



MARKET DESIGN PROJECT: C.A.S.T.L.E.

Capacity-Aligned Stable Teaching Location Engine
Tackling classroom allocation in Universities

December 2024

Matthieu MINGUET



1

EXECUTIVE SUMMARY

CONTENTS

1	Executive Summary	2
2	Introduction	4
3	Due Diligence	5
3.1	Current System at Ecole Polytechnique	5
3.2	Current system at Clermont School of Business (Clermont SB)	6
3.3	Market Sizing and existing solutions	7
4	Proposed Solution : C.A.S.T.L.E.'s core principles	8
4.1	Matching algorithm implemented	9
4.2	Properties of the algorithm	9
5	Implementation and Testing	12
5.1	Implementation	12

2

INTRODUCTION

The way the rooms are organized leads to inefficiencies, between professors who value and need different things. Some professors need to use boards to write specific, while others do not value such things. Moreover, the impact of the geographical positioning of the rooms relative to both the students and the teachers is something to take into consideration, or the proximity of the coffee machine, even the amount of stairs required to reach the room. As such it is my belief that a new system designed around the principles of market design could lead to a more efficient allocation of the school's resources, as well as collect valuable information term on term on what professors value and what type of classrooms they preconize.

3

DUE DILIGENCE

3.1 CURRENT SYSTEM AT ECOLE POLYTECHNIQUE







X currently operates a centralized room allocation system managed by the Registrar's office through a shared spreadsheet. While the basic objective of assigning rooms to all classes is met, the system faces significant operational challenges. The manual allocation process is time-consuming, requires frequent bargaining between departments, and often fails to meet teacher preferences and specific classroom needs. Moreover, it does not provide a reporting of what the

The system's current stability relies heavily on the natural distribution of class sizes, particularly in the *Cycle Ingénieur* program where enrollment decreases from 550 students in first year to increasingly smaller specialized groups through second and third year. This pattern allows for predictable allocations, such as Poincare for large lectures and smaller rooms for tutorials, especially when third-year students are off-campus. With the inversion of tutorials and lectures times based on the year, the system can support the needs of the program.

However, mounting pressure from expanding masters and bachelor programs is straining this delicate balance. The competition for rooms, especially between second and third-year needs, is intensifying. While the system functions now due to fortunate enrollment patterns, it lacks the robustness to handle future growth and increasing complexity.

3.1.1 • FLAWS IN THE CURRENT SYSTEM

In terms of market design, this system is a centralized dictatorship, as in a control economy where resources are allocated based on the sole criteria of class size. It's flaws which we will detail in the following table:

Flaw	Description	Intensity
Information Problem	<ul style="list-style-type: none"> No preference elicitation Manual spreadsheet means no way to communicate preferences, constraints, or trade 	▼ 
Matching Mechanism	<ul style="list-style-type: none"> Single-Criterion Optimization: capacity matching Manual Coordination Cost: bargaining No formal Trading Mechanism: no beneficial trades 	▼ 
Incentive Problems	<ul style="list-style-type: none"> No strategic reporting: no formal way to express the preferences for rooms or features Lack of Price Mechanism: no way of quantifying preferences 	▼ 
Scalability	<ul style="list-style-type: none"> Brittle equilibrium: program change or growth could pose a threat to the current system Manual processing bottleneck 	▼ 
Efficiency problem	<ul style="list-style-type: none"> No Optimization Algorithm Suboptimal Outcomes: room for Pareto improvements Resource Utilization: inefficient use of features and capabilities 	▼ 
Adaptation and flexibility	<ul style="list-style-type: none"> Rigid system: changes mid-term can be difficult Limited feedback loop: for allocation and features 	▼ 

As we can see, while the current system has major flaws like the lack of a matching mechanism, it still ensures all teachers have a room, and the school has functioned with this system for the past few years without




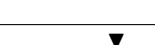


any issues, especially when the *Cycle Ingénieur* program was the only program at the school.

3.2 CURRENT SYSTEM AT CLERMONT SCHOOL OF BUSINESS (CLERMONT SB)

The Clermont SB has a similar system to Ecole Polytechnique, with a centralized room allocation and planning system. A dedicated member of staff is responsible for managing the system in place: Program Directors submit room requirement and planning forms to the Planning Office, which then manually attributes the rooms based on the information provided, and the availabilities of the rooms. These types of requirements include the number of rooms, the equipment, expected cohort size, and if the rooms need to be close to each other or not (for example for group work). The number of variables makes it very difficult to find a solution satisfying all requirements of all courses. While the system allows for inputs from the faculty, the major issue is that the large number of programs and the increasing number of students in each program is making the system increasingly complex to manage by hand and is leading to inefficiencies and tensions in the allocation. Moreover, this system is quite slow and requires a large amount of time to manage, which could be streamlined and better used elsewhere. Finally, the system is not transparent, and the faculty does not have a clear understanding of how the rooms are allocated, while it also doesn't encourage truthful reporting of preferences.

3.2.1 • FLAWS IN THE CURRENT SYSTEM

Similarly as what we did for Ecole Polytechnique, we have identified the following flaws in the current system at Clermont SB:

Flaw	Description	Intensity
Information Problem	<ul style="list-style-type: none"> • Preference reporting through requirement forms • Lack of transparency in allocation process 	
Matching Mechanism	<ul style="list-style-type: none"> • Manual attribution based on submitted forms • Single-person decision making process • No formal mechanism for resolving conflicts 	
Incentive Problems	<ul style="list-style-type: none"> • No incentive for truthful preference reporting • Lack of feedback mechanism 	
Scalability	<ul style="list-style-type: none"> • Growing complexity with increasing programs • Rising student numbers strain current system • Single point of failure (dedicated staff member) 	
Efficiency problem	<ul style="list-style-type: none"> • Time-intensive manual process • Inefficiencies in allocation • Resource under-utilization due to information gaps 	
Adaptation and flexibility	<ul style="list-style-type: none"> • System struggles with program diversity • Difficult to make mid-term adjustments • Slow response to changing needs 	

As we can see, while this system offers more inputs from the different stakeholders, it is limited by the fact that it has to be manually run, and an increasing number of programs and students compared to the number of rooms, or even more rooms could cause issues regarding the length of the process and the fairness of the allocation.

3.3 MARKET SIZING AND EXISTING SOLUTIONS

3.3.1 • MARKET SIZING

Their are around 18500 universities worldwide, representing around 250m students. However, a new system such as C.A.S.T.L.E. would be more suited to larger universities, especially in developed countries, where the number of students is higher, and the number of programs is more diverse. As such, we estimate a theoretical market size of 5000 universities, mainly in Europe and North America. At first, this system would target around 80 universities in these markets, which considering their size could already impact upwards of 200000 students. Thus, C.A.S.T.L.E. could bring an improvement over the current systems in place in these universities, and potentially be a game-changer in the way universities allocate their resources.

3.3.2 • EXISTING PAPERS

Classroom allocation or time-tabling is a well-studied problem in the literature, with many papers proposing different solutions to the problem. For example, Rivero, Escárcega, and Velasco (2021) proposed an Integer Linear Programming model for classroom assignment to tackle this issue, while other solutions focus on graph coloring algorithms or genetic algorithms to allocate classrooms and timetables. In almost all cases, the solution to solve both timetabling and classroom allocation is to decouple the two problems and solve them sequentially, as the classroom allocation algorithm generally output near 100% utilization of the rooms, which we will see is also the case for C.A.S.T.L.E.. The basic formulation of the problem for classroom allocation is as follows:

- Let x_{ij} be a binary variable where i the classroom, j the course and : $x_{ij} = 1$ if course j is assigned to classroom i , 0 otherwise.
- The objective function is : $\min \sum_{i,j} c_{ij}x_{ij}$ where c_{ij} is the cost of assigning course j to classroom i .
- c_{ij} represents the cost of assigning class j to classroom i , and can incorporate functions like room size mismatch, distance from the professor's department, equipment mismatch, etc.

The constraints are generally the following:

- Single assignment constraint: $\sum_i x_{ij} = 1$ for all j
- Room capacity constraint: $\text{enrollment}_j \leq \text{capacity}_i \forall x_{ij} = 1$

In cases where timetables are also considered, the time t is added as a third dimension to the problem, and the constraints are modified to include the time dimension (professor, room and student availabilities) This formulation guarantees an optimal solution (if solvable) and is able to balance different objectives by weighting the c_{ij} , while being very flexible in terms of constraints and objectives.

However it is an NP-hard problem to solve and the solution space grows exponentially with the number of rooms, classes (and time slots). As an example, the Ecole Polytechnique has 76 rooms, around 65 to 70 classes at one time, for a complexity of $76^{70} \approx 10^{140}$, which is infeasible to solve with a brute force algorithm, and we have not even considered the time dimension. Moreover, the model is static and can't easily handle dynamic changes.

Work exists to improve upon this by for example relaxing the strict constraints and adding a violation penalty, doing pre-processing to reduce the search space, or using heuristics to find a good solution in a reasonable time. Other systems such as graph coloring exist, but they are also NP-hard, like the Integer Linear Programming model, and are generally less flexible as they have difficulty incorporating different objectives and soft constraints.

In conclusion, while significant academic work has been done on the subject of classroom allocation and timetabling, the problem remains complex and difficult to solve, especially in a dynamic environment like a university. Findings from these papers are useful to understand the problem and the constraints, but a new

approach is needed to tackle the problem in a more efficient way. We do keep in mind the idea of decoupling the timetabling from the allocation, as C.A.S.T.L.E. only focuses on the allocation part of the problem.

4

PROPOSED SOLUTION : C.A.S.T.L.E.'S CORE PRINCIPLES

The key objectives of C.A.S.T.L.E. in light of the current systems and the existing literature are the following:

- **Efficiency:** Maximizes room utilization and stakeholder satisfaction through optimal allocation of classrooms to courses.
- **Computability:** Delivers solutions within reasonable time frames on standard hardware, even at university scale.
- **Flexibility:** Adapts to diverse objectives and constraints while accommodating institution-specific requirements.
- **Transparency:** Ensures allocation decisions are understandable and promotes honest preference reporting.
- **Reliability:** Minimizes manual interventions by consistently producing near-complete room assignments across complex scenarios.

Moreover, we should also try to limit the time spent by faculty on this process, as it takes away from their time. C.A.S.T.L.E. should also not be based on any currency as this could cause problems with faculty and require more thought and time to use the system. Based on the objectives of fitting the cohorts in the rooms and allowing the maximum amount of stakeholders such as faculty, students, and outside speakers to benefit from the best and most sought after rooms, a cardinal parameter is to consider the "fill amount" of the rooms, or in other terms, to minimize $\text{room}_{\text{capacity}} - \text{course}_{\text{cohortsize}}$.

To achieve these objectives, two main algorithms were initially thought of:

YOU REQUEST MY ROOM, TAKE IT IF YOU'RE BIGGER (YRMRTIYB)

This is based on a similar idea as *You request my house I get your turn* which is already based on the Top Trading Cycles (TTC) algorithm. Each course would point towards the room they want, and the rooms towards their top choice of course. Then, the algorithm would find cycles of courses and rooms, and allocate the rooms to the courses in the cycle. courses then point towards their next choices, and the algorithm continues until someone requests a room that is already taken, at which point if their cohort size is bigger, they get the turn of the first course. Then the algorithm runs once more, and the cycle continues until all rooms are allocated.

The main problem of this algorithm is that stability is not guaranteed, which is a prerequisite for a long term allocation process. It does however bring strategy-proofness which is desirable to avoid strategic reporting.

DEFERRED ACCEPTANCE ALGORITHM (DA)

In this case, faculty members (or inversely rooms) would propose to a room (or inversely a faculty member) based on their preferences following the classic deferred acceptance algorithm. This approach would guarantee stability, albeit without being inherently strategy-proof. Additionally, it is a well-know algorithm that has been implemented in many different contexts. The final version of C.A.S.T.L.E.'s algorithm is similar to that of "the Match" for medical students as it is a one-to-many matching.

4.1 MATCHING ALGORITHM IMPLEMENTED

Room of identical types, such as rooms in the same corridors having the same equipment are grouped into a single type that represents them, order to minimize the number of choices having to be made by faculty members. As such their **preferences are for room types**. For the core algorithm we consider the fit function f . Finally, before running the algorithm, we consider the four following parameters known: room_{type} , $\text{room}_{capacity}$, $\text{course}_{cohortsize}$, and $\text{course}_{preferences}$.

1. Courses propose to their top choice of room type based on their preferences. We then run the steps 2 to 4 for each room type separately.
2. All proposals to a type are grouped, including any courses already tentatively allocated in the rooms.
3. In order of increasing capacity, rooms tentatively choose the course corresponding to $\min_{course} f(course, room)$, and it is removed from the list of proposals.
4. Once all rooms in the type have chosen the remaining courses are sent back to the global proposal pool, or their are no remaining proposals in this type.
5. The remaining courses change their current choice of room type to the next one and go back to step 1.
6. The algorithm stops when all courses have been allocated a room, or the remaining courses have tried all their choices.

4.2 PROPERTIES OF THE ALGORITHM

The proposed algorithm has multiple desirable properties for our market design problem.

4.2.1 • DEFINITIONS AND NOTATION

Let us define:

- $C = \{c_1, \dots, c_n\}$: Set of courses
- $R = \{r_1, \dots, r_m\}$: Set of rooms
- $T = \{t_1, \dots, t_k\}$: Set of room types
- P : the maximum number of preferences per course ($P \leq T$)
- \succ_c : Preference relation for course c over room types
- μ : A matching function $C \rightarrow R \cup \{\emptyset\}$
- $s(c)$: Size of course c
- $cap(r)$: Capacity of room r
- $type(r)$: Type of room r

4.2.2 • COMPUTABILITY

Let us define the following variables: C the number of courses to be allocated, R the number of rooms, T : the room types, P the maximum number of preferences per course ($P \leq T$)

The allocation process can be visualized as the following flowchart:

1. For each unmatched course ($O(C)$):
 - Group proposals by type: $O(C)$
 - For each room type ($O(T)$):
 - Sort rooms by capacity: $O(R \log R)$
 - For each room, evaluate all available courses: $O(R \cdot C)$

The algorithm has a worst-case time complexity of: $O(C \cdot T \cdot (R \log R + R \cdot C))$ and a space complexity of $O(R + C)$.

However, in practical applications, several factors contribute to better average-case performance:

1. Most courses are matched in early iterations, reducing the effective number of rounds from $O(C)$ to approximately $O(P)$
2. Room types T is a small constant (approximately 10 in our implementation), and $P \leq T$.
3. With a relatively small number of rooms ($R < 1000$), the room sorting component becomes nearly linear: $O(R \log R) \approx O(R)$

As such, the results are an average-case time complexity of simply $O(C \cdot R)$, and space complexity $O(R + C)$. This is a significant improvement and enables the algorithm to run in a reasonable time frame on standard hardware, even at university scale. Empirical evidence from our implementation supports this analysis. With a typical scenario of 70 courses and 70 rooms and spread out preferences, the algorithm completes in 150-300 allocation steps, with most courses being matched within their first three preferences.

4.2.3 • STABILITY ANALYSIS

Theorem 1 (Stability). *C.A.S.T.L.E. allocation mechanism produces a stable matching.*

Proof. Suppose, for contradiction, that the matching μ is unstable. Then there exists a blocking pair (c, r) where: c prefers r to its current assignment $\mu(c)$ and r prefers c to its current occupant $\mu^{-1}(r)$. However, by construction of the algorithm:

1. If $type(r) \succ_c type(\mu(c))$, then c would have proposed to rooms of type $type(r)$ before $type(\mu(c))$
2. Within each type, rooms are assigned to best-fitting courses based on capacity utilization
3. If $s(c) \leq cap(r)$ and $capafit(r, c) < capafit(r, \mu^{-1}(r))$, then r would have been assigned to c in the per-type processing phase

Thus contradicting the existence of a blocking pair, proving stability. □

4.2.4 • PARETO EFFICIENCY

Theorem 2 (Pareto Efficiency). *C.A.S.T.L.E. allocation produces a Pareto efficient matching among stable matchings within each type.*

Proof. Consider a matching μ produced by the algorithm. Suppose, for contradiction, that there exists another stable matching μ' that Pareto dominates μ . This means:

1. $\forall c \in C: type(\mu'(c)) \succeq_c type(\mu(c))$

2. $\exists c^* \in C: \text{type}(\mu'(c^*)) \succ_{c^*} \text{type}(\mu(c^*))$

For types processed in round k , let C_k be the set of courses assigned in that round. The algorithm maximizes:

$$\sum_{c \in C_k} \text{capafit}(c, \mu(c))$$

Therefore, any improvement for c^* would necessarily worsen the fit for some other course in the same round, contradicting Pareto dominance. \square

4.2.5 • STRATEGY-PROOFNESS ANALYSIS

While the classic deferred acceptance algorithm is not generally strategy-proof, C.A.S.T.L.E.'s variant has interesting properties that make it difficult to manipulate.

Theorem 3 (Limited Manipulability). *Capacity constraints, Multi-dimensional preferences, Information asymmetry limit manipulability of the mechanism.*

Proof. Consider a course c attempting to manipulate its preference list \succ_c . For successful manipulation:

1. c must know:
 - Preferences of other courses: $\{\succ_{c'}\}_{c' \in C \setminus \{c\}}$
 - Sizes of other courses: $\{s(c')\}_{c' \in C \setminus \{c\}}$
 - Room capacities: $\{\text{cap}(r)\}_{r \in R}$ which he does know as it is public information
2. c must find a preference list \succ'_c such that:

$$\text{type}(\mu'(c)) \succ_c \text{type}(\mu(c))$$

where μ' is the matching under manipulated preferences

\square

Practical Implications In conclusion, theoretical properties have important practical implications:

1. Stability ensures no course-room pair has incentive to deviate and should allow for a long-term deployment
2. Pareto efficiency guarantees no wasteful assignments within types, maximizing room utilization and stakeholder satisfaction
3. Limited manipulability reduces gaming concerns in practical deployments, promoting honest preference reporting and makes it easier to understand the preferences of the faculty
4. Thus, the mechanism balances competing objectives:
 - Course preferences over room types
 - Efficient room capacity utilization
 - Fair allocation among courses

Hence, the proposed mechanism is well-suited to the challenges posed by university classroom allocation, offering a robust, efficient, easy to use solution, that offers good properties for the stakeholders. It manages to achieve computability by restricting the tested combinations to those asked by the faculty in the Gale-Shapley algorithm, which also guarantees stability and Pareto efficiency. Finally, it is not strategy-proof, but the limited manipulability encourages honest reporting of preferences and reduces the risk of gaming the system.

5

IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION

The algorithm was implemented in Java, leveraging object-oriented programming principles to model the problem and solution effectively. Moreover, it allows for clear organization of the code and easy extension and modification of the algorithm with different parameters, fit functions, rooms or constraints through the use of abstract classes and interfaces. This creates a modular and flexible environment for coding and testing the algorithm. Most importantly, the execution speed of java versus python is nearly 100 times faster, which eases the use of the algorithm on a large scale and quick changes. This is achieved without the need for complex parallelization or optimization, as the algorithm is already efficient enough to run on a standard laptop. On a scenario of the size of Ecole Polytechnique, the algorithm runs in around 10 milliseconds, which is more than acceptable for a near real-time allocation process. The code structure and key elements are detailed in the annex. The main classes are the allocation algorithm, all the objects representing the courses, rooms, and room types, and the fit function.