1.Describe in how the CSP was formulated, namely the variables, domains, and constraints considered.

The variables of the CSP are defined as a list of strings, each string being one variable and each variable contains the student classes that have a specific weekly class and that weekly class. The size of the variables list is the same as the number of weekly classes.

Variable example from the input file at the bottom: ‘MEEC,MEAer,|IASD,T,1’

The domains of the CSP are defined as a dictionary that has the variables as keys and a list of strings as the values of each key. Each string of those lists contains a timetable slot and a room. The values of each key are sorted in the ascending order of the time slot (hour) of the timetable slot, in order to have the backtracking algorithm selecting the values with the lowest time slot before the ones with the largest time slot, so a solution with the lowest hour of the latest class over all

weekdays is found faster. The values of each key (list of strings) have the size of TxR, being T the number of timetable slots available and R the number of rooms available.

Domain example from the example input file at the bottom: MEEC,MEAer,|IASD,T,1: [‘Mon,8|EA1’, ‘Mon,8|EA2’, ‘Tue,8|EA1’, … , ‘Thu,10|EA2’]

The constraints of the CSP considered were binary constraints, unary constraints and global constraints. (CONFIRMAR ESTA 1ª FRASE COM O RESTO, GLOBAL?). The constraint graph is fully connected, all the variables have binary constraints with all the remaining variables.

Constraints implemented:

- Two different weekly classes cannot occur at the same timetable slot in the same room. (Binary constraint)

- Two different weekly classes of the same course cannot occur at the same timetable slot. (Binary constraint)

- The i-th weekly class of a course and kind cannot occur at the j weekdays, if j is smaller than i, being j an index that represents the weekdays, ie, a weekly class with index 2 cannot occur on Mondays, a weekly class with index 3 cannot occur on Mondays neither on Tuesdays, etc. (Unary constraint)

- The (i+1)-th weekly class of a course and kind cannot occur before or on the same weekday as the i-th weekly class of the same course and kind (Binary constraint)

- A student class cannot have two different weekly classes at the same timetable slot. (Binary constraint)

- The number of timetable slots available should be bigger than the maximum number of different weekly classes the student classes have. (Binary/Unary/Global constraint ?).

- The hour of the latest class over all weekdays should be the minimum possible. To implemented this constraint the CSP problem ran several times, calculating the hour of the latest class over all weekdays in between runs and setting all the time slots that were larger or equal than the previous hour of the latest class over all weekdays found to unavailable (removing from the domains and running the backtracking algortihm again with shorter domains). (Unary constraint)

Input file example:

T Mon,8 Mon,9 Mon,10 Tue,8 Tue,9 Tue,10 Wed,8 Wed,9 Wed,10 Thu,8 Thu,9 Thu,10

R EA1 EA2

S MEEC MEAer

W IASD,T,1 IASD,T,2 IASD,PB,1 SAut,T,1 SAut,T,2 SAut,L,1 SAut,L,2 SAut,L,3

A MEEC,IASD MEAer,IASD MEAer,SAut

2.Indicate which heuristics and constraint propagation (inference) methods were used, and justify your choice.

After analyzing the available heuristics, the ones chosen were the MRV (Minimum Remaining Values) and the LCV(Least Constraining Value), since they were the ones that made more sense in this problem. The choice was between the MRV, the degree heuristic, the least constraining value and the first unassigned variable first. The last one is the simplest strategy, where the algorithm chooses the next variable available to process, without any other criteria. This is hardly the best choice.

For the degree heuristic, it attempts to reduce computational cost by selecting the variable that is involved in the largest number of constraints on other unassigned variables. Has for this strategy, since our problem doesn’t have a large number of constraints, this wouldn’t be a relevant comparison, giving that most of the variables would share the same number of constraints.

For the LCV is a complement to the variable selection heurist. Having the variable selected, the idea is to choose the domain element to branch that gives more possible options to the next variables to be processed. This way, a single solution is found faster.

Finally, the MRV is used as the variable selection heuristic. This method selects the variable with the least options remaining in its domain, for a faster converging solution. Choosing the variable that has the least options available is a optimized way to cause a failure faster, reducing the number of backtrackings needed to reach a solution by reducing the number of unnecessary branches. The only fault in this heuristic is that it’s no help in choosing the initial variables to process.

As for the inference methods tested, they were the forward propagation and MAC(Maintaining arc consistency). The one chosen was forward propagation as it’s explained next.

The forward propagation is the simplest propagation method. It simply checks the chosen variable’s neighbor’s domains after a selection has been made and deletes the elements of the domains that incompatible with the value chosen. Although forward checking detects many inconsistencies, it does not detect all of them. The problem is that it makes the current variable arc-consistent, but doesn’t look ahead and make all the other variables arc-consistent.

The MAC inference method does just that, besides checking if the current variable is arc-consistent it also looks ahead. This makes it more powerful than forward checking, since a value choice can immediately cause an empty set in a certain domain, making it not viable, but it may not be in the direct neighbors of the current variable, and when this happens, the csp knows immediately to backtrack. It also has a heavier computational cost.

For this last reason and for the fact that forward checking and MRV have a natural compatibility, forward checking was chosen. Since our optimization algorithm implies running the csp several times depending on the result, it’s logical that it might not be a good idea for it to be to heavy to run. Therefore, the MAC was not a good option. It was verified that the same results were obtained for the available tests using booth inference methods, always having a faster result with forward checking.