Universiteit Antwerpen

Artificial Intelligence for optimising the timetables of university courses and exams

De Schepper Cedric

Principal Advisor: Serge Demeyer
Assistant Advisor: Joey De Pauw

January 4, 2023

AnSyMo
Antwerp Systems & Software Modelling
University of Antwerp

Contents

List of Figures

List of Tables

Abstract

English abstract.

Nederlandstalige Samenvatting

Dutch Abstract.

Acknowledgments

Student's personal acknowledgements.

1 Introduction

We also included a comments command which makes it easier for students and advisors to give feedback or ask questions directly on the text. We can hide all comments in the pdf by changing a single line in the main tex file.

2 Background

2.1 Tabu Search

Tabu Search is a metaheuristic search algorithm based on local search, a heuristic optimisation method that traverses the possible search space by performing local changes to the current solution. Since local search will only accept improving solutions, this traversal can end up being stuck in a local optimum. Tabu search is different from local search by relaxing this rule and also accepting worsening solutions. Additionally, tabu search maintains a memory structure to avoid changes being reversed. The usage of short- to long-term memory is based on the assumption that optimisation techniques must incorporate memory to qualify as intelligent and that a bad strategic choice is superior to a good random choice [1]. This memory structure is implemented by maintaining a tabu list which contains the x most recent changes performed. A change is thus considered 'tabu' if it is present within the tabu list.

Tabu search starts by constructing an initial solution. The solution can be generated randomly or by applying a deterministic approach. During the entire process, the best seen solution to date is maintained. This is necessary due to tabu search allowing worsening changes or so called moves to avoid getting trapped in local minima. After the solution initialisation, the iterative procedure starts searching for a feasible solution. This loop ends when a specified stopping condition is met. This condition is generally a combination of a solution its objective function scoring below a certain threshold or the procedure reaching a maximum number of iterations.

The first step of the main loop is to generate the complete list of possible neighbours of the current solution and ranking them based on the objective function. Subsequently, the best neighbour that is either not tabu or that meets the aspiration criterion is chosen as the next solution. A neighbour. A neighbour is considered tabu if it is present within the tabu list. The aspiration criterion is added to be able to override the tabu requirement. A possible aspiration is to accept solutions that are better than the best seen solution. After choosing the next solution, the best seen solution is updated if the newly generated solution is superior based on the objective function.

When the stopping criteria is met, the algorithm ends and the best solution is returned. The full flow of the algorithm can be seen in Fig.1.

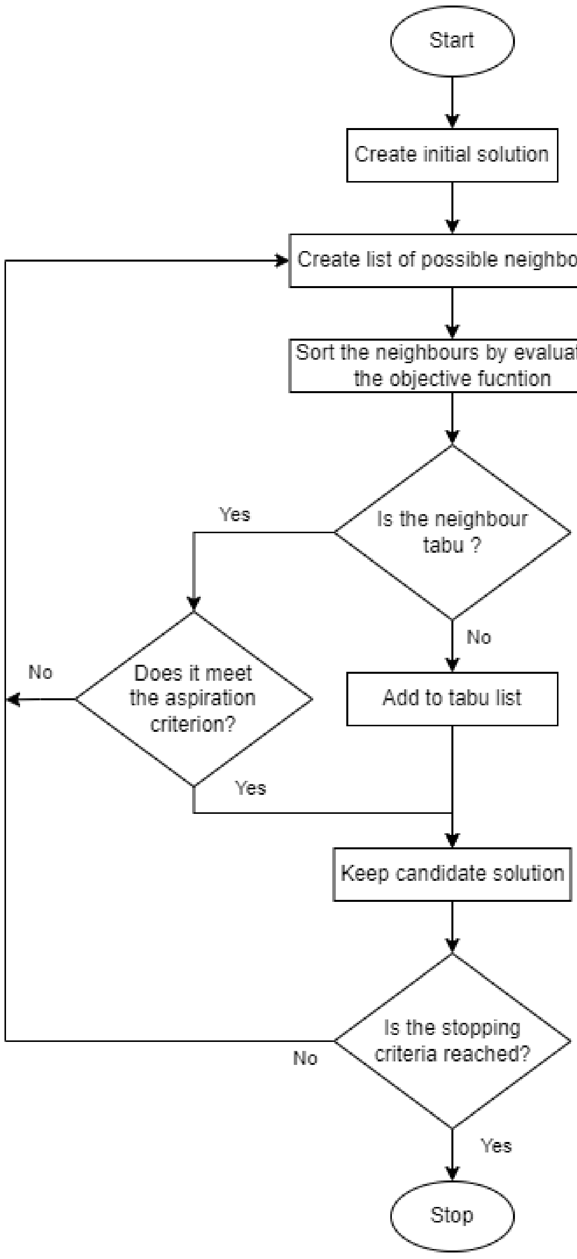

Start
Create initial solution
Create list of possible neighbours
Sort the neighbours by evaluating the objective fucntion
Is the neighbour tabu ?
Yes
No
Does it meet the aspiration criterion?
No
Yes
Add to tabu list
Keep candidate solution
Is the stopping criteria reached?
No
Yes
Stop

Figure 1: Tabu search flow

3 Method / Experimental Design

3.1 Problem definition

The task of creating a university exam timetable can be reduced to a scheduling problem. The goal is to assign all exams to available time slots in order to produce a schedule without conflicts. Additionally, the distribution of exams has to be optimised as to provide a student with the highest chance of success. These requirements generally are defined as hard and soft constraint. Hard constraints are requirements that have to be met in order to considered a feasible solution, while soft constraints are preferred to be violated as little as possible. Not all timetabling problems might have feasible solutions depending on the data set used. After consulting with the responsible parties for the scheduling of the exams for the FTI and FWET faculties, the following constraints were determined:

- Constraint 1: Each exam is scheduled to a timeslot.

- Constraint 2: Each exam is scheduled to a room of the correct type.
- Constraint 3: The number of students enrolled in an exam cannot exceed the capacity of the scheduled room.
- Constraint 4: No student must be scheduled to more than 1 exam on the same day.
- Constraint 5: The number of days between exams must be maximised

Constraint 1 to 4 are considered to be hard constraints with contraint 5 being the sole soft constraint. Formally this means that the first 4 constraints must be satisfied in order to provide a feasible solution, with constraint 5 being taken into account to improve it.

In order to solve this scheduling problem, some data must be provided or generated:

Students The set of all students that are enrolled in exams and will impact the timetabling.

Exams The set of all exams that must be scheduled. Every exam has its own set of students that are enrolled which will allow to verify if two exams have common students and would thus conflict if they occur on the same day. Additionally, an exam belongs to a faculty and is of a certain type. Two examples of exam types are oral and written exams.

Rooms The set of rooms present for examinations. Each room has a room type, a student capacity and list of faculties it is available for. An exam can only be scheduled in a room if the capacity is sufficient, the type of exam is possible within the room, and the room is available to the faculty to which the exam belongs.

Time slots The set of all time slots that can be used to schedule exams. A time slot is determined by its date, exam time, and room. The exam time determines the amount of times a room can be used per day.

Periods Set of periods, one for each date within the examination schedule. Each period contains all time slots of that date. By checking all students that are enrolled within exams scheduled to a time slot of a period, the amount of constraint 4 ("no student must be scheduled to more than 1 exam on the same day") violations can be determined. These periods can be generated by adding a time slot for every room and time combination. The amount of time slots in a period corresponds to $|\text{Rooms}| * |\text{Exam Times}|$.

3.2 Implementation frameworks

The algorithm implemented to run the experiments is written using Python3.10. The code requires no packages that are not in the Python Standard Library. The profiler cProfiler is used to analyse the performance. It generates a list detailing how long each part of the algorithm runs for and allows to pinpoint and improve bottlenecks.

3.3 Implementation: version 1

For this thesis, the initial version or version 1 of the tabu search algorithm to be used in the experiments is based on the algorithm proposed by Alvarez-Valdes et al [2]. The algorithm described makes use of a 2-phased approach. During phase 1, the emphasis is on the hard constraints. More specifically, reducing the amount of common students conflicts on the same day is prioritised. Optimally, this initialisation phase would return a solution that is already feasible. The second phase can be considered an optimisation phase, which tries to satisfy the soft constraints as much as possible and thus attempt to distribute the exams evenly.

3.3.1 Initialisation phase

The algorithm starts with the initialisation for which the pseudo code is shown in algorithm 1. Firstly, the initial solution must be generated. In the proposed algorithm this is done by randomly assigning all exams to time slots. Generally this will result in several hard constraint violations such as students have more than 1 exam on the same day and room capacity being exceeded. Since Alvarez-Valdes et al consider the room capacity a soft constraint, this is acceptable and will be improved during the iterations. However, in this case, room capacity is seen as a hard constraint that can't be violated at all. In order to circumvent these capacity violations, exams are sorted by student count and only then randomly assigned to time slots with rooms having sufficient capacity. As long as the time slot quantity and room capacity is sufficient, this will result in a solution with no room capacity violations. Another added complexity is the presence of different room types which was not the case for Alvarez-Valdes et al. In order to satisfy constraint 2, only rooms with a suitable type are considered when assigning exams to time slots.

After generating the initial solution, the iterative procedure starts by generating the set of neighbours of the current solutions. We consider a solution $s' \in X$ a neighbour of $s \in X$, whenever we can move an exam to a time slot in a different period. Evaluating the entire search space would be too time consuming. Instead, we first sort all periods by

Algorithm 1: Initialisation phase

```
1  Generate an initial solution s in the space of solutions X;
2  $s^* = s$ (with $s^*$ the best solution seen so far);
3  $k = 0$ (with k the number of iterations);
4  while k < maximum number of iterations and $f(s^*) \neq 0$ do
5      $k = k + 1$;
       /* perform single moves of one exam to another time slot */
6      Generate set of solutions $V^* \subseteq N(s, k)$ (set of neighbours of s);
7      Sort $V^*$ by ascending $f(s')$ and select the best $s'$;
8      foreach solution $s'$ in $V^*$ do
9          if not $tabu(s')$ or $f(s') < f(s^*)$ (aspiration criterion) then
10             $s = s'$;
11             break;
12         end
13     end
14     if $f(s') < f(s^*)$ then
15         $s^* = s'$;
16     end
17 end
18 return $s^*$
```

its contribution to the objective function. From the most conflictive period, we select the most conflictive time slot. Finally, we calculate all available time slots that we can swap with. In order to swap two time slots, the two affected exams (or sole affected exam when swapping to a time slot with no scheduled exam) must be able to be scheduled in the new time slots keeping the room type and capacity into account. This will ensure that no additional constraint 2 and 3 violations are introduced.

For each possible move, the objective score is calculated in order to rank all neighbours. The objective function for a solution s can be defined as

$$f(s) = \sum_{i=1}^{N}(\sum_{j,k \in E_i} px_{jk}) \tag{1}$$

with N the amount of periods, E_i the set of exams scheduled to period i, x_{jk} the number of common students between exams j and k, and p the weight of conflicts.

After sorting all found neighbours by objective function, the best solution, that is not tabu or for which the aspiration criterion applies, is chosen. The aspiration criterion accepts solutions that are tabu as long as they are superior to s^* (the best solution seen so far) If no moves are possible, we select the next most conflictive time slot until a suitable move has been found. When choosing the new solution, the tabu list is updated with the new tabu move consisting of the exam and period involved. Whenever the tabu list exceeds its maximum size, the oldest move is deleted, in order to allow that move again in future iterations.

This phase runs until a solution without conflicts has been found or the maximum number of iterations has been reached. For the latter case, the schedule will have conflicting exams during the same period with common students.

3.3.2 Optimisation phase

After the initialisation phase, we start the optimisation of the solution. Here the focus is including the soft constraints in the objective function to generate a solution that is the most optimal. The foundation of this phase is similar compared to the first phase with some distinct features. The overall flow can be seen in the pseudo code described in algorithm 2.

Firstly, the objective function is updated to take the distribution of exams into account. The objective of a solution is now defined as

$$f(s) = \sum_{i=1}^{N}\sum_{j=1}^{N}(p_{|i-j|}\sum_{k \in E_i}\sum_{l \in E_j} x_{kl}) \tag{2}$$

Algorithm 2: Optimisation phase

```
Data: Solution s (the result of the initialisation phase)
1  $s^* = s$ (with $s^*$ the best solution seen so far);
2  $k = 0$ (with k the number of iterations);
3  while $k$ < maximum number of iterations and $f(s^*) \neq 0$ do
4      $k = k + 1$;
5      if $k \bmod 3 \neq 0$ then
           /* perform single moves of one exam to another time slot   */
6          Generate set of solutions $V^* \subseteq N(s, k)$ (set of neighbours of s after single move);
7          Sort $V^*$ by ascending $f(s')$ and select the best $s'$;
8          foreach solution $s'$ in $V^*$ do
9              if not $tabu(s')$ or $f(s') < f(s^*)$ (aspiration criterion) then
10                 $s = s'$;
11                 break;
12             end
13         end
14     else
           /* swap entire period   */
15         Generate set of solutions $V^* \subseteq N(s, k)$ (set of neighbours of s after swapping periods);
16         Sort $V^*$ by ascending $f(s')$ and select the best $s'$;
17         $s = s'$;
18     end
19     if $f(s') < f(s^*)$ then
20         $s^* = s'$;
21     end
22 end
23 return $s^*$
```

with N the amount of periods, E_i the set of exams scheduled to period i, x_{kl} the number of common students between exams k and l, and $p_{|i-j|}$ the penalty for two exams scheduled at a distance of $|i - j|$ periods or days. While the objective function in the first phase only used the common students between exams during the same period, the new objective function will take all combination into account. By having a large penalty p_0 for exams scheduled on the same day, the hard constraint of having no exams on the same day for a student is still being prioritised. Additionally, placing exams at a further distance will now be preferred since the penalty p_{x+1} will be smaller than the penalty p_x.

Secondly, two options are now available when generating the set of neighbours of the solution. The first option is a copy of the process in the initialisation phase where the set of neighbours of the solution is generated by performing single moves. Then the best solution that is either not tabu or that meets the aspiration criterion will be used. While this option might introduce constraint violations, the high p_0 will discourage and depending on the values used prevent this. This method will be alternated with a period permutation. Instead of moving a single exam, two entire periods will be swapped. Since the exams in the periods are not updated, this swap can not introduce any hard constraint violations. By moving entire periods, the objective function can change significantly in a single move. However, the amount of available moves is much lower and can be defined as

$$\begin{aligned}\text{\# possible moves} &= \binom{N}{2} \\ &= \frac{N!}{2!(N-2)!}\end{aligned} \tag{3}$$

3.4 Implementation: version 2

After running experiments using version 1, some run time and result characteristics were observed. Version 2 diverts even more from the original algorithm in order to avoid these occurrences. Firstly, the objective function originally defined in the initialisation phase as

$$f(s) = \sum_{i=1}^{N} \Big(\sum_{j,k \in E_i} p x_{jk} \Big) \tag{4}$$

has an unwanted characteristic: it does not punish students with several exams on the same day extra. For exam schedules without a conflict free solution, this often results in students, who are enrolled in smaller exams, having a large number of exams on the same day. In order to avoid this behaviour, the objective function is updated to

$$f(s) = \sum_{i=1}^{N} (\sum_{s \in S(E_i)} p^{c_{si}-1}) \tag{5}$$

with N the amount of periods, E_i the set of exams scheduled to period i, $S(E_i)$ the set of students with more than 1 exam during E_i, c_{si} the number of exams student s has on period i, and p the weight of conflicts. By placing c_{si} in the exponent, having multiple exams on the same day is punished severely.

Secondly, the way a tabu move is defined was changed. Originally, when moving an exam to a different period, the combination of the exam and period was used to define the tabu move. This resulted often resulted in the scenario that the same exam was continuously being moved. since this generally was one of the largest exams, it always has a large impact on the objective function and kept being determined as the most conflictive time slot. This behaviour can be avoided by having the exam as the sole identifier of the tabu move. By doing this, a single exam will not be moved as frequently.

3.5 Implementation: version 3

Version 1 and 2 both reduce the search space by only looking at the most conflictive time slot of the most conflictive period in order to determine the possible moves. Different time slots are only looked at when no moves are possible. This version steps away from that principle and continues evaluating the next most conflictive time slots their possible moves until a specified amount of possible moves have been found. Only then it selects the move generating the best next solution.

3.6 Experiment

For this case study, the data set to be used is provided by the FTI and FWET faculties of the University of Antwerp. The data consists of the exam information for the January and June exam period of 2021 and the schedule used. Even though the two faculties compose their own schedules, the problems are not independent. A section of the rooms are available for both faculties. As a consequence, the schedule has to be generated together and only then be split up. Otherwise a shared exam room could be double booked. The statistics for the data sets provided can be seen in table 1 and 2. The fact that the scheduling problem for FTI and FWET is not independent can be seen here as well. The statistics for FTI and FWET combined do not equal the sum for the statistics of the faculties separated. The amount of time slots can be calculated as follows:

$$\text{\# of time slots} = \text{\# Rooms} \times \text{\# Periods} \times \text{\# Exam times} \tag{6}$$

with there being two exam times, namely a morning and afternoon exam. For the January and June schedule there are respectively 20 and 24 periods.

Table 1: Data set statistics for January 2021

	FTI	**FWET**	**FTI+FWET**
Students	770	1041	1811
Exams	185	360	536
Rooms	15	45	54
Time slots	600	1800	2160

Table 2: Data set statistics for June 2021

	FTI	**FWET**	**FTI+FWET**
Students	782	1071	1852
Exams	169	342	491
Rooms	15	45	54
Time slots	720	2160	2592

Tabu search has a limited amount of hyperparameters compared to other optimisation methods. This aids in the parameter tuning process. The hyperparameters available for the different versions can be seen in table 3.With

MAX_ITER_OPTIMISATION and MAX_ITER_INITIALISATION defining the duration of the algorithm its run time, the main variables to set are the weights. Unlike P_INITIALISATION being a single value, P_OPTIMISATION is a list with the value at index i containing the penalty for two exams at distance i. If there is no value at index i, the penalty is equal to 0.

Table 3: Tabu serach hyperparameters

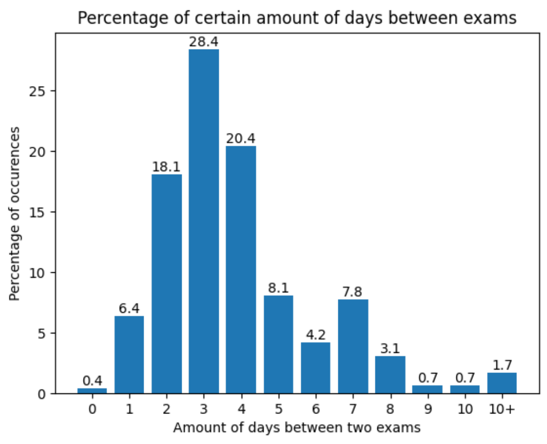
	Parameter	Description
All versions		
	P_INITIALISATION	weight of conflicts during initialisation phase
	P_OPTIMISATION	weight of distance between exams
	MAX_ITER_INITIALISATION	max amount of iterations during initialisation phase
	MAX_ITER_OPTIMISATION	max amount of iterations during optimisation phase
Version 3		
	MAX_MOVES	amount of moves to evaluate each iterations

4 Results / Evaluation

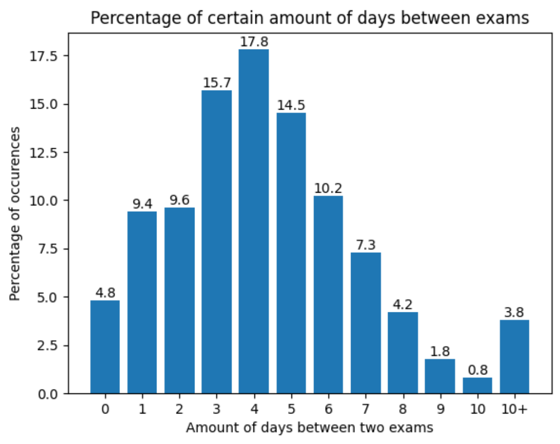
In order to evaluate the results from the experiment, we have to verify whether the phases perform their objective as described. Firstly, does the initialisation phase succeed in generating a 'feasible' solution namely one without or with a minimal number of hard constraint violations? Secondly, does the optimisation phase succeed in transforming the initial solution into one with an acceptable solution?

4.1 Reference solutions

The actual exam schedules, manually created by the administration of the University of Antwerp, can be used as a baseline to compare the generated schedules and showcase the priorities. These exam distributions can be seen in Figures 2 and 3. Most notably, the manual schedules have a focus on keeping the amount of same day exams to a minimum and having 3+ days in between exams as much as possible. This distribution is especially visible in the schedules for the faculty FTI. In order to be considered a superior solutions, automatically generated schedules will have to minimise the amount of exams with fewer than 2 days in between with a focus on same day exam violations.



(a) FTI

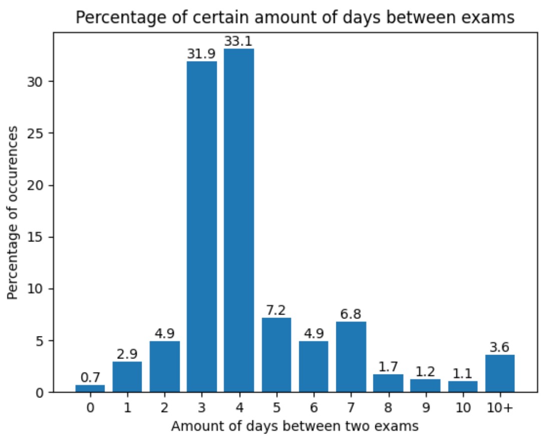


(b) FWET

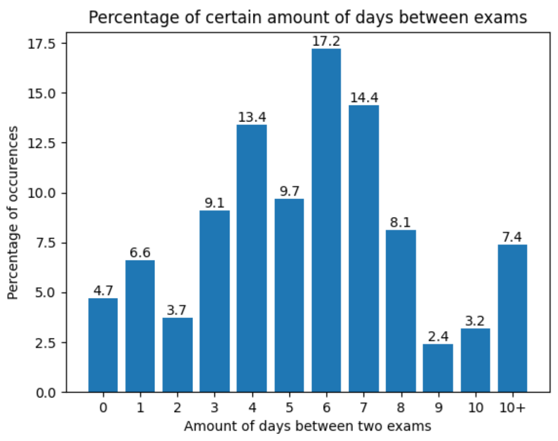
Figure 2: Manual exam distribution for January 2021

4.2 Initialisation phase

In order to quantify the feasibility of a solution, it is possible to look at either the objective function or the amount of hard constraint violations. Since the objective function was adapted to contain exponents and the conflict weight having a large impact on the size of the objective, looking at the amount of hard constraint violations can provide



(a) FTI



(b) FWET

Figure 3: Manual exam distribution for June 2021

a better perspective. By plotting the violations per iterations, the progress made by the initialisation phase can be visualised. Figure 4 shows that the initialisation phase is able to generate a solution for FTI+FWET without hard constraint violations. It reaches a feasible solution after fewer than 150 iterations, generally taking under 500 seconds. However, Figure 5 shows that no attention was given to the distribution which can be confirmed by the presence of a high percentage of exams after 1 day.

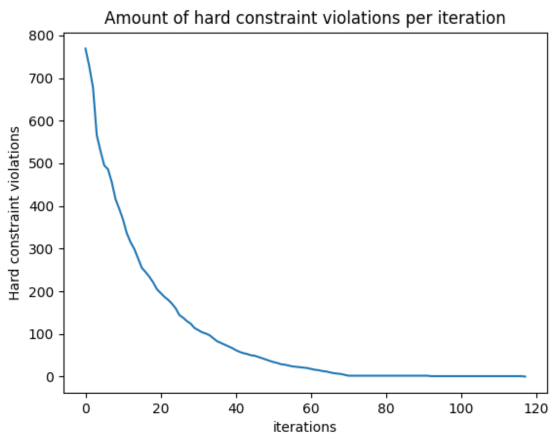


Figure 4: Hard constraint violations

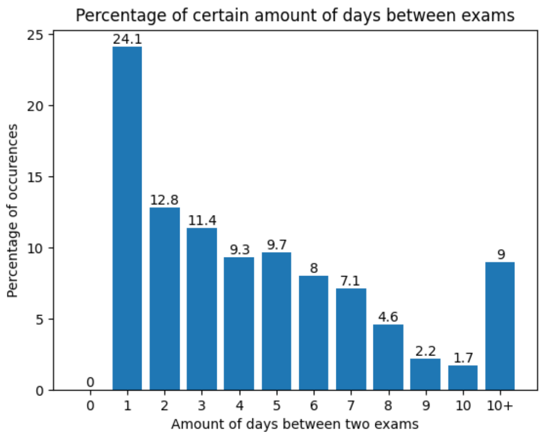
4.3 Optimisation phase

While we have shown that tabu search can efficiently generate a feasible solution from the provided data set, the distribution will determine whether the automated schedules are able to beat the manual schedules.

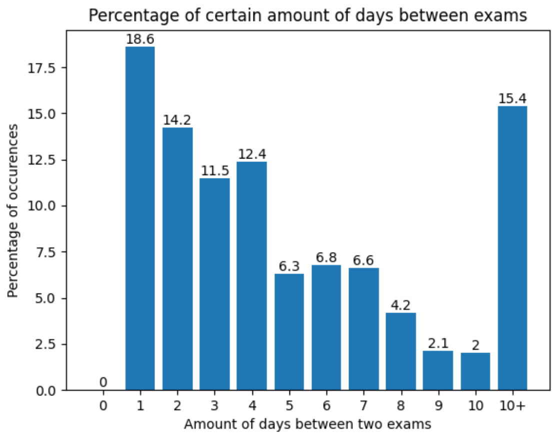
5 Threats to Validity

6 Related Work

7 Conclusion



(a) FTI



(b) FWET

Figure 5: Exam distribution after initialisation phase for June 2021

References

[1] Fred Glover and Manuel Laguna. *Tabu search I*, volume 1. 01 1999. pages 3

[2] Ramón Álvarez Valdés-Olaguíbel, Crespo E., and Tamarit Goerlich José Manuel. A tabu search algorithm to schedule university examinations. *Qüestiió: quaderns d'estadística i investigació operativa*, 21(1), 1997. pages 5