Matching under constraints

by Bruno Rabelo and Maurício Lima.

In this project, we are analyzing a matching problem between a set of students I and a set of schools S - which is a many to one problem. Each student $i \in I$ has a list of preference for $S \cup \emptyset$, denoted by $\succ_i$. The students have two characteristics: a group $g_i \in G$ and a cost $w_i \in \mathbb{R}_+$. Similarly, each school $s \in S$ has a list of preference for I, denoted by $\succ_s$. The schools have one characteristic: a constraint $F_s \in 2^I$. We will denote μ for the standard notion of matching. We will define the stability of a matching as being: feasible, individual rational, fair and non wasteful.

For this first part, we are implementing the algorithms ourselves, mainly following the famous idea of Gale and Shapley.

Task 1: *the above notion of stability coincides with the standard notion from Gale and Shapley.*

We will prove the equivalence in both ways. First we assume that it's stable as defined in the introduction. As it's stable, it's fair, so no student has a justified envy towards any other, which is the definition of stability for Gale and Shapely.
We now assume that it's stable by the notions from Gale and Shapely. As we saw before, it's also fair. By the construction of the Gale Shapely problem, we can't match a student with a school that doesn't support him - as we will see in the next task, it's doable by multiplying each school by its capacity and then doing single matches with the students. So it's feasible. Using the same argument, it's also non-wasteful, because the Gale Shapely stability optimizes the matching choices, so if a student is whiling to go to a school and he can't, it must be that the school is already fully matched, hence adding the student wouldn't respect the constraint and it wouldn't be feasible. Finally, it's also individually rational. If it's possible, it's also going to be individual rational by optimality of Gale Shapley algorithm.

Task 2: *acceptance algorithm for the school admission problem with capacity constraints (i.e., the many-to-one matching problem).*

The idea to solve the problem is to use the Gale Shapley algorithm with an adapted set of lists of preferences. For this, we duplicate the preference of each school s by q_s in each students preference lists. Similarly we also duplicate the lists of preferences of each school s by q_s. It's important to say that, we do all of this with the creation of a sub-class of the class school. For example, if $s_1 \in S$ has a capacity of two, we will be creating the sub-schools s_{10} and s_{11}. Like this, the most desirable student for the school s_1 will be assigned to the sub-school s_{10} (if he also wishes so). The algorithm has the same complexity of Gale Shapley, as the adaptation operations are doable in time $O(|I| \cdot |S|)$. So in the end we still have a $O(|I| \cdot |S|)$ time of execution.

Task 3: *modification of the algorithm to respect maximum quotas constraints*.

The idea now is to make an additional condition for checking if the group quota is still letting the match to happen. If it's not, we simply pass to the next school in the preference list of the student. It's important to notice that this can just happen if the student is less desirable for the school than all the current students matched to it. It's indeed the case because, before all, we continue to check the justified envies, so that if a student is matching to a sub-school that has no students matched, this is the last sub-school non-matched and so this is the less current desirable student for this school, like described in the last task. In general, just this verification will not change the order of the algorithm, but it will surely get slower.

Task 4: *tests*.

All the results with more details can be found in the Jupiter notebook that we are attaching to this project.

Instance 1:

$I = \{i_1, i_2, i_3, i_4\}, \quad \succ_{i_1}: s_1, s_2, \quad \succ_{i_2}: s_1, s_2, \quad \succ_{i_3}: s_2, s_1, \quad \succ_{i_4}: s_2, s_1.$

$S = \{s_1, s_2\}, \quad \succ_{s_1}: i_4, i_3, i_2, i_1, \quad \succ_{s_2}: i_4, i_3, i_2, i_1.$

$G = \{A, B\}, \quad \{i_1, i_2, i_3\} \in A, \quad i_4 \in B.$

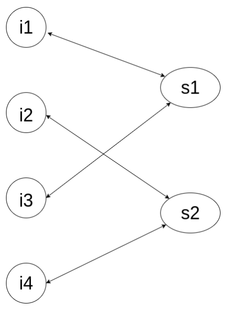
$w_1 = w_2 = w_3 = w_4 = 10.$

$Capacities = quotas(A, B) = 2.$

Results:

Both algorithms for **task 2** and **task 3** obtained the same results as follow:

$\mu_{s_1} = \{i_1, i_3\}$ and $\mu_{s_2} = \{i_2, i_4\}$



Instance 2:

$I = \{i_1, \ldots, i_n\}, \quad \succ_i: s_1, s_2, \ or \ \succ_i: s_2, s_1$ -with probability 50% for each for all the students.

$S = \{s_1, s_2\}$ - with the preference of each school defined by the quality of each student $W_i = \hat{W}_i + \epsilon_{is}$, where $\hat{W}_i$ and $\epsilon_i s$ are two probability functions that follows the standard normal distribution (index i for each student and index s for each school).

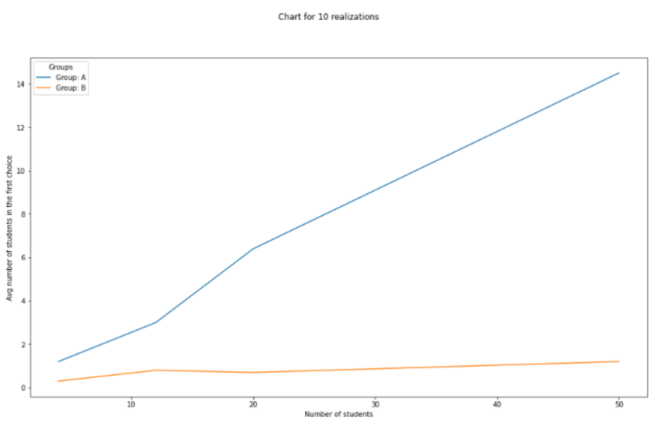
$G = \{A, B\}, \quad \{i_1, \ldots, i_m\} \in A$, where $m = \lfloor 9n/10 \rfloor$ and the other students are in B.

$w_A = 1$ and $w_B = 10$.

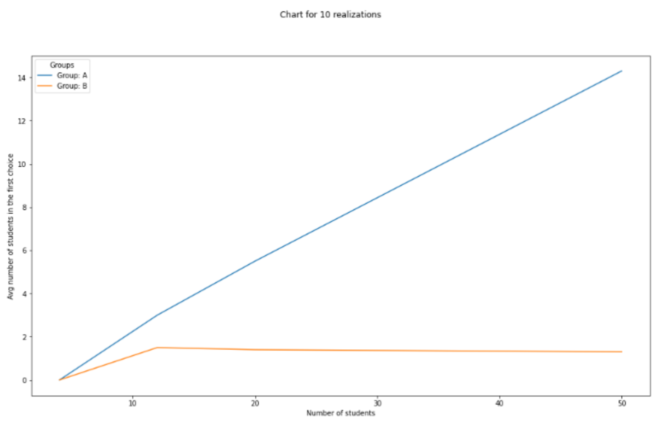
Each school has a capacity of $\lfloor n/4 \rfloor$, and a quota of $\lfloor 0.9n/4 \rfloor$ for each group.

Results:

Task 2:



Task 3:



The results suggest that both tasks performs similarly for 2 groups, but **task 3** tends to give more chances for students of group B when the number of students is not so big, which confirms the fact that this algorithm make the quotas worth it.

Instance 3:

$I = \{i_1, \ldots, i_n\}, \quad \succ_i: s_1, s_2,\ or\ \succ_i: s_2, s_1$ -with probability 50% for each for all the students.

$S = \{s_1, s_2\}$ - with the preference of each school defined by the quality of each student $W_i = \hat{W}_i + \epsilon_{is}$, where $\hat{W}_i$ and $\epsilon_i s$ are two probability functions that follows the standard normal distribution (index i for each student and index s for each school).

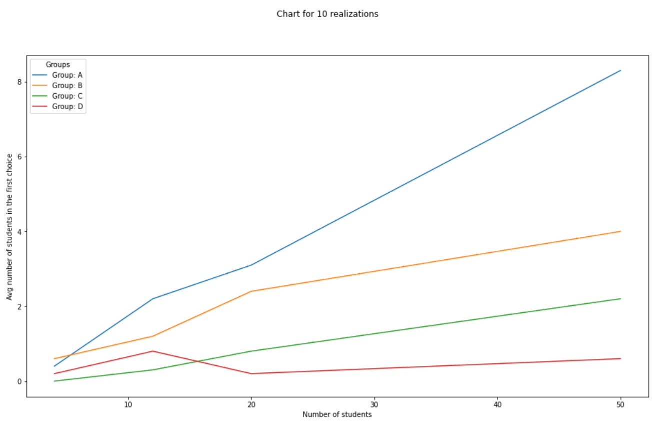
$G = \{A, B\}, \ \{i_1, \ldots, i_m\} \in A$, where $m = \lfloor 9n/10 \rfloor$ and the other students are in B.

$w_A = 1$ and $w_B = 10$.

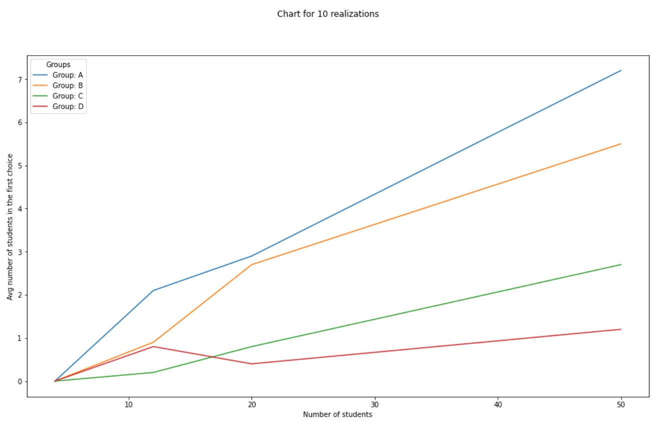
Each school has a capacity of $\lfloor n/4 \rfloor$, and a quota of $\lfloor 0.9n/4 \rfloor$ for each group.

Results:

Task 2:



Task 3:



For 4 groups now, the results suggest that **task 3** tend to equalize the first choice students then in **task 2**. Indeed, we have more students of groups D and B, which are minority when compared with group A. We still have not improved the group C, but nothing is perfect!

For this second part, we are applying a 4/5-rule to the system of quotas. We will approach it with the idea of modifying the schools preference lists such that, for any rank k, the subset of students ranked k or better satisfy the 4/5-rule up to one unit.

Task 5: *implementation of the algorithm.*

We are just creating a function that compares the group percentage of the student of rank k in all the students that are better or equally placed than him with the 4/5-rule up to one limits. If it works well, we can pass to the next rank. If not, we will change it with a student of another group, with the goal of equalizing things.

The order of the global time of execution will not change, since we are changing the set of preferences in an $O(n^2)$ order in the worst case, but it tends to perform a lot better in general.

Task 6: *test.*

Instance 1:

Matches: i_3(A): s_1, i_1(A): s_1, i_4(B): s_2, $i2$(A): s_2

Ratios: A: {s_1: 4/3, s_2: 2/3}, B: {s_1: 0, s_2: 2}

Algorithm runs well.

Instance 2

Students: i_0(A), i_1(A), $i2$(A), i_3(A), i_4(A), i_5(A), i_6(A), i_7(A), i_8(A), i_9(A), i_{10}(A), i_{11}(A), i_{12}(A), i_{13}(A), i_{14}(A), i_{15}(A), i_{16}(A), i_{17}(A), i_{18}(B), i_{19}(B)

Matches: i_6(A): : s_2, i_4(A): None, i_9(A): None, i_7(A): None, i_0(A): None, i_{14}(A): None, i_{15}(A): None, i_{16}(A): None, i_3(A): None, i_{18}(B): : s_2, i_{11}(A): : s_2, i_{12}(A): : s_2, i_{17}(A): : s_2, i_8(A): : s_1, i_1(A): : s_1, i_{13}(A): : s_1, i_5(A): : s_1, i_2(A): : s_1, i_{10} (A): None, i_{19}(B): None

Ratios: A: {s_2: 8/9, s_1: 10/9}, 'B': {s_2: 2, s_1: 0}}

Algorithm runs well.

Instance 3

Students: i_0(A), i_1(A), i_2(A), i_3(A), i_4(A), i_5(A), i_6(A), i_7(A), i_8(A), i_9(A), i_{10}(B), i_{11}(B), i_{12}(B), i_{13}(B), i_{14}(B), i_{15}(B), i_{16}(C), i_{17}(C), i_{18}(C), i_{19}(D)

Matches: i_2(A): s_2, i_7(A): None, i_5(A): None, i_0(A): None, i_{13}(B): s_2, i_{11}(B): : s_2, i_3(A): None, i_{14}(B): None, i_{12}(B): None, i_{16}(C): None, i_{17}(C): None, i_{18}(C): s_1, i_6(A): None, i_{19}(D): s_2, i_9(A): s_2, i_1(A): s_1, i_{10}(B): s_1, i_4(A): s_1, i_8(A): $s1$, i_{15}(B): None}

Ratios: A: {s_2: 0.8, s_1: 1.2}, B: {s_2: 4/3, s_1: 2/3}, C: {s_2: 0, s_1: 4/3}, D: {s_2: 4, s_1: 0}

We have an error in group D because of the way we implemented the rule. We just check the "new" student in the set, but it could change the percentage of other groups, so we would have to verify it to all the students in the set better or equal to rank k. But it would take the code to run n times slower and we can sometimes never have an result, so we guarded as it is. In the end, the error is really not big and the algorithm works pretty well for its purpose.

For this third and last part we are applying

Task 7: *implementation of the algorithm from Kamada and Kojima with a feasibility function encoding arbitrary feasible sets.*

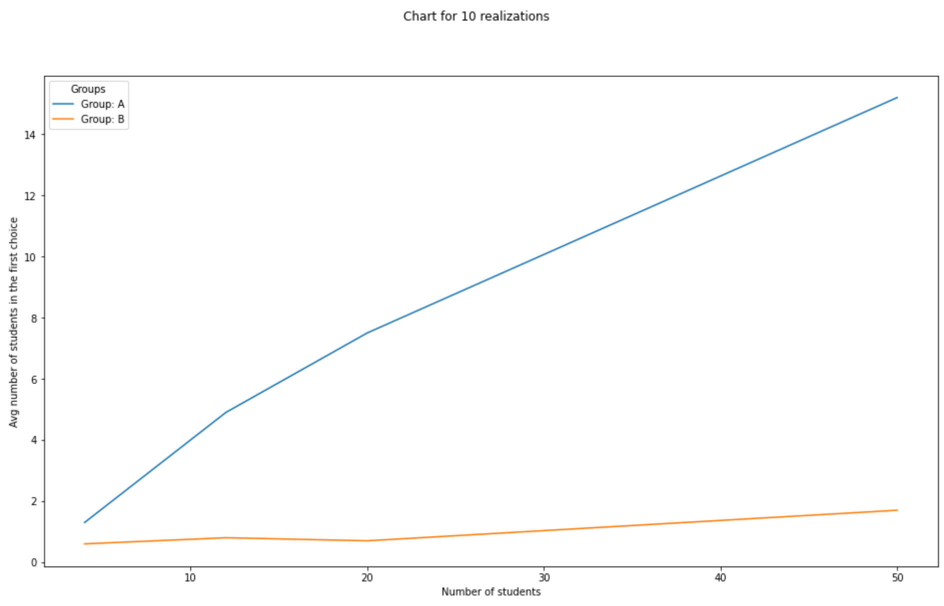
We will follow the algorithm described in the paper by creating a set P and iterating the mapping T for each schhol $s \in S$ to find the fixed point of each. By the main theorem of the researchers, the demands of the fixed points for each school $D_s(p)$ will give us the matching μ_s for all $s \in S$.

This algorithm has an global time of execution of order $O(|I|^2 \cdot |S|^2)$.

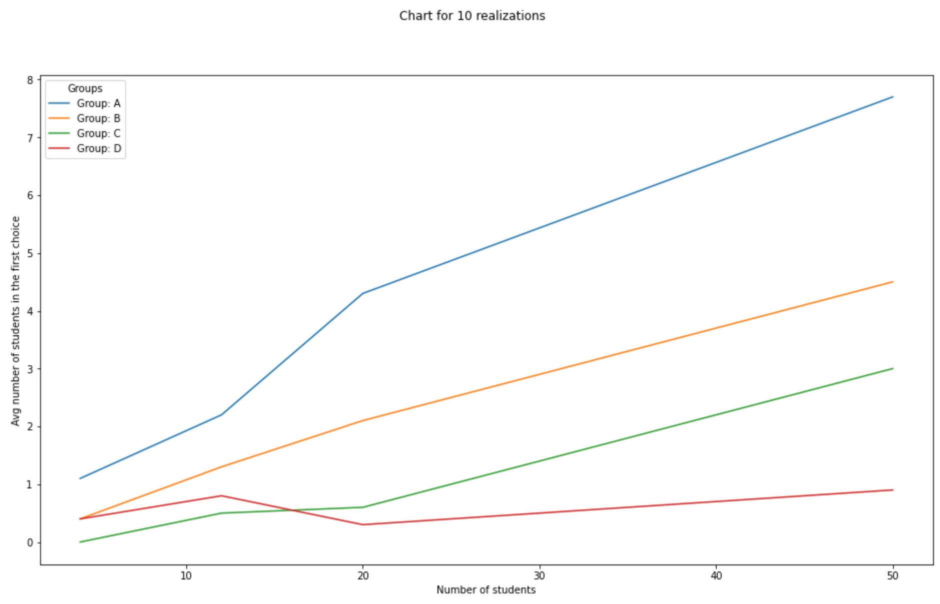
Task 8: *testing with only capacity constraints*.

Instance 1: same results as in **task 3**.

Instance 2: same results in general as in **task 2** algorithm.



Instance 3: same results in general as in **task 2** algorithm.

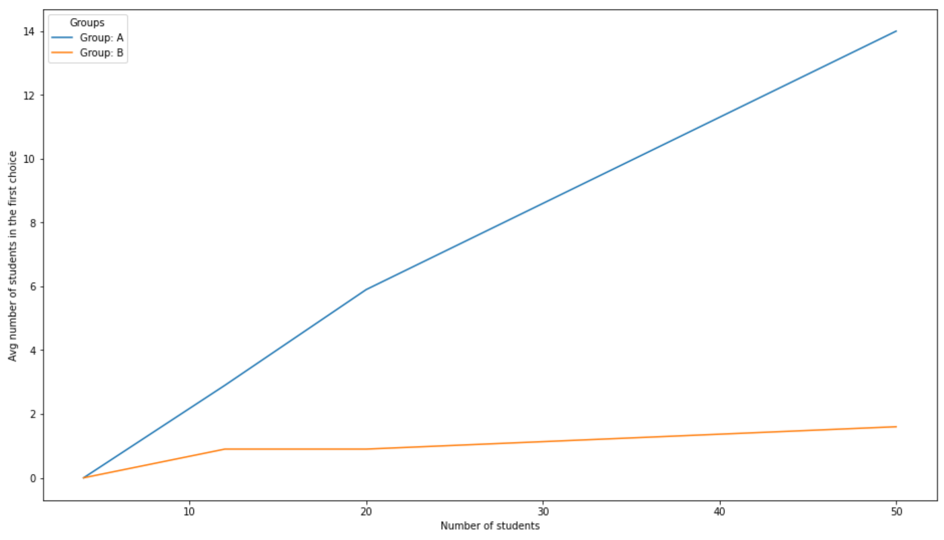


Task 9: *adding the maximum quotas constraints*

Instance 1: same results as in **task 3**.

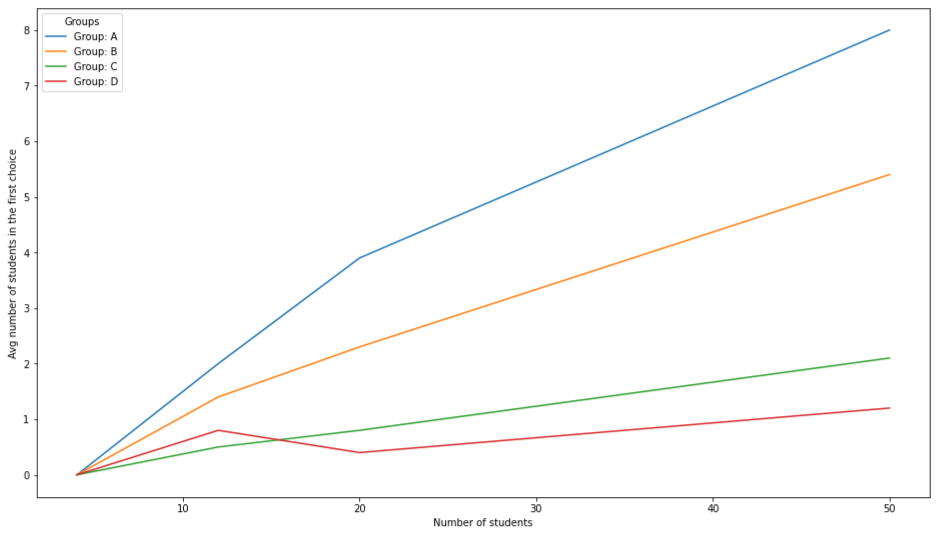
Instance 2: same results in general as in **task 3** algorithm.

Chart for 10 realizations



Instance 3: same results in general as in **task 3** algorithm.

Chart for 10 realizations



In general, the algorithm tends a perform as well as the others, even if we are ignoring the non-wastefulness. It can be that for bigger set of students it performs worse, bu results are quite good until 50 students (for higher numbers, our pc gets slow).

Task 10: *what happens if we try to run the algorithm with the 4/5-rule constraint?*

It will just doesn't work. The algorithm doesn't find a fixed point because of the way we altered the set of preferences of S.