4, 2, 0, x],\ [5, 1, 0, x],\ [5, 2, 0, x]\ \}\ \ x \in N$

$\# p_0 + p_1 \leq 2 = b_0(C)$

$\# p_0 + p_1 = 5 = b_0(B),\ p_0 + p_2 = 4 = b_0(A),\ p_1 + p_2 = 2 = b_0(C)$

$\# p_0 = 4 = b_0(A),\ p_0 + p_2 = 4 = b_0(A),\ p_1 + p_2 = 2 = b_0(C)$

$\# p_0 + p_2 = 5 = b_0(B),\ p_1 + p_2 \leq 2 = b_0(C)$

$N^G(p,\ \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \Delta \subseteq [0,\ 1]$
$= \{[5, 1, 0, 0] + \delta \cdot [0, 0, 1, 0]\}$
$= [5, 1, 1, 0]$ $(\delta = 1)$

$N(p) = \{[5, 1, 1, 0]\} = N^G$

Examination:

$p = [5, 1, 1, 0]$

$A([3, 3, 3, 3],\ [5, 1, 1, 0],\ 4) \rightarrow max[0, 1, 1, 0]$

$B([3, 3, 3, 3],\ [5, 1, 1, 0],\ 5) \rightarrow max[1, 0, 0, 1]$

$C([4, 4, 4, 4],\ [5, 1, 1, 0],\ 2) \rightarrow max[0, 1, 1, 0]$

$z = [0, 0, 0, 0]$, $||z||_2 = 0$ $\rightarrow min$

$update$: $p = [5, 1, 1, 0]$

Iteration 5:

2) $||z||_2 = 0 \Rightarrow p^* = p = [5, 1, 1, 0]$

Algorithm results:

Final course prices: $p^* = [5, 1, 1, 0]$

Final distribution of courses: A: $[0, 1, 1, 0]$

B: $[1, 0, 0, 1]$

C: $[0, 1, 1, 0]$

Example run 4

Extra capacities

$n = 2$ # Number of students

$m = 3$ # Number of courses

parameter:

$\beta = 6,\ b_0 \in [1,\ 7]$

$capacities = [1,\ 2,\ 3]$

$utilities = [[4, 3, 2],\ [5, 1, 2]]$ [A, B]

$b_0 = [6, 4]$

$required = [2,\ 2]$ # The number of courses each student should take

Running the algorithm:

1) $p \leftarrow uniform(1, 7)^3$

$p = [1, 2, 3]$

$H \leftarrow \phi$

2) $z_j(u, c, p, b) = \sum_{i=1}^{n} a_{ij}(u, p, b) - c_j\ ,\ a_i(u, p, b) = arg\ max\ u_i(x)$

$A([4, 3, 2],\ [1, 2, 3],\ 6) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [1, 2, 3],\ 4) \rightarrow max[1, 0, 1]$

$z = [1, -1, -2]\ ,\ ||z||_2 = \sqrt{6}$

3) $H = \{[1, 2, 3],\ [3, 1, 1],\ [3, 3, 1],\ [3, 2, 1],\ [1, 3, 1],\ \dots\}$

$\#\ p_0 + p_1 \leq 6 = b_0(A),\ p_0 + p_2 \leq 4 = b_0(B)$

$N^G(p,\ \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \Delta \subseteq [0,\ 1]$

$= \{[1, 2, 3] + \delta \cdot [1, -1, -2]\}$

$= [2, 1, 1]$ $(\delta = 1)$

$N(p) = \phi$

Examination:

$p = [2, 1, 1]$

$A([4, 3, 2],\ [2, 1, 1],\ 6) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [2, 1, 1],\ 4) \rightarrow max[1, 0, 1]$

$z = [1, -1, -2]\ ,\ ||z||_2 = \sqrt{6} \quad \rightarrow min$

$update\colon\ p = [2, 1, 1]$

Iteration 2:

2) $||z||_2 = \sqrt{6} \neq 0$

3) $H = \{[1, 2, 3],\ [3, 1, 1],\ [3, 3, 1],\ [3, 2, 1],\ [1, 3, 1],\ [2, 1, 1],\ \dots\}$

$\#\ p_0 + p_1 \leq 6 = b_0(A),\ \ p_0 + p_2 \leq 4 = b_0(B)$

$N^G(p,\ \Delta) = \{p + \delta \cdot z\colon \delta \in \Delta\},\ \ \Delta \subseteq [0,\ 1]$

$= \{[2, 1, 1] + \delta \cdot [1, -1, -2]\}$

$= [3, 0, -1] \qquad (\delta = 1)$

$\approx [3, 0, 0] \quad \#\ When\ the\ price\ is\ negative\ we\ will\ update\ it\ to\ 0$

$N(p) = \{[4, 1, 1]\}$ $\quad \#\ Student\ C\ will\ be\ forced\ to\ change\ his\ choice$

Examination:

$p = [3, 0, 0]$

$A([4, 3, 2],\ [3, 0, 0],\ 6) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [3, 0, 0],\ 4) \rightarrow max[1, 0, 1]$

$z = [1, -1, -2]\ ,\ ||z||_2 = \sqrt{6}$

$p = [4, 1, 1]$

$A([4, 3, 2],\ [4, 1, 1],\ 6) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [4, 1, 1],\ 4) \rightarrow max[0, 1, 1]$

$z = [0, 0, -2]\ ,\ \ ||z||_2 = 2 \quad \rightarrow min$

$update\colon\ p = [4, 1, 1]$

Iteration 3:

2) $||z||_2 = 2 \neq 0$

3) $H = \{[1, 2, 3],\ [3, 1, 1],\ [3, 3, 1],\ [3, 2, 1],\ [1, 3, 1],\ [2, 1, 1],\ [4, 2, 1],\ [4, 2, 2],$

$[3, 2, 2], \ldots \}$

$\#\ p_0 + p_1 \leq 6 = b_0(A),\ p_0 + p_2 \leq 4 = b_0(B)$

$\#\ p_0 + p_1 \leq 6 = b_0(A),\ p_1 + p_2 \leq 4 = b_0(B),\ p_0 + p_2 > 4 = b_0(B)$

$N^G(p, \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \Delta \subseteq [0, 1]$

$= \{[4, 1, 1] + \delta \cdot [0, 0, -2]\}$

$= [4, 1, -1] \qquad (\delta = 1)$

$\approx [4, 1, 0]$ *# When the price is negative we will update it to 0*

$N(p) = \phi$

Examination:

$p = [4, 1, 0]$

$A([4, 3, 2],\ [4, 1, 0],\ 6) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [4, 1, 0],\ 4) \rightarrow max[1, 0, 1]$

$z = [1, -1, -2]\ ,\ ||z||_2 = \sqrt{6} \qquad \rightarrow min$

$update\colon\ p = [4, 1, 0]$

$update\colon\ \bar{z} = [1, -1, 0]$

Iteration 4:

2) $||z||_2 = \sqrt{6} \neq 0$

3) $H = \{[1, 2, 3],\ [3, 1, 1],\ [3, 3, 1],\ [3, 2, 1],\ [1, 3, 1],\ [2, 1, 1],\ [4, 2, 1],\ [4, 2, 2],$
$[3, 2, 2], \ldots \}$

$\#\ p_0 + p_1 \leq 6 = b_0(A),\ p_0 + p_2 \leq 4 = b_0(B)$

$\#\ p_0 + p_1 \leq 6 = b_0(A),\ p_1 + p_2 \leq 4 = b_0(B),\ p_0 + p_2 > 4 = b_0(B)$

$N^G(p, \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \Delta \subseteq [0, 1]$

$= \{[4, 1, 0] + \delta \cdot [1, -1, 0]\}$

$= [5, 0, 0] \qquad (\delta = 1)$

$N(p) = \{[6, 1, 0]\}$ *# Student C will be forced to change his choice*

Examination:

$p = [5, 0, 0]$

$A([4, 3, 2],\ [5, 0, 0],\ 6) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [5, 0, 0],\ 4) \rightarrow max[0, 1, 1]$

$z = [0, 0, -2]\,,\ \ ||z||_2 = 2$

$p = [6, 1, 0]$

$A([4, 3, 2],\ [6, 1, 0],\ 6) \rightarrow max[1, 0, 1]$

$B([5, 1, 2],\ [6, 1, 0],\ 4) \rightarrow max[0, 1, 1]$

$z = [0, -1, -1]\,,\ ||z||_2 = \sqrt{2}$ $\quad \rightarrow min$

$update\colon\ p = [6, 1, 0]$

$update\colon\ \bar{z} = [0, -1, 0]$

Iteration 5:

2) $||z||_2 = 1 \neq 0$

3) $H = \{[1, 2, 3],\ [3, 1, 1],\ [3, 3, 1],\ [3, 2, 1],\ [1, 3, 1],\ [2, 1, 1],\ [4, 2, 1],\ [4, 2, 2],$
$[3, 2, 2],$
$[4, 3, 1] \ldots \}$
$\#\ p_0 + p_1 \leq 6 = b_0(A),\ \ p_0 + p_2 \leq 4 = b_0(B)$
$\#\ p_0 + p_1 \leq 6 = b_0(A),\ \ p_1 + p_2 \leq 4 = b_0(B),\ \ p_0 + p_2 > 4 = b_0(B)$
$\#\ p_0 + p_2 = 6 = p_0(A)\ \ ,\ p_1 + p_2 = 4 = p_0(B)$

$N^G(p,\ \Delta) = \{p + \delta \cdot z\colon \delta \in \Delta\},\ \ \Delta \subseteq [0,\ 1]$
$= \{[6, 1, 0] + \delta \cdot [0, -1, 0]\}$
$= [6, 0, 0] \qquad (\delta = 1)$

$N(p) = \{\,[6, 0, 0]\,\} = N^G$ $\quad \#\ For\ under\text{-}demanded\ courses,\ we\ consider\ setting\ them\ to\ zero$

Examination:

$p = [6, 0, 0]$

$A([4, 3, 2],\ [6, 0, 0],\ 6) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [6, 0, 0],\ 4) \rightarrow max[0, 1, 1]$

$z = [0, 0, -2]\,,\ ||z||_2 = 2$ $\quad \rightarrow min$

$update\colon\ p = [6, 0, 0]$

$update$: $\bar{z} = [0, 0, 0]$

Iteration 6:

2) $||z||_2 = 0 \Rightarrow p^* = p = [6, 0, 0]$

Algorithm results:

Final course prices: $p^* = [6, 0, 0]$

Final distribution of courses: A: $[1, 1, 0]$

B: $[0, 1, 1]$

Example run 5

lack of capacity

$n = 2$ # Number of students

$m = 3$ # Number of courses

parameter:

$\beta = 6,\ b_0 \in [1,\ 7]$

$capacities = [1,\ 1,\ 1]$

$utilities = [[4, 3, 2],\ [5, 1, 2]]$ [A, B]

$b_0 = [5, 3]$

$required = [2,\ 2]$ # The number of courses each student should take

Running the algorithm:

1) $p \leftarrow uniform(1, 7)^3$

$p = [2, 2, 2]$

$H \leftarrow \phi$

2) $z_j(u, c, p, b) = \sum_{i=1}^{n} a_{ij}(u, p, b) - c_j\ ,\ \ a_i(u, p, b) = arg\ max\ u_i(x)$

$A([4, 3, 2],\ [2, 2, 2],\ 5) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [2, 2, 2],\ 3) \rightarrow max[1, 0, 0]$

$z = [1, 0, -1]\ ,\ \ ||z||_2 = \sqrt{2}$

3) $H = \{[2, 2, 2],\ [3, 2, 2],\ [3, 1, 1],\ [3, 2, 1],\ [3, 1, 2],\ [2, 3, 2],\ [2, 3, 3],\ \dots\}$

$\#\ p_0 + p_1 \leq 5 = b_0(A),\ \ p_0 \leq 3 = b_0(B),\ \ p_0 + p_1 > 3,\ \ p_0 + p_2 > 3$

$N^G(p,\ \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \ \Delta \subseteq [0,\ 1]$

$= \{[2, 2, 2] + \delta \cdot [1, 0, -1]\}$

$= [3, 2, 1] \qquad (\delta = 1)$

$N(p) = \{\ [2, 2, 0]\ \}$

Examination:

$p = [3, 2, 1]$

$A([4, 3, 2],\ [3, 2, 1],\ 5) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [3, 2, 1],\ 3) \rightarrow max[0, 1, 1]$

$z = [0, 1, 0]\ ,\quad ||z||_2 = 1 \qquad \rightarrow min$

$p = [2, 2, 0]$

$A([4, 3, 2],\ [2, 2, 0],\ 5) \rightarrow max[1, 1, 0]$

$B([5, 1, 2],\ [2, 2, 0],\ 3) \rightarrow max[1, 0, 1]$

$z = [1, 0, 0]\ ,\quad ||z||_2 = 1$

$update\colon\ p = [3, 2, 1]$

Iteration 2:

2) $||z||_2 = 1 \neq 0$

3) $H = \{[2, 2, 2],\ [3, 2, 2],\ [3, 1, 1],\ [3, 2, 1],\ [3, 1, 2],\ [2, 3, 2],\ [2, 3, 3],$
$[4, 1, 1],\ [3, 1, 1],\ [3, 1, 2],\ \dots\}$

$\#\ p_0 + p_1 \leq 5 = b_0(A),\ p_0 \leq 3 = b_0(B),\ p_0 + p_1 > 3,\ p_0 + p_2 > 3$

$\#\ p_0 + p_1 \leq 5 = b_0(A),\ p_1 + p_2 \leq 3 = b_0(B),\ p_0 + p_1 > 3,\ p_0 + p_2 > 3$

$N^G(p,\ \Delta) = \{p + \delta \cdot z\colon \delta \in \Delta\},\ \Delta \subseteq [0,\ 1]$

$= \{[3, 2, 1] + \delta \cdot [0, 1, 0]\}$

$= [3, 3, 1] \qquad (\delta = 1)$

$N(p) = \phi$

Examination:

$p = [3, 3, 1]$

$A([4, 3, 2],\ [3, 3, 1],\ 5) \rightarrow max[1, 0, 1]$

$B([5, 1, 2],\ [3, 3, 1],\ 3) \rightarrow max[1, 0, 0]$

$z = [1, -1, 0]\ ,\quad ||z||_2 = \sqrt{2} \qquad \rightarrow min$

$update\colon\ p = [3, 3, 1]$

Iteration 3:

2) $||z||_2 = 2 \neq 0$

3) $H = \{[2,2,2],\ [3,2,2],\ [3,1,1],\ [3,2,1],\ [3,1,2],\ [2,3,2],\ [2,3,3],$
$[4,1,1],\ [3,1,1],\ [3,1,2],\ \dots,$
$[3,3,1],\ [3,x,2]\}\ \ x \in N$
$\#\ p_0 + p_1 \le 5 = b_0(A),\ p_0 \le 3 = b_0(B),\ p_0 + p_1 > 3,\ p_0 + p_2 > 3$
$\#\ p_0 + p_1 \le 5 = b_0(A),\ p_1 + p_2 \le 3 = b_0(B),\ p_0 + p_1 > 3,\ p_0 + p_2 > 3$

$N^G(p, \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \Delta \subseteq [0, 1]$
$= \{[3,3,1] + \delta \cdot [1,-1,0]\}$
$= [4,2,1] \qquad (\delta = 1)$

$N(p) = \{[4,3,1],$ *# Student B will be forced to change his choice*
$[3,0,1]\}$ *# For under-demanded courses, we consider setting them to zero*

Examination:

$p = [4,2,1]$

$A([4,3,2],\ [4,2,1],\ 5) \to max[1,0,1]$

$B([5,1,2],\ [4,2,1],\ 3) \to max[0,1,1]$

$z = [0,0,1],\ \ ||z||_2 = 1 \qquad \to min$

$p = [4,3,1]$

$A([4,3,2],\ [4,3,1],\ 5) \to max[1,0,1]$

$B([5,1,2],\ [4,3,1],\ 3) \to max[0,0,1]$

$z = [0,-1,1],\ \ ||z||_2 = \sqrt{2}$

$p = [3,0,1]$

$A([4,3,2],\ [3,0,1],\ 5) \to max[1,1,0]$

$B([5,1,2],\ [3,0,1],\ 3) \to max[1,1,0]$

$z = [1,1,-1],\ \ ||z||_2 = \sqrt{3}$

$update:\ p = [4,2,1]$

Iteration 4:

2) $||z||_2 = 1 \neq 0$

3) $H = \{[2,2,2],\ [3,2,2],\ [3,1,1],\ [3,2,1],\ [3,1,2],\ [2,3,2],\ [2,3,3],$
$[4,1,1],\ [3,1,1],\ [3,1,2],\ \dots,$

$[3, 3, 1],\ [3, x, 2],$

$[5, 1, 0]\ \}\ \ x \in N$

$\#\ p_0 + p_1 \leq 5 = b_0(A),\ p_0 \leq 3 = b_0(B),\ p_0 + p_1 > 3,\ p_0 + p_2 > 3$

$\#\ p_0 + p_1 \leq 5 = b_0(A),\ p_1 + p_2 \leq 3 = b_0(B),\ p_0 + p_1 > 3,\ p_0 + p_2 > 3$

$$N^G(p, \Delta) = \{p + \delta \cdot z : \delta \in \Delta\},\ \Delta \subseteq [0, 1]$$
$$= \{[4, 2, 1] + \delta \cdot [0, 0, 1]\}$$
$$= [4, 2, 2] \qquad (\delta = 1)$$

$N(p) = \{\,[4, 2, 2]\,\} = N^G$ *# Student B will be forced to change his choice*

Examination:

$p = [4, 2, 2]$

$A([4, 3, 2],\ [4, 2, 2],\ 5) \rightarrow max[0, 1, 1]$

$B([5, 1, 2],\ [4, 2, 2],\ 3) \rightarrow max[1, 0, 0]$

$z = [0, 0, 0]\,,\quad ||z||_2 = 0 \qquad \rightarrow min$

$update\colon\ p = [4, 2, 2]$

Iteration 5:

2) $||z||_2 = 0 \Rightarrow p^* = p = [4, 2, 2]$

Algorithm results:

Final course prices: $p^* = [4, 2, 2]$

Final distribution of courses: $A\colon\ [0, 1, 1]$

$B\colon\ [1, 0, 0]$

Example run 6

obvious cases:

 # Case 1:

$n = 100$ # Number of students

$m = 500$ # Number of courses

$capacities = [200, 200, \dots, 200]$

$\Rightarrow$ *Each student will get all the courses*

 # Case 2:

$n = 100$ # Number of students

$m = 100$ # Number of courses

$capacities = [1, 1, 1, \dots, 1]$

$\forall student\ i \in [n]$: $utilities = only\ course\ i$

$\Rightarrow$ *Each student i will get course i*

 # Case 3:

$n = 100$ # Number of students

$m = 100$ # Number of courses

$capacities = [1, 1, 1, \dots, 1]$

$\forall student\ i \in [n]$: $u_i = [100, 99, 98, \dots, 1]$

$b_0 = [100, 99, 98, \dots, 1]$

$\Rightarrow$ *Each student i will get course i, because student i have the highest i budget.*

 # Case 4:

$n = 100$ # Number of students

$m = 300$ # Number of courses

$capacities = [200, 200, \dots, 200]$

$required = 3$ # The number of courses each student should take

$\Rightarrow$ *Each student will get his* 3 *favorite courses*

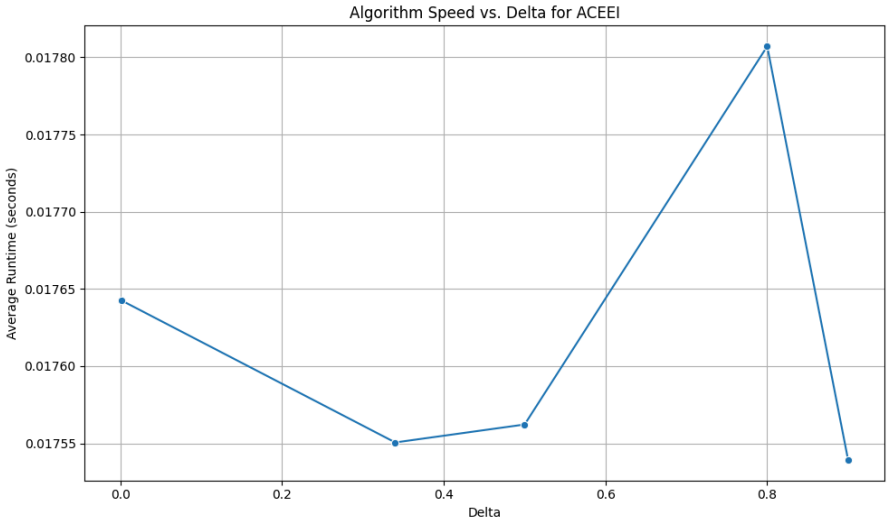
Performance comparison

ACEEI algorithm

The effect of delta δ and epsilon ϵ on the running time

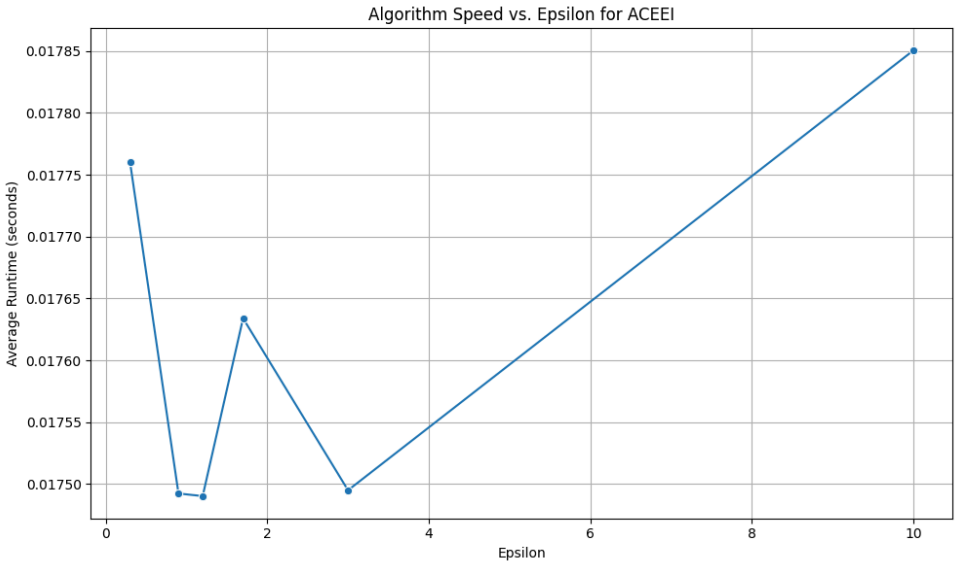
Delta represents the size of the price change in each iteration.

According to the graph you can see that for $\delta = 0.5$ we got the shortest running time.



Epsilon is the size of the budget disruption we check for each student

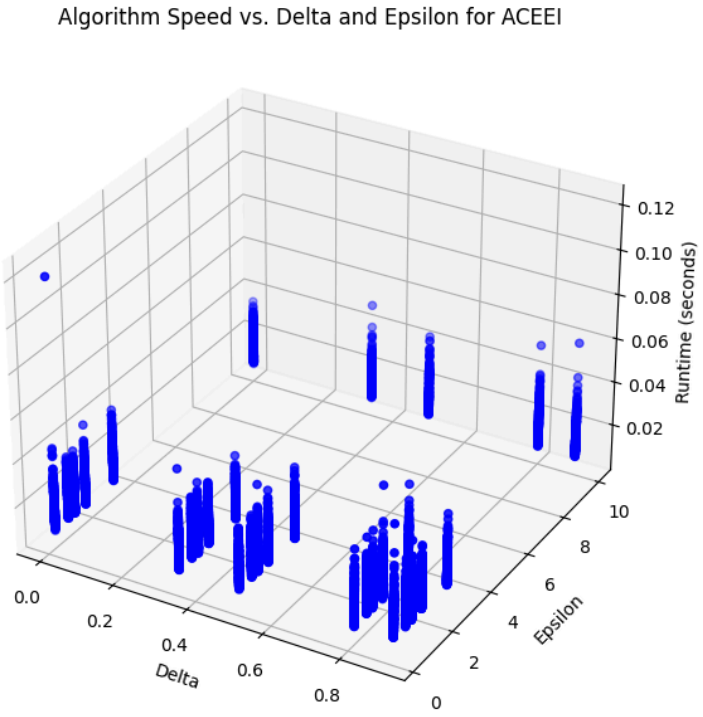
It can be seen from the graph that the shortest running time was obtained for $\epsilon = 1.2$



We have added a graph representing the running time for each combination of epsilon and delta.

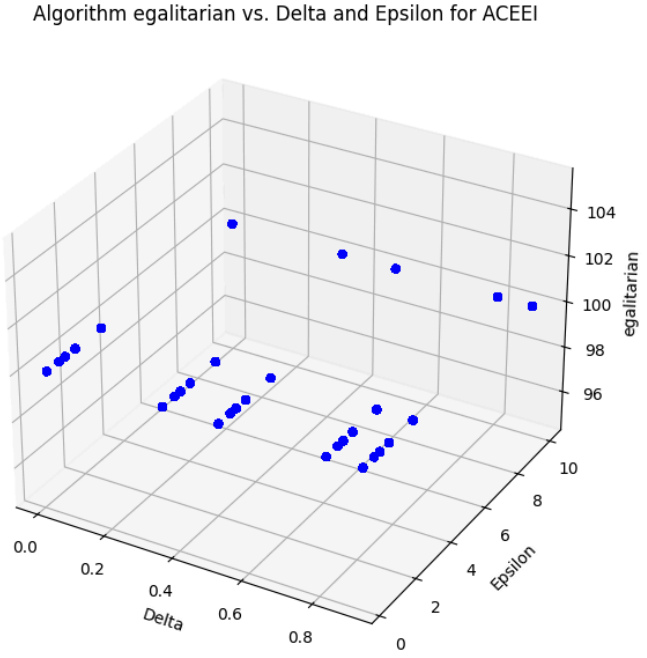
The minimum runtime was obtained for the values:

```
Best delta: 0.5
Best epsilon: 3.0
```



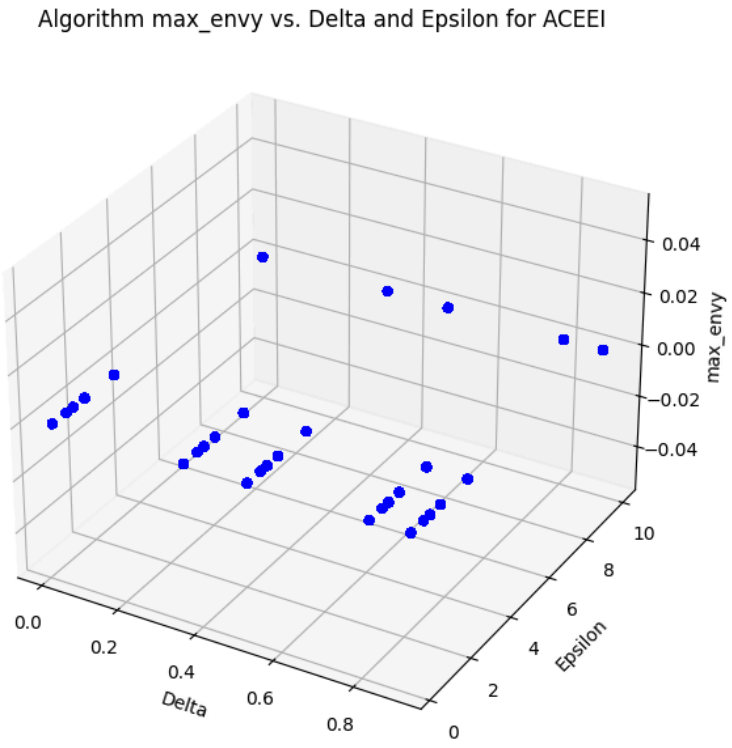
The effect of delta δ and epsilon ϵ on the egalitarian value

```
Best delta: 0.001
Best epsilon: 0.3
```



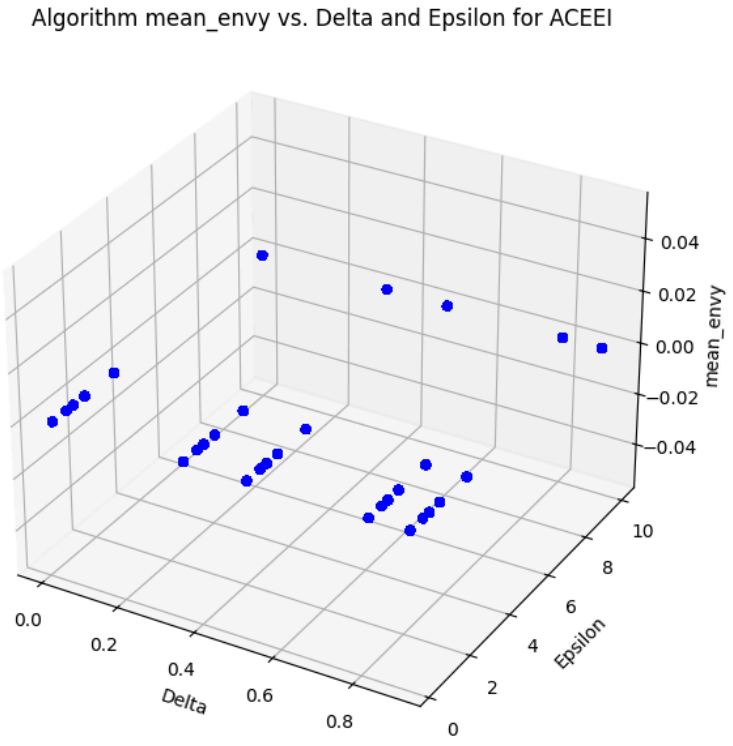
The effect of delta δ and epsilon ϵ on the max envy

```
Best delta: 0.001
Best epsilon: 0.3
```



The effect of delta δ and epsilon ϵ on the mean envy

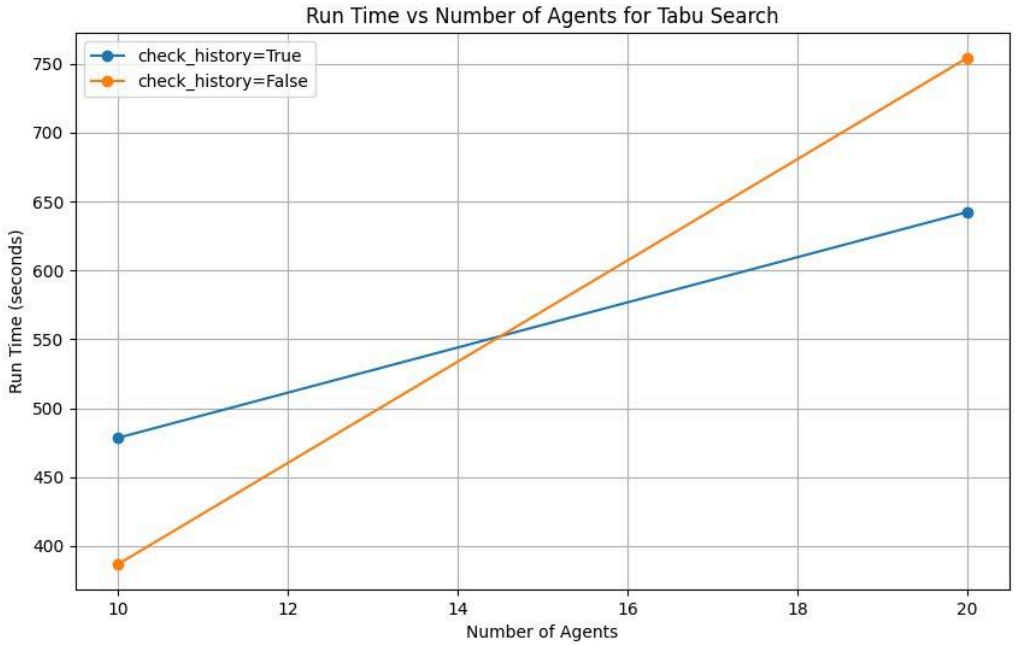
```
Best delta: 0.001
Best epsilon: 0.3
```



Tabu Search algorithm

Using history

Our algorithm is tabu search. In the algorithm we want to divide courses, and to do this, we define a price vector for the courses. The algorithm defines a history that saves price vectors .that give the same result, so that we don't check them again
.We wanted to check if in the event that history is not defined, we can save running time
You can see in the performance graph that with a small input, the use of history slows down .the running of the algorithm, but with larger inputs, the history improves the performance

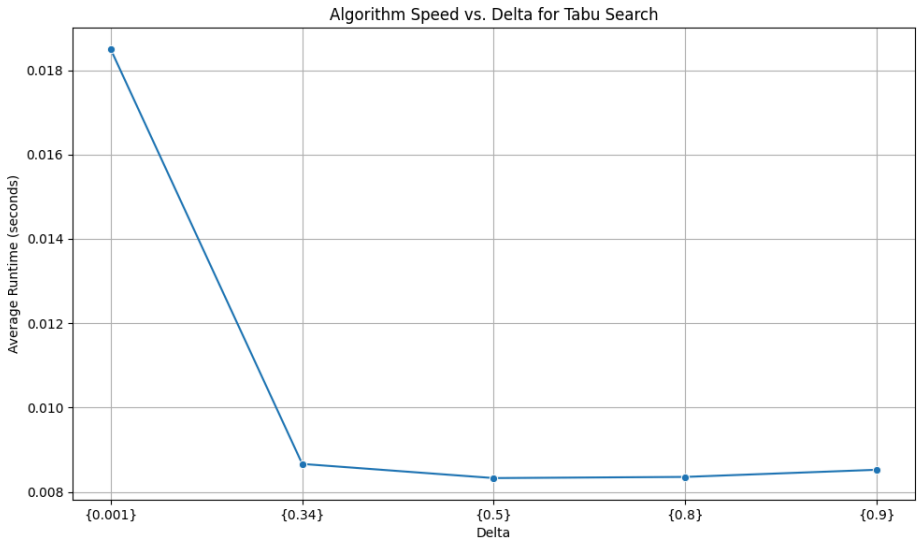


The effect of beta β and delta δ on the running time

.Delta represents the size of the price change in each iteration

You can see that the delta is very small (for example 0.001) the running time is long, because the progress in the price vector is very slow.

.It seems that for a delta equal to 0.5 the running time is the shortest

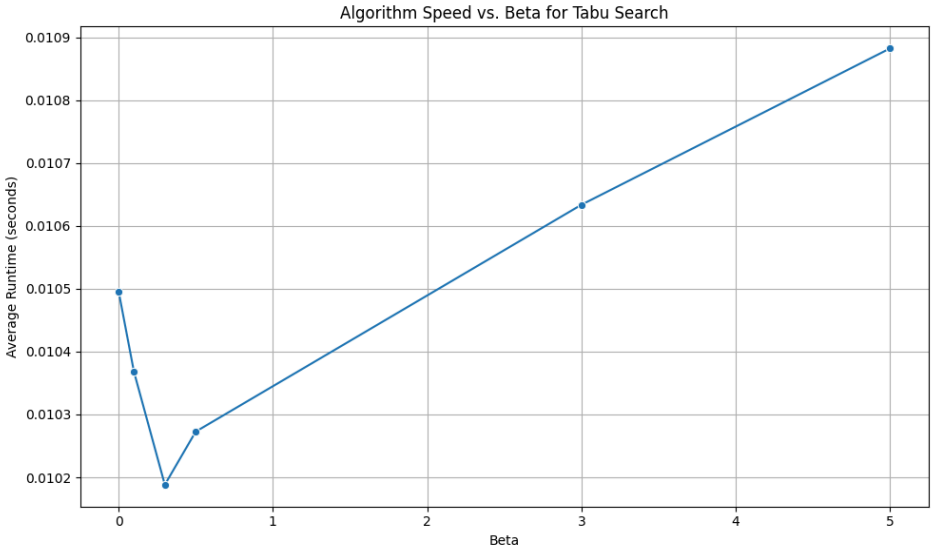


.Beta represents the range of distribution of course prices and student budgets

.You can see in the graph that as beta increases, the running time increases

The search for neighbors in individual price adjustments is performed in jumps of a very small number, therefore the larger the price range, the longer the algorithm takes to find a suitable division

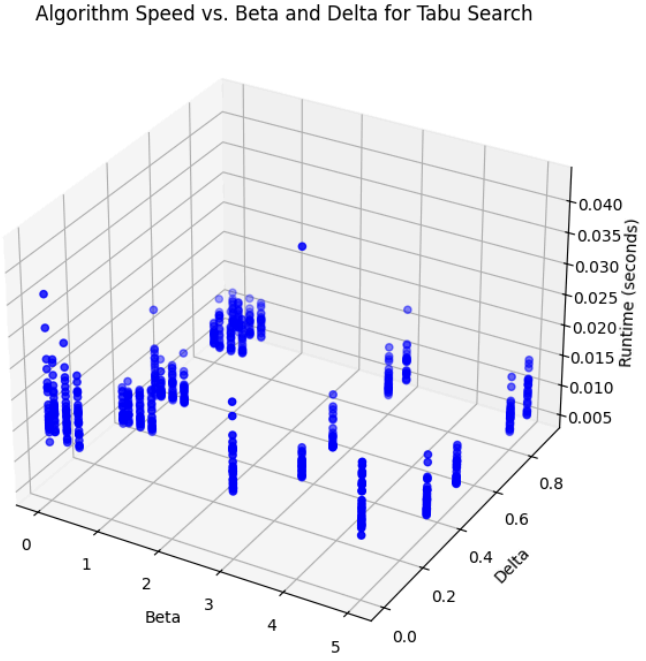
.



The last graph shows the running time of the algorithm as a function of beta and delta together.

The results obtained are:

```
Best beta: 0.001
Best delta: {0.34}
```



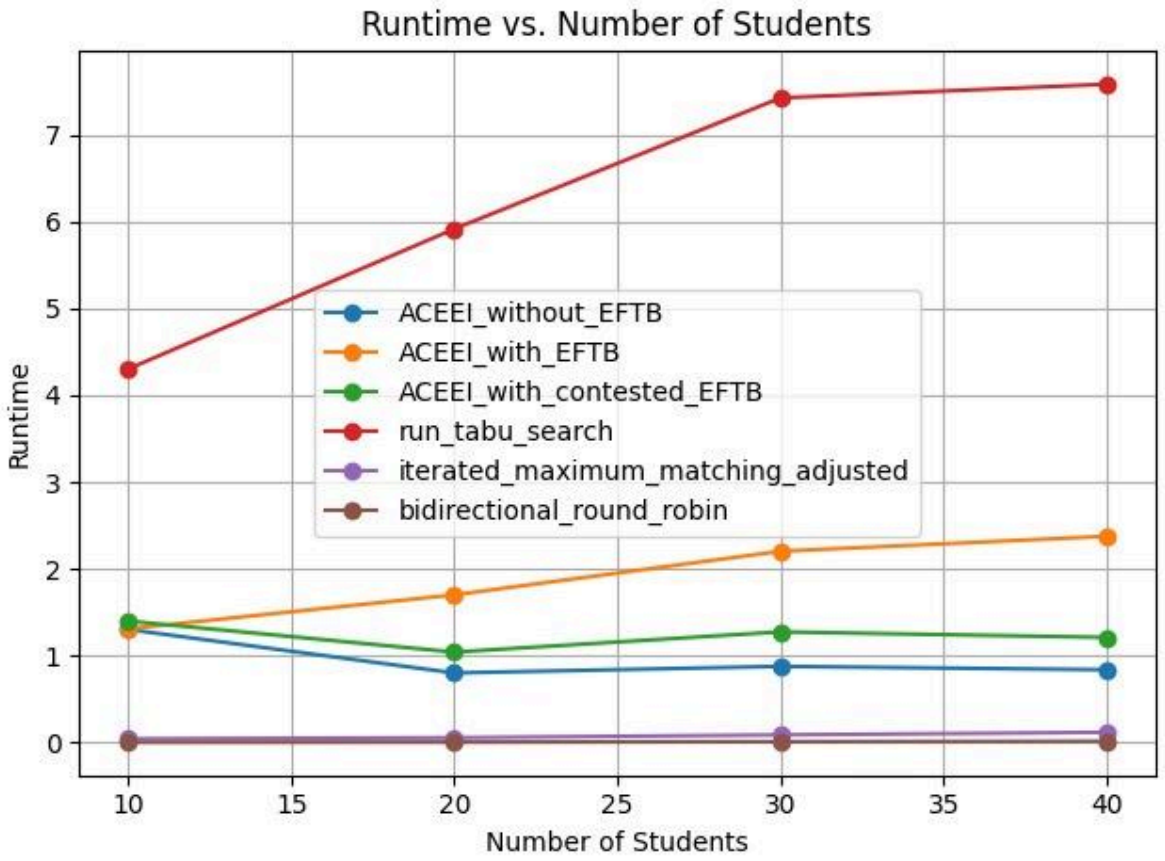
Comparison with existing algorithms

Running time comparison

We performed a performance comparison on the following algorithms:

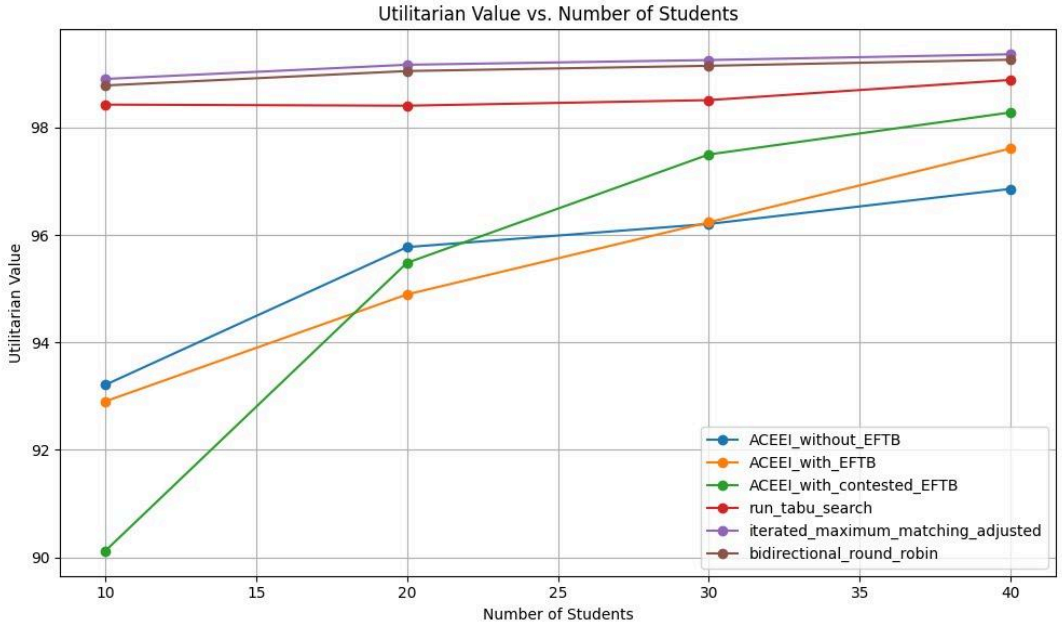
ACEEI_without_EFTB,
ACEEI_with_EFTB,
ACEEI_with_contested_EFTB,
run_tabu_search,
crs.iterated_maximum_matching_adjusted,
crs.bidirectional_round_robin

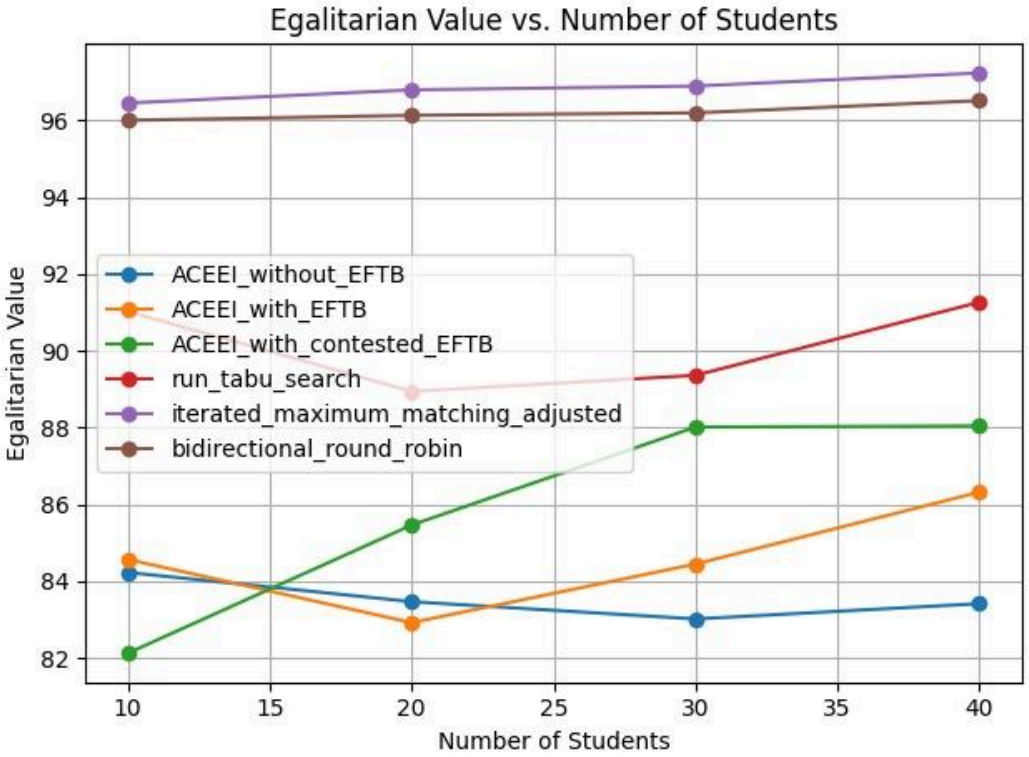
Unfortunately, it can be seen that our algorithms are slower compared to the other algorithms.
On the other hand, the algorithms we developed have a great advantage in that they perform division without jealousy, and therefore they increase the satisfaction of the students.

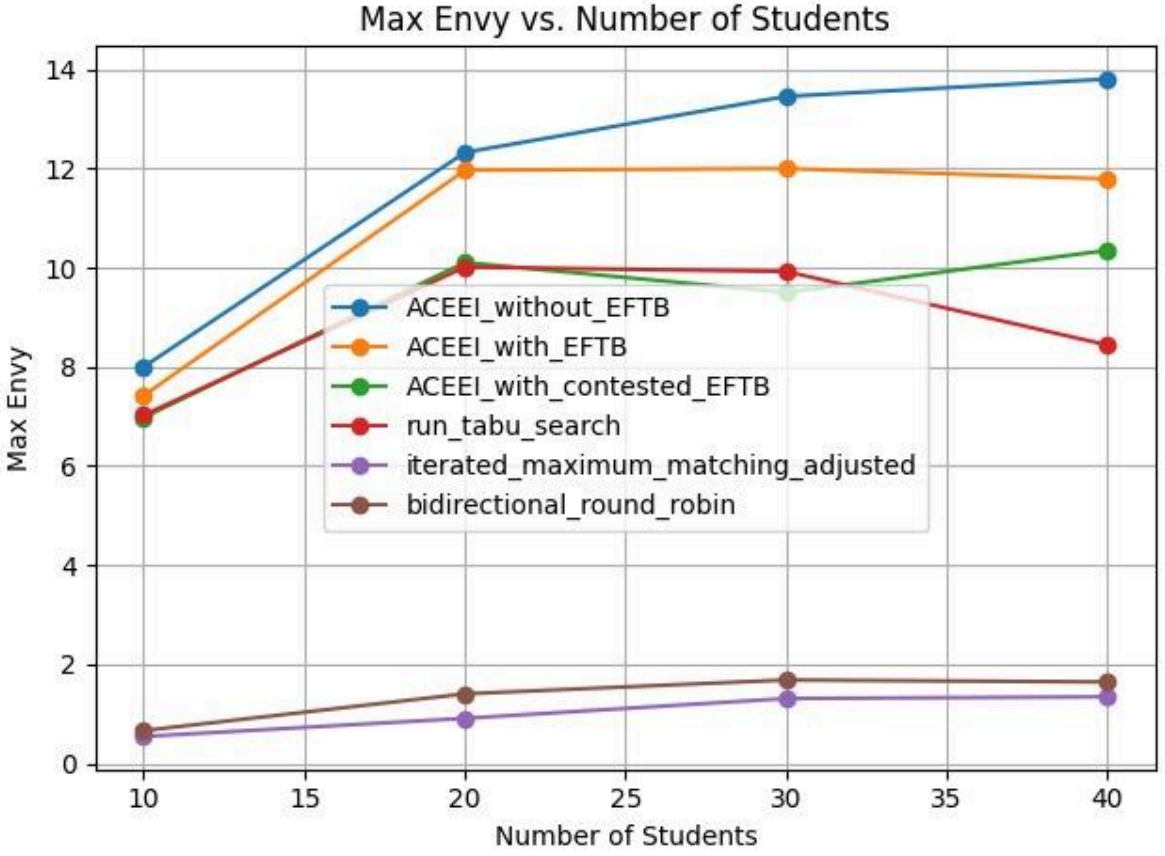


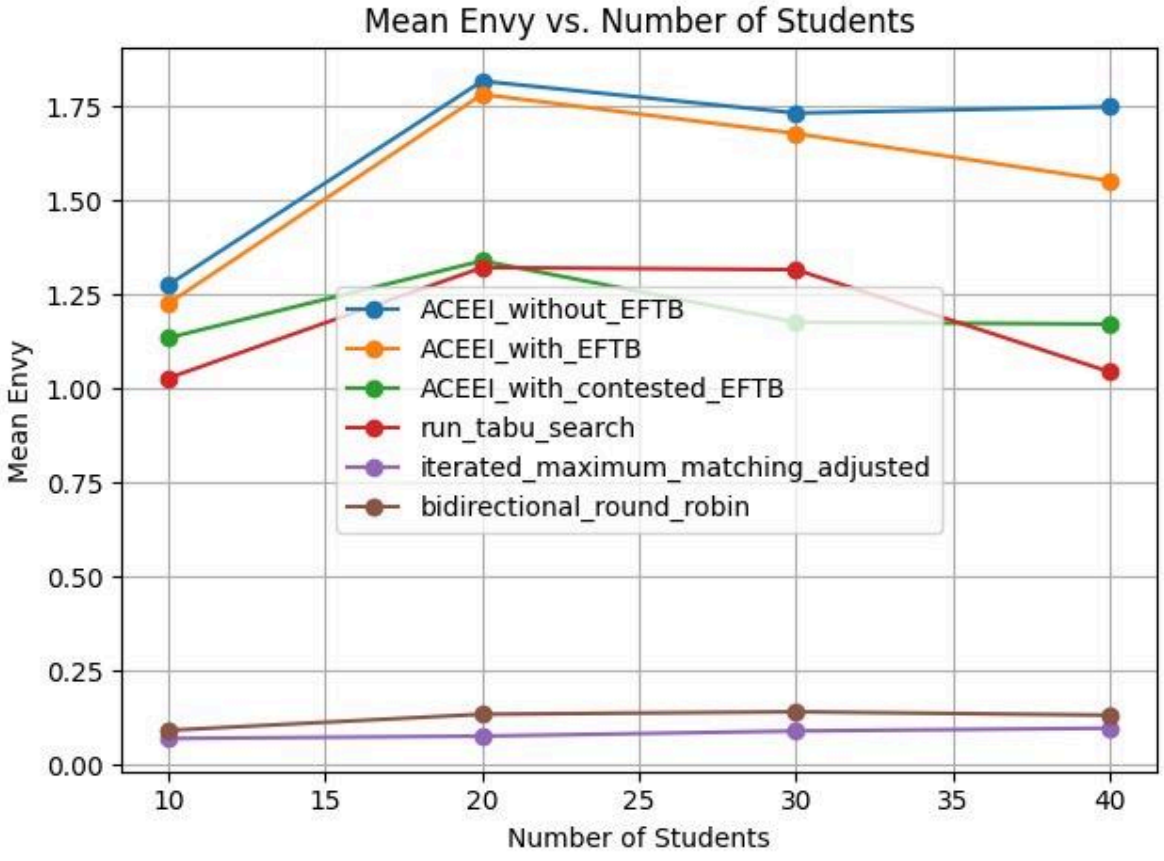
Comparison of division dimensions

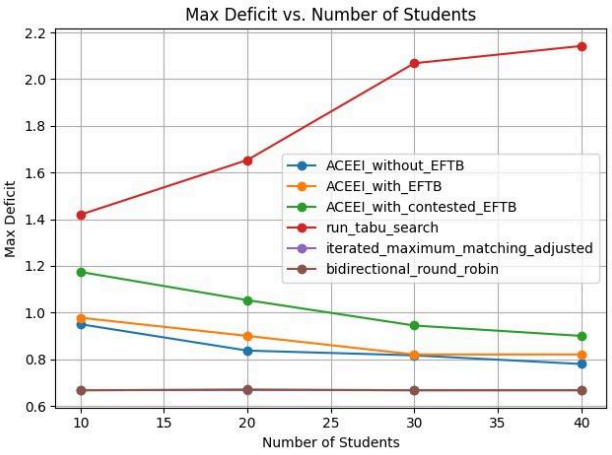
From the comparison between the algorithms in the article and those in our project, it appears that there is no significant improvement in the values obtained. It seems that when we measure the envy index as defined in the paper, our algorithms achieve good results. However, when compared according to the common definition of envy, the performance of our algorithms is relatively low.

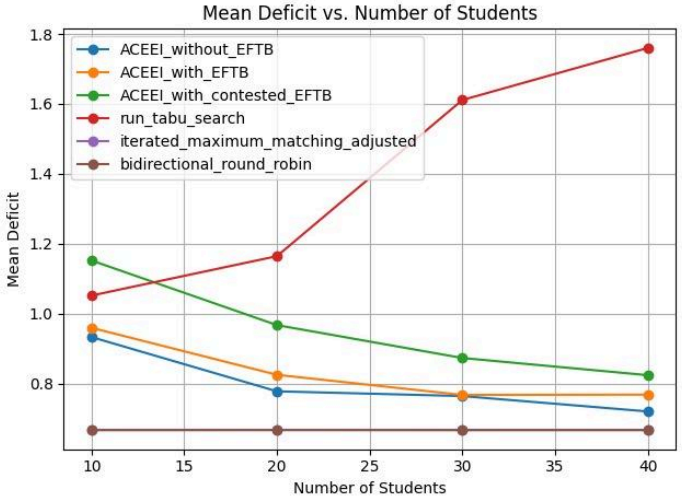





Max Envy vs. Number of Students
Max Envy
Number of Students
ACEEI_without_EFTB
ACEEI_with_EFTB
ACEEI_with_contested_EFTB
run_tabu_search
iterated_maximum_matching_adjusted
bidirectional_round_robin
0
2
4
6
8
10
12
14
10
15
20
25
30
35
40


Mean Envy vs. Number of Students
Mean Envy
Number of Students
ACEEI_without_EFTB
ACEEI_with_EFTB
ACEEI_with_contested_EFTB
run_tabu_search
iterated_maximum_matching_adjusted
bidirectional_round_robin
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
10
15
20
25
30
35
40

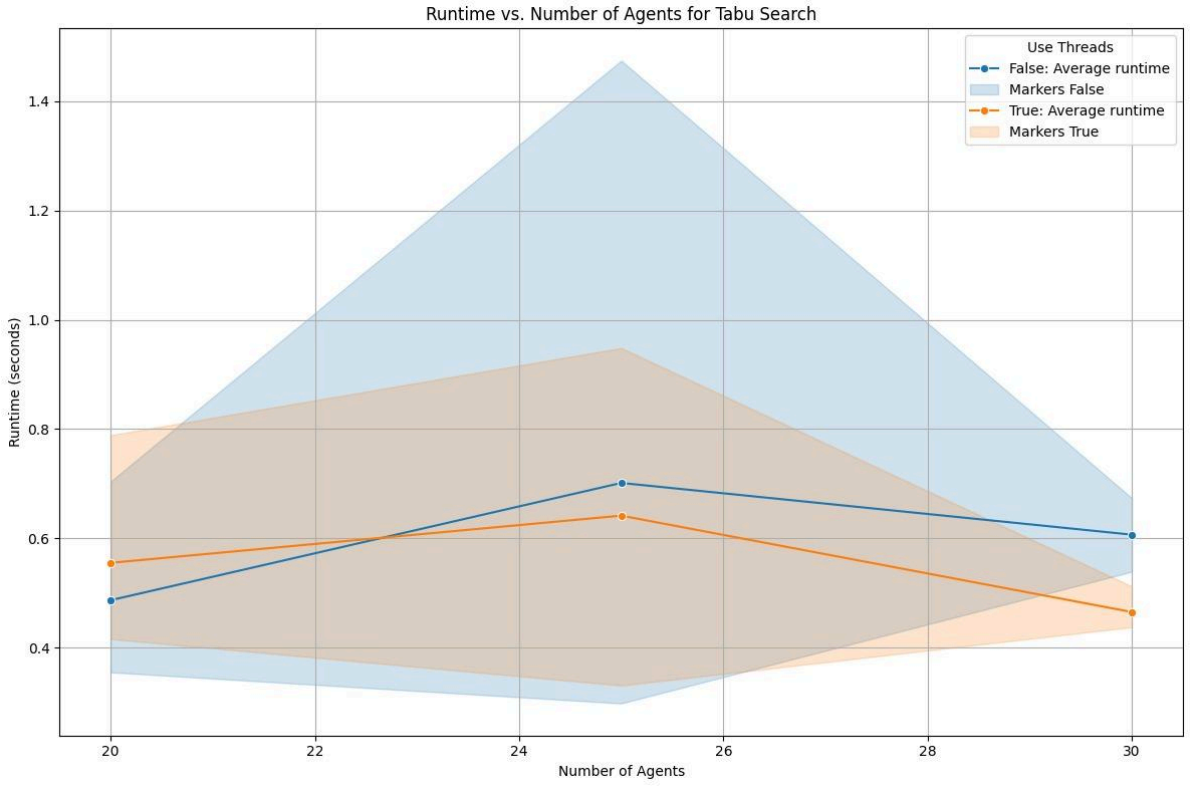

Max Deficit vs. Number of Students
Max Deficit
Number of Students
ACEEI_without_EFTB
ACEEI_with_EFTB
ACEEI_with_contested_EFTB
run_tabu_search
iterated_maximum_matching_adjusted
bidirectional_round_robin


Mean Deficit vs. Number of Students
Mean Deficit
Number of Students
ACEEI_without_EFTB
ACEEI_with_EFTB
ACEEI_with_contested_EFTB
run_tabu_search
iterated_maximum_matching_adjusted
bidirectional_round_robin

Performance improvement

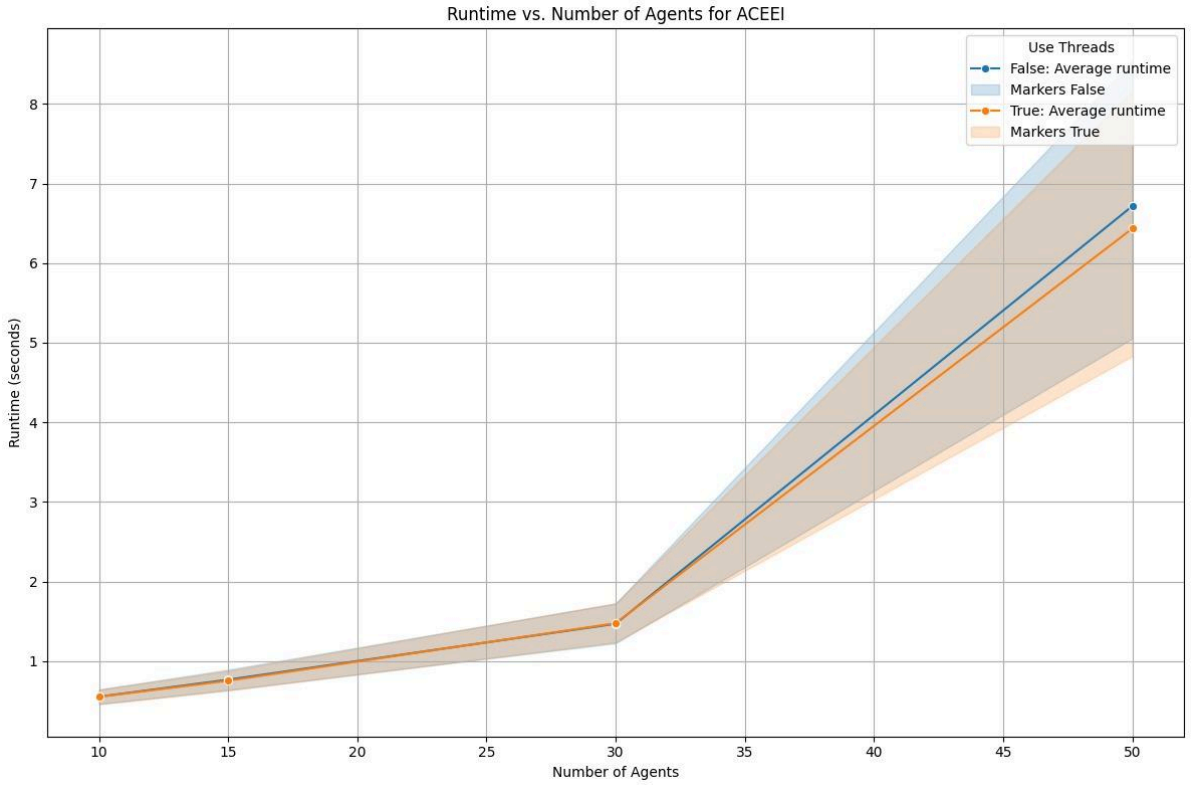
Threads in Tabu search

The function student_best_bundles returns for each student all the baskets he can take with his maximum benefit.
We have memorized the implementation of this function so that for each student it will be calculated in a separate order.
We did see an improvement in the running time:



Threads in ACEEI

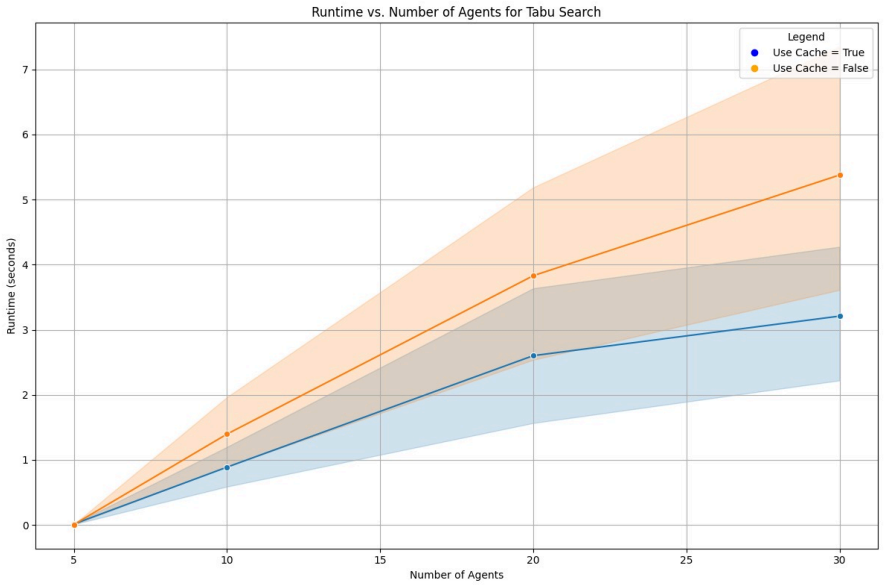
It can be seen that adding the annoyances improved the running time:



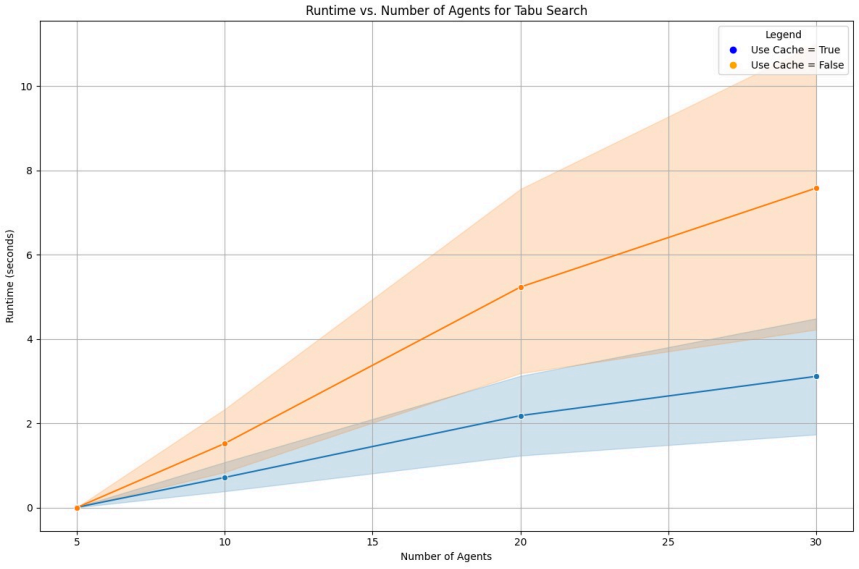

```
Best use_threads: True
Corresponding Runtime: 0.014205799998308 seconds
```

Cache in Tabu Search

Since we use this function multiple times, the same calculations may be performed over and over again.
Therefore the cache will save us the repeated calculations.
Indeed, we see that the larger the number of students, the cache has an effect in lowering the running time.



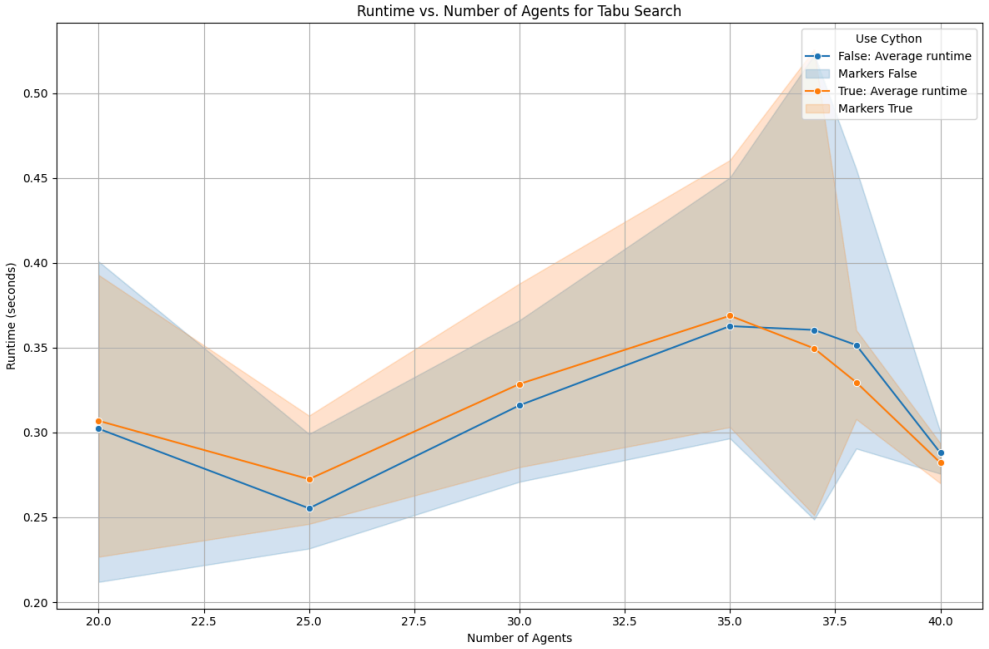
Another improvement we made is moving the calculation of the combinations to the main function, so that the calculation will be performed only once during the running of the algorithm, and not every time the student_best_bundles function is run.
The following graph shows the runtime differences when the improved function both uses the cache and the combination dictionary is passed to it, and the original function does not use the cache, and in each run it recalculates all the combinations for each student.

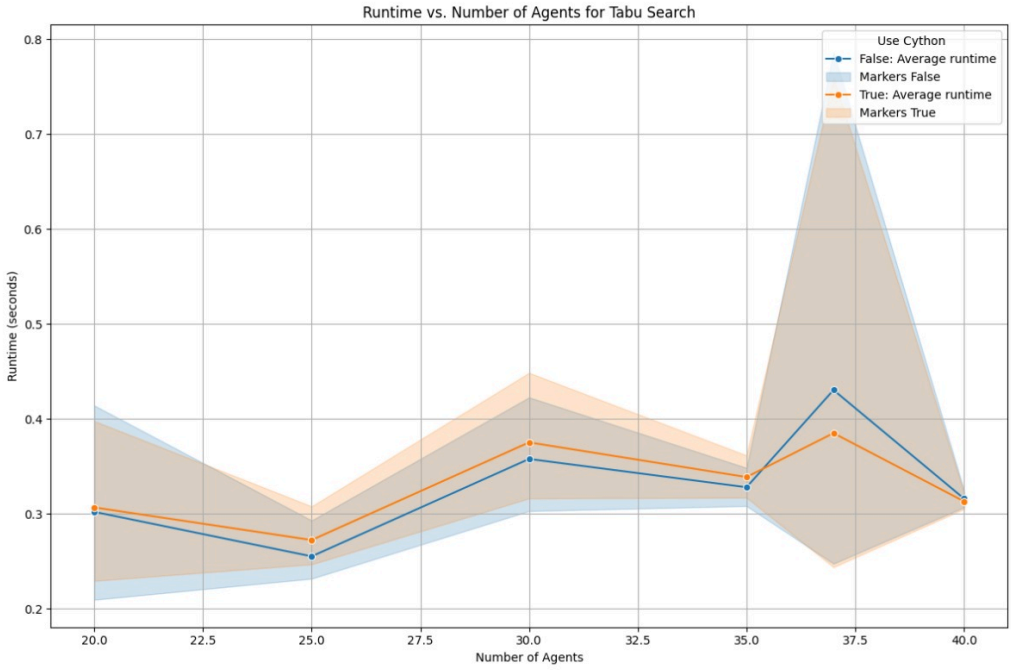


C code in Tabu Search

The calculation in the function takes a lot of time, so by changing to C code we will get an improvement in the running time.

It can be seen that as the number of students increases, the use of the C code contributes more to the runtime and reduces it.





Research conclusions

In light of everything detailed in the performance comparison section, we came to the conclusion that the algorithms in the article give better results according to their definition of envy, but not according to the usual definition of envy,
We contacted the authors and are now waiting for a reply.
Attached is the email to the authors:

Dear Authors,

After working extensively on the algorithms and conducting a thorough comparison of the algorithm discussed in the paper with alternative algorithms, we observed that your algorithms find allocations with a smaller envy according to the definition of envy in the paper, that depends on budget. However, surprisingly, the "standard" envy (that does not consider the budget) is higher in your algorithms. Please see the plots below. Does it make sense?

Please let us know your thoughts

Thanx