Match Making Algorithm for solving the Classroom Assignment problem

Vazquez Corte Mario*

E-mail: mario.mentat@gmail.com

Abstract

This paper proposes a novel approach to solve the *classroom assignment problem*, which is a well known NP-Hard multi-objective optimization problem, as a *match-making problem*. In order to generate the preferences we ask the professors and students to enunciate their own preferences over possible set of classrooms. Furthermore, we propose a democratic mechanism to aggregate students' preferences. The mechanism can follow the standard democratic approach and aggregate votes or impose any type of voting system like Borda count. We show that any mechanism that aggregates preferences or represents preferences can help model the problem as a match-making one as long as it satisfies some assumptions. Mainly if preferences can be represented as a matrix, the problem can be solved with a match-making algorithm. We proceed to analyze the complexity of the algorithm and the normative implications of our approach. Finally, we propose a tie-breaker method that helps minimize the overlap of the courses for the alumnae. The complexity of our algorithm is $O(n^4)$.

1 Introduction

The *classroom assignment problem*[1], or *timetabling problem*[2], is about scheduling a set of lectures of a school in a prefixed period of time, such that some constraints are satisfied (for example, two classes should not meet simultaneously in the same room or a class should not meet in two different rooms[1]).

This problem is faced every scholar period by schools, when they schedule their activities. Many approaches have been taken, since 1960[3], in order to provide a solution for this NP-hard problem[4], specially from people of operation research, computer science and artificial intelligence fields. Also, the problem that checks whether a solution exist, *underlying problem*, is NP-complete[2].

Most of solutions for this problem use *heuristic* or *greedy*[5] *centralized* algorithms. It means that the scheduling of classrooms is left to a central entity (either the administration, the department or other central organism). Our approach tries to incorporate an other point of view. Economic literature and tradition have contrasted the centralized approach against the decentralized one. Adam Smith,Friedrich Hayek, John M. Keynes, Amartya Sen and Kenneth Arrow are part of this tradition[6 7 8]. It is common to implement de-centralized solutions to complex problems[9], like market economies (capitalism) and centraliced economies (socialism).

In this work, we propose an algorithm that models students as consumers and professors as suppliers, or in terms of matching, literature students and professors are proposers and accepters. In order to aggregate the preferences from students, a *democratic* mechanism is analized. Although, any aggregation mechanism will work as long as it satisfies the assumptions described in the next sections. The algorithm belongs to the greedy and divide and conquer categories. The multi-objective nature, for us, can be resumed in the preferences of the agents, since each agent has his own muli-objective criteria and thus is assumed to be a better representation of possibly conflicting objectives.

The contributions of this paper are as follows:

- We construct a general framework and state the necessary assumptions needed to aggregate student and professor's preferences in order to obtain rational preferences.
- We provide an algorithm for mapping the *classromm-assigment problem* to the *matching-*

problem and theoretically prove it has a complexity of $O(n^4)$.

- It is showed that any aggregation mechanism preserves its core properties under the algorithm we proposed.

The remaining sections are organized as follows. Section 2 gives a literature review of alternative approaches. Section 3 formalizes the notion of preferences, the *matching problem* and the Gale Shapley Algorithm for solving the *marriage problem*. Section 4 describes how transform the *classroom assignment problem* to a *match-making problem*. Section 5 analyzes the complexity of the algorithm proposed in Section 4. Section 6 gives a brief description of how the algorithm would behave under indifference conditions. Section 7 provides some simulations of the algorithm and compares it against the approaches used in Carter & Tobin,[1] and Yoshikawa[10] .Section 8 is a brief discussion from a normative point of view. Section 9 deals with future work. Section 10 gives some remarks and conclusions.

2 Related Work

The *timetabling problem* is usually solved using a policy of first-come, first-served[1], in a way that those classrooms are not changed from one year to the next. If this solution is not "enough", those systems are adjusted *manually* by the centralized operator, using a kind of greedy approach.

On the other hand, for automated solutions, a classification of them is proposed by Schaerf[2]:

1. **Search problem**.- When the problem is about finding any timetable that satisfies all the constraints given.

2. **Optimization problem**.- When the problem satisfies all the *hard* constraints, and maximized a given objective function which embeds the *soft* constraints.

One approach taken by Carter, is solving each day of the assignment problem through linear program [1]. Other approaches are about transforming this problem into a graph coloring problem with a time complexity of $O(2^n n)$ [11,12] (each lecture is associated to a vertex in a graph, there is an edge between a pair of lectures that cannot be scheduled at the same time and each color represents a different period).

Yoshikawa et al. [10] proposed a solution for this problem as an optimization function with constraints, and an associated penalty if the constraints are violated. Therefore, the objective is to minimize the overall penalty cost.

A genetic algorithm has been used [13] (and they were improved with local search algorithms), and most recently, algorithms that uses some hyper-heuristic techniques were proposed [14] (which select some heuristics, in order to search a solution given a set of low level heuristics).

Other approaches include *tabu search* [15–18] or *simulated annealing* [19,20] meta-heuristics. Kannan et al. [21] used a theoretic approach that decomposed a graph and apply randomized heuristics to solve this problem. On the other hand, Moura and Scaraficci [22] proposed a *greedy randomized adaptive search procedure* (GRASP) heuristic, and Pillay [23]used an evolutionary algorithm that used hyper-heuristics selection methods.

3 Framework

3.1 Rationality and Preferences

The standard model of rational choice centers around a decision maker (DM) who maximizes a given preference in every menu.

Definition 1 *A DM is rational if and only if her preferences are complete and transitive*

Definition 2 *A binary relation $\succeq$ on X is:*

(1) ***Complete****:= if for all $x, y \in X$, either $x \succeq y$ or $y \succeq x$.*

(2) ***Transitive****:= if for all $x, y, z \in X$, and $x \succeq y$ and $y \succeq z$, then $x \succeq z$*

In loose terms, a DM or agent is rational if he is capable of "comparing" any two alternatives over the set of possible alternatives, and if some alternative x is preferred over y, and y over z, then must be that x is preferred over z. This rationality assumptions are the theoretical base for any standard matching-algorithm, thus we are looking to create preferences that satisfy these two key conditions. In order to formulate the problem in terms of a match-making algorithm, we need to have individual rational preferences for each set of actors. In our scenario the actors are classes and classrooms, represented by the aggregated preferences of professors and students respectively.

Note that from this definitions is not clear how to aggregate preferences or how two compare them. For example, if an agent prefers a pizza over a hamburger, and an other agent prefers hamburger over pizza, which item should the aggregate preferences favor? It is a well know result that aggregating individual preferences does not lead to rationality[24]. One possible response to this problem is democracy. Economic and political sciences have studied the normative aspect of democracy or aggregating methods. There are many ethical points of view that support many types of democracy: relative majority, absolute majority, Borda count, etc.

Another key advantage of democracy is that it helps delegate and divide problems. We formulate the Classroom problem for a matching environment, so we can take advantage of the individual capacity of agents, in this case the students and professors. In general the match-making algorithms and individuals are greedy, since the agents will always look for their best option. This means that these algorithms will behave and choose the solution that a rational greedy agent would choose. In other words, the algorithm simulates the greedy behaviour of agents at every point.

Finally, the fact that we delegate the schedule planning to the actors, by aggregating the student preferences in the form of classrooms over the set of possible classes and representing the professors' preferences as the class' preferences, is extremely important. It is easier to part the problem and let the agent optimize or decide over the set of possibilities that only concern him. The other algorithms try to solve problem at once, while our approach allows for every agent choose her favorite outcome. This means that the optimization variables and sets are much smaller for the agents, and so they find it easier to optimize over such set. The divide and conquer approach is supplemented with the ability of real optimizing agents that act greedy and possibly rational.

3.2 Matching and Marriage Problem

In computer science, mathematics and computer science the stable marriage problem (SMP) tries to find an stable solution to a matching problem between two sets of agents that have predetermined preferences. A mapping from the elements of one set to an other is called a matching[25]. The SMP can be stated as follows: Given n men and w women, with defined (rational) preferences over the set of the opposite sex, marry each woman to a different man so that neither of them would marry an other partner, and the latter is better in terms of preferences[26].

The most well known algorithm for solving this problem is "Gale-Shapley" (**GS**). The algorithm makes two key assumptions: first, the preferences are rational; and second they are strict. The former follows the standard definition of rationality previously established, and the latter does not allow for ties or indifference between two potential couples. We proceed to show the algorithm[26]:

The complexity of the algorithm is know to be $O(n)$, where n is the maximum between the number of men and women[27]. One of the key futures of the algorithm is that the stable match is *pareto-stable*, which means that no one can be better without hurting somebody.[28] This an extremely desired normative property, since it implies that there is no room for improvement and thus there shall not be any re-allocation. Nevertheless, the *pareto-quilibrium* is not unique, this means that there may be an other equilibria that satisfy the same conditions, with a different matching. A key insight about the multiplicity of *pareto-stable* equilibria

Algorithm 1 Gale Shapley Algorithm

Initialize all $m \in M$ and $s \in S$ to *free*
while $\exists$ *free* lecture m who still has a classroom s to propose to **do**
  $s \leftarrow$ first classroom on s's list to whom m has not yet proposed
  if s is *free* **then**
    (m, s) become *paired*
  else some pair (m', s) already exists
    if s prefers$m$ to m' **then**
      m' becomes free
      (m,s) become *paired*
    else
      $(m'$,s) remain *paired*

    end

is the *man-proposal* assumption. It is know that the set of agents that proposes achieve a *dominant* matching, which means that of all possible stable-matchings the proposers get the best partner. The normative implications of such result should be examined before implementing any solution.

As mentioned before the first objective must be modelling the Classroom problem as a matching problem. We will create two sets of agents with preferences, one contains the professors and the second the classrooms. The classroom's preferences will try to aggregate and represent the students' preferences.

We proceed to analyze the match-making algorithms that allow indifference or ties in the preferences.

4 Matching Approach to Classroom Assignment

Before constructing the aggregated preferences we need to introduce some assumptions and mathematical notation to guarantee the stability of the algorithm adn ease exposition. Is important to recall our definition of classroom, formalized in this section, which consists of a pair that contains a room and a corresponding schedule for such room.

Assumption 1 *: Every course has the same duration and structure.*

For example, if a course will be taught for 1h two times per week, then every other course will follow the same pattern. This assumption will be relaxed and further in Section 7. If courses have different duration, as long as the number of classes are finite the problem can be partitioned in sub problems, where each sub-problem corresponds to a particular set of classes that have the same duration.

Definition 3 *: Let T represent the set of possible hours and days the rooms are available and C the rooms available.*

Assumption.M 1 *: Each room that belongs to C is identical.*

Definition 4 *: Let $S \subseteq T \otimes C$ be the set of all available classrooms matched with corresponding times and s_k a generic element of S.*

For our mechanism, a classroom not only depends on the place, but also on the time. A classroom is different if it has a different time associated with it. For example, if classroom 101 is available at 10 a.m and 11 a.m, then we will treat the pair of classroom/hour as essentially different between them since they may host different classes. Note some pre-processing may be needed, i.e. pairing actual physical rooms with each available hour to form a class room. Note set S must respond to the structure of the courses. If courses are imparted t-hours and f-times a week, then a classroom represents a unit of t-hours and f-days. Different days will create different types of classrooms.

Definition 5 *: Let $t : S \mapsto T$, be a function that returns the hours and potentially the days that a classroom s_k represents.*

Definition 6 *: Let I be the set of all students, and i denote a generic element in I.*

Definition 7 *: Let M be the set of all courses that will be imparted, and m_n the corresponding singleton.*

Assumption 2 *: Each professor teaches only one course.*

This means that m_n also serves as a unique tag to the professor teaching the course. From now on we may refer to the preferences' of course n as the preferences' of the professor since it has a biyective relation. This assumption will be relaxed on Section 7 and further explained. If a professor needs to teach more than one course, we can add an extra round to the algorithm where we check the overlap of the professor's schedule.

Definition 8 *: Let* $P : M \mapsto S^{|S|}$ *be the function that maps the course to the preferences' over classrooms. Then* $P_n = P(m_n)$ *represents the preferences of the professor giving the course* m_n *over the possible classrooms.*

In the algorithm the preferences will be represented as a vector, where every element in $P_n[h]$ is preferred to any element in $P_n[h+1]$, so the rows or index of the vector can be directly translated to preferences. This assumption will be relaxed in Section 7, we keep it to ease exposition.

Assumption 3 *:* P_n *is rational for all courses.*

This assumption is key and potentially the must difficult to relax. If the preferences are not rational the algorithm may iterate forever, since a cycle may be present. Note that *transitivity* implies that any element in $P_n[h]$ is preferred over any element that corresponds to a higher index.

Assumption 4 *: Students know which courses will be imparted and available classrooms in advance, and are capable of providing individual demands that satisfy any criteria established by the aggregation rule.*

Definition 9 *: Let the demand for student* i *be represented by* $d^i := \{(m_n^{i1}, .., m_{n'}^{ix}), (s_k^{i1}, ..., s_{k'}^{iy})\} = \{m^i, s^i\}$*, where* $m^i \in M^x, s^i \in S^y, \forall(i), and\ x, y \in$*. Let* D *represent the set of all student demands.*

The demand is represented by two vectors of possible finite distinct lengths. The first one represents the courses the student plans to take and will be denoted by m^i, while the second represents the top y classrooms for taking courses according to his preferences.

Assumption.M 2 *$m^i[x] \neq m^i[y]$ and $t(s^i[x]) \neq t(s^i[y])$ $\forall i, x \neq y$.*

This is required to enforce only one vote per course to help the mechanism avoid overlap. If a student votes for classrooms that have the same schedule, then is encouraging classes to overlap. We can relax this assumption, but it may cause classes to overlap. *** THIS IS THE KEY PROBLEM FOR THE ALGORITHM***

Assumption.M 3 *For our mechanism we require both vectors to have the same length and $s^i[x]$ represent i's favorite classroom to take course $m^i[k]$.*

As mentioned before this simulates the democratic nature of preferences, since we can see the entry x on both vectors as the vote for that particular match of course and classroom, and thus ideal match of course and classroom for the student i. In general terms we can see the entry $[x]$ as a pair that represent the willingness of the student to take that class the next semester and his favorite classroom to do so.

Definition 10 ***Democratic-Mechanism**: Let $\Delta : D \mapsto R^{|M|*|S|}$ be the matrix that represents the aggregation of preferences.*

In particular for our *democratic* mechanism:

Definition.M 1 ***Relative-rule***

$$\delta_{n,k} := \sum_{i \in I} 1(m_n = m^i[l]) * 1(s_k = s^i[l])$$

Each element $\delta_{n,k}$ represents the number of votes that the course n had for taking place in classroom k. In other words, each element represents the number of students that are willing to take the course n and ranked the classroom k as his favorite option. From now on we will refer to the function Δ as a matrix, since it remains unchanged trough the matching algorithm.

Δ	s_1:	...	$s_{\|S\|}$:
m_1:	$\delta_{1,1}$	...	$\delta_{1,\|S\|}$
...	...	...	...
m_n:	$\delta_{n,1}$	...	$\delta_{n,\|S\|}$
...	...	...	...
m_1:	$\delta_{\|M\|,1}$	...	$\delta_{\|M\|,\|S\|}$

Note that given the construction of the Δ. Each column k of Δ represents the number of votes the classroom k has for hosting the course n. We will extract the preferences of the classrooms from Δ. We will extract the preferences of the classrooms from Δ.

Definition.M 2 ***S-Preferences*** *If* $\delta_{n,k} \geq \delta_{n',k} \iff m_n \succeq^k m_{n'}$

The preferences follow the classic structure of democracy. If more students voted for a particular course to take place in that particular classroom, then it should be preferred over the ones that got fewer votes. In case of a tie the aggregated preferences will be indifferent between the options. These preferences can be represented as an ordered vector that orders the courses in descendant order of votes.

We will denote the preferences for classroom k as the **already ordered** vector λ_k, where every element in $\lambda_k[h]$ is preferred to any element in $\lambda_k[h+1]$.

Theorem 0.1 *Any individual preference $\succeq^k$ induced by the democratic mechanism is rational.*

Proof: Fix a k. Define $\succeq^k$ as before. We will show that the preferences are complete and transitive.

Complete:Let $m_n, m_{'n} \in M$, this implies $\exists\ \delta_{n,k}$, $\delta_{n',k} \in R$. We know that R is a complete order so there are two cases: $\delta_{n,k} \leq \delta_{n',k}$ or $\delta_{n,k} \geq \delta_{n',k}$, then it must follow by definition of *preferences that $m_n \succeq^k m_{n'}$ or $m_n \preceq^k m_{n'}$. Then preferences are complete.*

Transitive:Let $m_a \succeq^k m_b$ and $m_b \succeq^k m_c$. We need to show "$\delta_{a,k} \geq \delta_{c,k}$". By supposition and definition of preferences we know $\delta_{a,k} \geq \delta_{b,k}$ $\delta_{b,k} \geq \delta_{c,k}$ by the definition of the preference k. Then by transitivity of R we know $\delta_{a,k} \geq \delta_{b,k} \geq \delta_{c,k}$, then must follow $\delta_{a,k} \geq \delta_{c,k}$.

As mentioned before transitivity and completeness implies rationality.

The computation of Δ is a direct representation of the normative and theoretical basis for aggregating preferences. If the normative background changes, then this function must change. We discuss exhaustively potential changes and the normative implications to the function in latter sections.

On the one hand we have constructed rational preferences for each classroom, that follow democratic principles to represent the students' preferences. On the other hand, the professors' preferences are assumed to be rational. Then the requirements for any rational matchmaking algorithm are satisfied, and thus all properties previously defined will be inherited up to the aggregated preferences. The former set of preferences represent the utility or well-being of the students, while the latter represent the desires of the professors.

The *adaptive* nature of the algorithm is yet to be described. This feature responds directly to the *overlapping* problem. In general the match-making algorithms solve preference ties at random. We propose a mechanism that weights the potential overlap of courses and decides in favor the course that has the minimum potential overlap. For this mechanism we introduce a final matrix.

Definition 11 *: Let $\Theta : S \mapsto R^{|S|*|S|}$.*

Definition.M 3

$$\theta_{n,n'} := \sum_{i \in I} 1(m_n, m_{n'} \in m^i) \; \forall \; n \neq n'$$

and

$$\theta_{n,n} := 0 \ \forall n$$

This matrix represents the number of potential overlaps of class m_n with any other class $m_{n'}$ if they are scheduled at the same time. Each non-diagonal coefficient of the matrix represents the number of students that are planning to take the same class, so classes with big coefficients should be scheduled in at different times. As before we refer indistinctly to the output matrix of Θ and the function, since it does not change during iteration.

Θ	m_1:	...	$m_{\|M\|}$:
m_1:	$\theta_{1,1}$	...	$\theta_{1,\|M\|}$
...	...	...	...
m_n:	$\theta_{n,1}$	...	$\theta_{n,\|M\|}$
...	...	...	...
m_1:	$\theta_{\|M\|,1}$	...	$\theta_{\|M\|,\|M\|}$

Definition 12 : ***Adaptive tie-breaker***: *Let $M^T = \{m_n \in M \mid m_n$ is already matched with s_k and $t(s_k)$ $T\}$. Suppose M^T is fixed and a tie arised between m_n and $m_{n'}$ for the classroom s, with $t(s) =$ T. Then the classroom s will accept m_n, with out randomizing, if $\sum_{m_l \in M^t} \theta_{l,n} > \sum_{m_l \in M^t} \theta_{l,n'}$. If there are more than two courses involved in the tie, take the one with the biggest sum, if all sums are equal, then decide randomly.*

The rationale behind this criterion is to pick the course that at this round has the minimum overlaps, given the already assigned classrooms.

Theorem 0.2 *The **Adaptive tie-breaker** conserves the properties of a match-making algorithm if the ties are solved at random.*

Proof: There are two possibilities, the first a tie never arises or the tie-breaker is need at round R .

1.- No tie: This is straight forward. Since the adaptive tie-breaker is never used, the output is the same as the unmodified match-making algorithm.

2.- Tie: If the first tie occurs at round r and is decided in favor of m_n, treat the match-making algorithm as the branch where the random solver picked m_n, then the match belongs to one of the potential outcomes of the unmodified match making algorithm.

Both cases were possible in the unmodified match-making algorithm, thus the properties are conserved.

5 Time Complexity

We proceed to analyze the complexity of the algorithm. We need to compute the complexity of each of the matrices created, the construction of P and the match-making algorithm. For this analysis we use the standard big O notation or $O(n)$ where $n = max(|I|, |S|, |M|)$.As mentioned before the match-making algorithm is $O(n^2)$ in its simplest form, and $O(n^4)$ in the most complex formulation.

The complexity of P is $O(n^2)$ since each professor, must input his preferences individually over the set of all possible classrooms (before cloning). If the space of available classrooms is extremely large, the capture algorithm should be able facilitate the input. For example, given assumption **1**, professors can input the (hour,days) tuple in a vector. Note that this can be "parallelized" since all professors at any given time should be able to input their preferences.

Then, it is necessary to calculate the complexity of calculating the Δ. As before once the alumnae has expressed the votes and courses they want to take the only thing left is to calculate $\delta_{n,k}$. There are $|M|*|S|$ coefficients. For calculating each coefficient, the algorithm must iterate through $|I|$ individuals, so the complexity is $O(n^3)$. Then an ordering algorithm is needed to order λ_k while it keeps track of the indexes corresponding to the max elements. Algorithms like merge-sort are capable of doing it in $O(nlog(n))$. Then the complexity of aggregating preferences is $O(n^3)$.

Finally, we need to calculate the complexity of the matrix Θ. The matrix has a dimension of $|S|*|S|$ and each coefficient must sum over $|I|$ individuals. But since the matrix is symmetric we only need to calculate half of it. Then complexity is $O(n^3)$. This suggest that the whole problem has a complexity of $O(n^3)$ if the least complex match-making algorithms are used and $O(n^4)$ if the most complex one is used. The algorithms are *Algorithm 1* and *Algorithm 2* respectively.

6 Stable marriage with indifference

In this version of the stable marriage problem, each person should rank each member of the opposite set in strict order of preference. However, in real-world examples, a person may prefer two or more persons as equally favorable partner. That tied preference is termed as indifference.

If tied preference lists are allowed, then the stable marriage problem will have three notions of stability:

1. *Weakly stable*:= A match is weakly stable unless there is a couple each of whom strictly prefers the other to his/her partner in the matching. The Galey-Shapley algorithm (Algorithm 1)[29] has a complexity of $O(n^2)$ for this weakly approach, where n is the size of stable marriage problem[26] .

2. *Super-stable*:=A matching is stable if there is no couple each of whom either strictly prefers the other to his/her partner or is different between them. The algorithm can reach a complexity of $O(n^2)$[26] .

3. *Strong stable*:- A matching algorithm is strongly stable if there is no couple x, y such that x strictly prefers y to his partner, and y either strictly prefers x to his/her partner or is indifferent between them. The algorithm that finds this *strong stable* matching

Algorithm 2 Gale Shapley Algorithm with indifference

```
Assign each classroom to be free
while (some lecture m is free) do
    begin
    s := first classroom on m's list;
    m proposes, and becomes paired, to s;
   if ( some lecture m' is engaged to s ) then
      assign m' to be free;
   for each (successor m" of m on s's list) do
      delete the pair (m", s)
end;
output the engaged pairs, which form a stable matching =0
```

reaches a complexity in time of $O(n^4)$ (Algorithm 2), and has a *for loop* that is repeated until it finds a stable matching or it does not find a stable matching (it does not exist)[26]
.

Example

Input for the algorithm proposed:

Table 1: demand: D

		am			am
$\delta 1 =$	m1	7	$\delta 4 =$	m1	11
	m2	8		m2	9
	m3	9		m3	10
$\delta 2 =$	m2	9	$\delta 5 =$	m1	12
	m3	8		m2	9
	m4	7		m3	10
$\delta 3 =$	m1	10	$\delta 6 =$	m2	11
	m3	11		m4	10
	m5	12		m5	7

Table 2: lectures

Θ	m1	m2	m3	m4	m5
m1	0	3	3	0	2
m2	3	0	3	2	2
m3	3	3	0	1	1
m4	0	2	1	0	1
m5	2	2	1	1	0

Table 3: deltas

Δ	7 am	8 am	9 am	10 am	11 am	12 am
m1	1	0	0	1	1	1
m2	0	1	3	0	1	1
m3	0	1	1	1	1	0
m4	1	0	0	1	0	0
m5	1	0	0	1	0	1

7 Normative Analysis and Design

There is a possibility of multiple classrooms of the same type (i.e. many classrooms available at the same hour with the same characteristics). If such situation arises then we have to follow the formulation of the problem as described in this section, but before iterating over the matching algorithm we need to create as many duplicates of the classrooms as their availability. Each duplicate or clone will have the same preferences. Then each new classroom will still have rational preferences and thus the assumptions for the match making algorithm are satisfied.

Furthermore, we can relax **Assumption 2**. If a teacher teaches two or more courses at the time. We can model each course as if it had a different professor, with potentially different preferences, and add a function that automatically checks before matching a course with a classroom if the professor has already another course at the same hour already matched, so it rejects the matching. The problem with this approach is that since preferences are not modified at the beginning and rejecting a course, not involved in a tie, may cause the algorithm to diverge or loss its properties. Another potential solution is modeling each course as before, but once the algorithm converges we can look up for all the courses that overlap and look for the next classroom in the professor's ranking and see if some course (not taught by the same agent) is willing to change position, if not look for the next classroom in the rank. This may not conserve the properties of the matchmaking algorithm but will converge, although if no classroom is willing to change, then overlap may no be solved.

Assumptions **1 and M1** may be relaxed. If there are two types of schedules for classes, then two disjoint but feasible set of classrooms may be created. Then the students will input two different types of demand, one for each type of schedule/course. Courses can not belong to both demands. Then each problem should be solved independently and thus each sub-problem will inherit the properties of the match-making algorithm. If a professor is imparting two classes with different schedules, he must state two set of preferences for each possibility set.

The **Democratic-Mechanism** may be modified, but also the **Relative-rule**. The latter has some important implications. For example instead of representing the student's ranking of classrooms for a particular course, we could treat the vector m^i and s^i independently and see the last as a potential schedule. Then the definition would be:

$$\delta_{n,k} := \sum_{i \in I} 1(m_n \in m^i) * 1(s_k \in s^i)$$

This could lead to a more compact schedule, but favors overlap of courses for students.

We could also modify d^i and the **Relative-rule** and establish other preference aggregation method as Borda count. This method is a unique winner election method in which voters order k options according to their preferences. The number of points awarded to an option is equal to the number of candidates ranked lower. The winner is the option with most votes. This system is described as consensus-based, since the option preferred by the majority may not win. This method satisfies monotonicity, consistency, participation, resolvability, plurality, reversal symmetry and Condorcet loser criteria[30].

The s^i will need additional information like the number of points given to each classroom. Then $\delta_{n,k} := \sum_{i \in I} 1(m_n \in m^i) * b_{ik}(s_k \in s^i)$ where b_{ik} is the number of votes assigned to the course m_n so it takes place in s_k. Since Borda count can be represented as a matrix Δ, then **Theorem 0.1** is also satisfied. The former analogy suggest the next theorem:

Theorem 0.3 *: Any form of aggregating preferences that can be represented as a matrix Δ and satisfies the **S-Preferences** also allows a **match-making** representation.*
*The proof can be directly derived from **Theorem 0.1** since the method of aggregating preferences is not involved. Only the construction of the Δ matrix.*

This result allows that any form of voting or aggregating preferences for the alumnae can help solve the problem by applying the matching algorithm. As mentioned before the aggregating preferences problem and voting systems had been widely studied, not only from an empirical or practical form, but from a normative point of view. Thus can provide a robust normative background for the scheduling problem. If the university or agent that faces the classroom-assignation problem has some normative properties in mind, he can search for voting or aggregating preferences systems that satisfies the axioms or normative conditions he finds desirable. The aggregation of preferences or the rule for breaking ties should try to incorporate the multiple objectives the designer has in mind.

Overlap

Assumption **M2** can be relaxed without disrupting the properties inherited from the match-making algorithm, but may allow for potentially more overlaps as mentioned before. Nevertheless, this assumption is created from an *heuristic* point of view and possibly reduces the overlap probability for students.

We think it is acceptable to think that in general students do not care about at which hour a particular class is going to be imparted. We argue that is more realistic to enumerate the preferences over the set of classrooms as a whole rather than assigning pair of (m, s) (class,classroom). For example, an student may just care to take classes at certain hours rather than an specific class at an specific hour. Relaxing this assumption is equivalent to treating function t as one that maps to S^y. Assumption **M2** should change to:

Assumption.M 4 *$m^i[x] \neq m^i[y]$ and $t(s^i[x]) = (s^{i1}_k, ..., s^{iy}_{k'})\ \forall i, x \neq y$.*

. This means that the student just cares about the schedule, and not an specific pair.

Relaxing this assumption posses a great problem. If two classes are going to be taken by a great number of students and an specific classroom (hour) constantly appears on the demand of some students, then it is possible that in the presence of many clones of the same classroom both classes get assigned at the same time. We want to avoid this, but the problem is how to define or detect under which criteria two classes can be considered to overlap. If we manage to establish this criteria, then at any given round we can make a sub-round where we allow one of classes in the overlapping pairs to find an other classroom and repeat the operation until no overlap appears. Note that the overlap criterion ideally should let the algorithm to eventually stop the cascade or succession of sub-routines. This may broke the pareto property. This can be solved by how we aggregate preferences or by how we define the overlap criterion. The directives we decide to take here may completely change the Time Complexity of the algorithm.

Note (not formal): I have been looking for a a computational problem to map this. I have been thinking in minimizing a function that represents Θ (some norm of omega, there are many possible norms). Also some algorithms that deal with graphs and establishing the edges as entries of the Θ function. Trying to map the Δ matrix to a market structure and establishing some prices to solve the equilibrium, but to do this in a countable domain the demand function I stablish need to be submodular. Some bounds can be guaranteed in a greedy structure if the function is subadditive or has certain linearity coefficients. In essence This papers deal with this last approach: https://www.sciencedirect.com/science/article/pii/S0022053196922693 https://www.sciencedirect.com/science/article/abs/pii/S0165489600000718 https://pdfs.semanticscholar.c
https://pdfs.semanticscholar.org/2c8a/ebda5c08db08c423f783c4f92a37571aaac8.pdf

If i dont manage to map this type of problem and obtain some theoretical results, i will find a suitable aggregate function that represents Θ and/or Δ and use heuristic techniques so solve it, probably genetic algorithms.

7 Experiments

** I have not coded the algorithm or ran any simulation** Informal (note): If i don't manage to get some theoretical results, then i will compare many heuristics that try to optimize the function mentioned before.

In order to test empirically the proposed algorithm against other algorithms that deal with the same problem we simulate different scenarios. Two sets agents that have preferences. These two sets represent the students and the teachers. In order to create the preference relation we draw a random classroom (one by one) from their own pool of classrooms which contains the set of all available classrooms without clones. For each individual, if a classroom is drawn in round one the is preferred to all subsequent classroom, and so on. Then, we proceed to modify the random draw of preferences and to a more structured one. The probability of preferring (drawing) a classroom over another depends on their proximity, in time, to previously drawn classrooms. We also proceed to run the simulations, but this time the probability function behaves as a laffer curve. To create the set of available classrooms, we use a full week schedule with different types of classrooms. The classes taken by the students and their respective duration are also random, up to 10 classes. We run all the simulations for different number of classroom clones.

*** IF THE GENETIC ALGORITHM/HEURISTICS IS IMPLEMENTED *** We also simulated different heuristics for solving the overlap problem. *** The metrics will depend on the heuristics we implemented and how i manage to resolve the overlapping problem*** We proceed to analyze the results.

8 Conclusions

Many algorithms, with different approaches, for solving the problem exists. The main difference with our approach is the integration of both agent's preferences, in our case professors and alumnae. Our approach takes ability of the agents to optimize according to their own preferences. Furthermore, it allows professors and students to meet in a simulated environment and express their preferences, while the algorithm optimizes according to the criteria established and the aggregation of preferences. Theoretically the outcome represents the chose of a relative majority, represented by a democratic game. This approach presents an advantage over completely centralized planning, since it really takes into account the actors involved in the process. From a normative point of view the problem should be solved with the welfare of the students and the teachers in mind, since both are the agents interacting in the market.

Another important result is the flexibility of the algorithm, since the aggregation rule can take many forms. The planner can take advantage of the normative literature over this topic and choose the aggregating rule that satisfies the axioms or properties she considers the best. One possible problem of the algorithm is the assumptions, which should be analyzed in detailed. Future lines of work can focus on the relaxation of assumptions and ways to solve them.

The Classroom-Assignment problem is known to be NP-hard. In this paper we propose an algorithm that gives a *pareto* outcome over the preferences in polynomial time $O(n^4)$. Even if the algorithm needed to solve the overlap problem is NP-hard, the pareto criterion is not broken. Furthermore, this pareto optimal criterion rests in the preferences of the students and teachers, and not over a centralized entity or an abstract function

Acknowledgement

Supporting Information Available

References

(1) Carter, C., Michael *Operations Research* **1992**, *40*, S28–S39.

(2) Schaerf, A. *Artificial intelligence review* **1999**, *13*, 87–127.

(3) Broder, S. *Communications of the ACM* **1964**, *2*.

(4) Elloumi, K. H. J. B. . D. A., A. *Applied Soft Computing* **2014**, *14*, 677–686.

(5) Pearl, J. **1984**,

(6) Arrow, K. J. . *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* **1951**,

(7) Arrow, K. J. *WW Norton Company* **1974**,

(8) Sen, A. *Oxford Economic Papers* **1993**, *45*.

(9) Timothy B, S. C. *Journal of Public Economics* **2003**, *87*, 2611–2637.

(10) Yoshikawa, M.; Kaneko, K.; Nomura, Y.; Watanabe, M. A constraint-based approach to high-school timetabling problems: a case study. AAAI. 1994; pp 1111–1116.

(11) Sabar, N. R.; Ayob, M.; Qu, R.; Kendall, G. *Applied Intelligence* **2012**, *37*, 1–11.

(12) Neufeld, G.; Tartar, J. *Communications of the ACM* **1974**, *17*, 450–453.

(13) Colorni, A.; Dorigo, M.; Maniezzo, V. *Politecnico di Milano, Milan, Italy TR* **1992**, 90–060.

(14) Kheiri, A.; Özcan, E.; Parkes, A. J. *Annals of Operations Research* **2016**, *239*, 135–151.

(15) Hertz, A. *Discrete Applied Mathematics* **1992**, *35*, 255–270.

(16) Schaerf, A. *Tabu search techniques for large high-school timetabling problems*; Centrum voor Wiskunde en Informatica, 1996.

(17) Jacobsen, F.; Bortfeldt, A.; Gehring, H. Timetabling at German secondary schools: tabu search versus constraint programming. Proceedings 6th international conference on the practice and theory of automated timetabling, PATAT2006. 2006; pp 439–442.

(18) Bello, G.; Rangel, M.; Boeres, M. *The practice and theory of automated timetabling VII* **2008**,

(19) Abramson, D. *Management science* **1991**, *37*, 98–113.

(20) Abramson, D.; Amoorthy, M. K.; Dang, H. *Asia-Pacific Journal of Operational Research* **1999**, *16*, 1.

(21) Kannan, A.; van den Berg, G.; Kuo, A. *Interfaces* **2012**, *42*, 437–448.

(22) Moura, A. V.; Scaraficci, R. A. *International Journal of Operational Research* **2010**, *7*, 152–170.

(23) Pillay, N. A study into the use of hyper-heuristics to solve the school timetabling problem. Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists. 2010; pp 258–264.

(24) Brown, D. J. *The Quarterly Journal of Economics* **1975**, *89*, 456.

(25) Manlove, D. *Discrete Applied Mathematics* **2002**, *122*, 167–181.

(26) Shapley, G. S., L *American Mathematical Monthly* **1962**, 9–14.

(27) Iwama, S., Kazuo; Miyazaki *International Conference on Informatics Education and Research for Knowledge-Circulating Society* **2008**, 131–136.

(28) Pancs, R. *Springer Press* **2017**, 104–144.

(29) Irving, R. W. *Journal of Algorithms* **1985**, *6*, 577–595.

(30) Lippman, D. *Math in Society* **2015**,