

Theoretical Exercise Sheet 5.

Solutions due Tuesday, June 9, 23:59 uploaded in the AI CMS.

Total points of the sheet: 24

Exercise 1: Minimax Search

8 Points

Max and Minnie are playing a classic children's game called Tic-tac-toe in which players take turns marking the cells in a 3×3 grid. Max marks Xs and Minnie marks Os. Minnie wins with utility -1 if **any line (horizontal, vertical, or diagonal)** fills up with three Os, whereas Max wins with utility $+1$ if **any** line fills up with three Xs. If there are no empty cells left and no one has won so far, the game ends in a draw with utility 0 .

- (a) Consider the state depicted below. Here, it is Max's turn to play.

o		o
		o
x	x	

Draw the **full** Minimax tree and annotate every node with its utility.

- (b) Consider the evaluation function $f(x) :=$ the number of (horizontal) rows which contain at most one O. For example, in the initial state $f(x) = 2$.

Draw the Minimax tree with this evaluation function and a depth of 2. Annotate every node with its utility.

- (c) Assuming a perfectly playing opponent, Minimax search without a depth limit will always guarantee a draw. However, it is not obvious whether this guarantee can still be made when imposing certain depth limits in combination with certain evaluation functions. As an example, consider the simple evaluation function g aimed to detect situations in which the opponent can win in one step (if it is their turn).

$$g(x) := \left\{ \begin{array}{ll} -1, & \text{if there is a line (horizontal, vertical, or diagonal) with} \\ & \text{two opponent marks and an empty field} \\ 0, & \text{otherwise} \end{array} \right\}$$

Now prove or disprove the following claim:

Running Minimax search with a depth limit of 3 and evaluation function g is sufficient to guarantee a draw against a perfectly playing opponent in a 3×3 Tic-tac-toe game. You may **not** assume that you are allowed to start the game.

Solution:

1. See Figure 1.

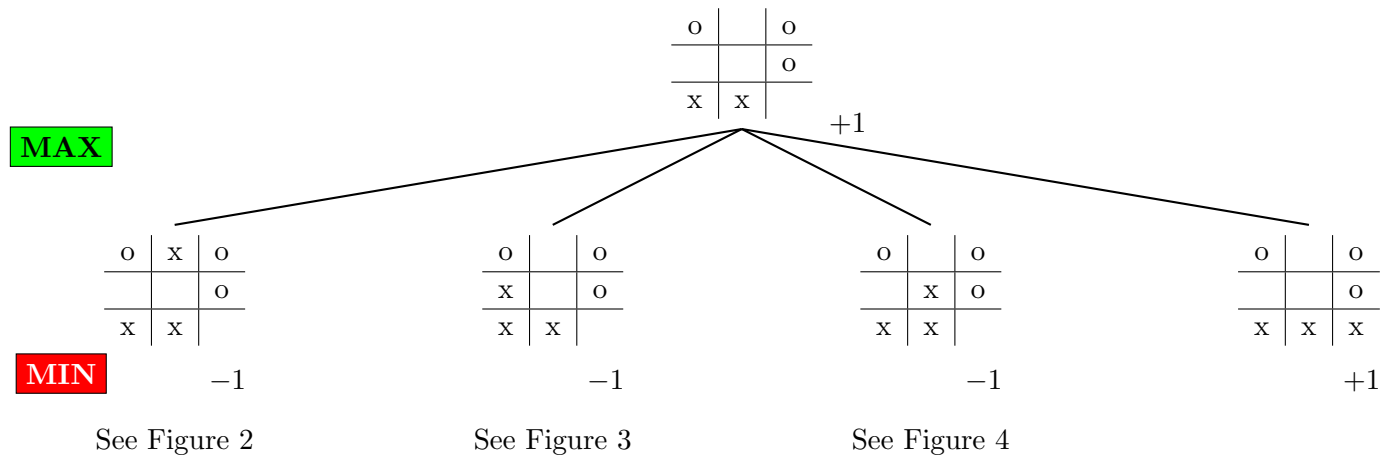


Figure 1: Solution of Exercise 1 (a).

MIN

MAX

MIN

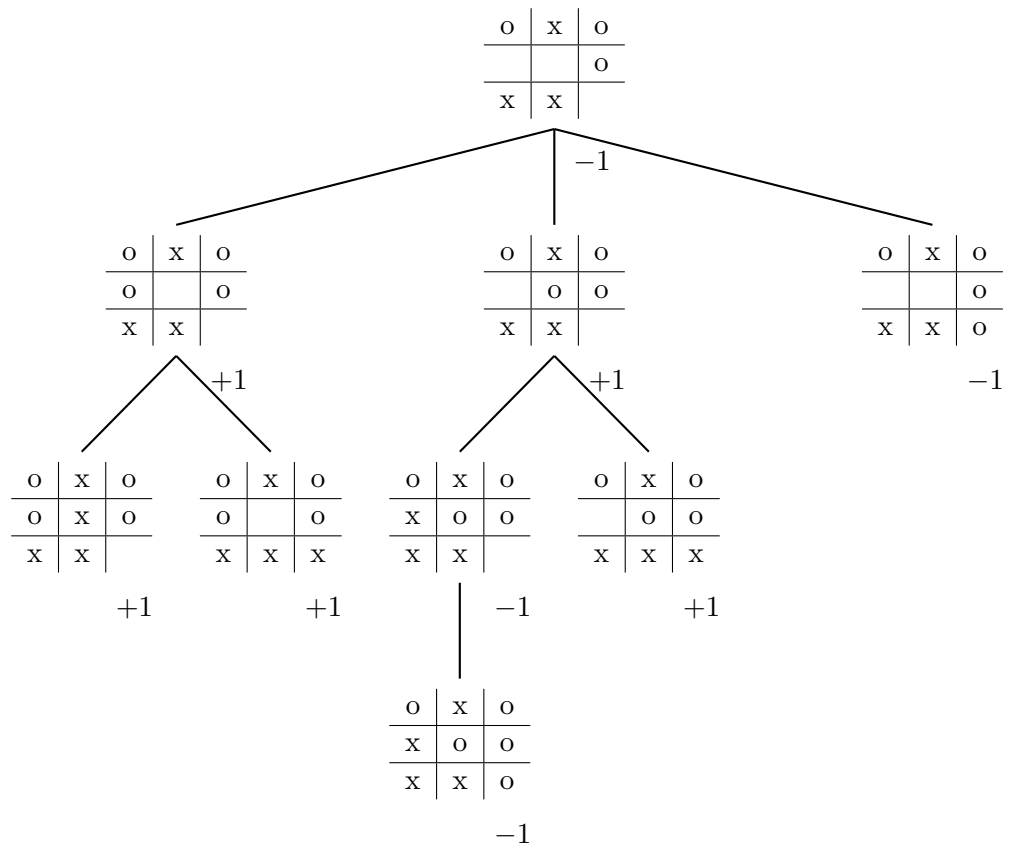


Figure 2: Continuation of Figure 1.

MIN

MAX

MIN

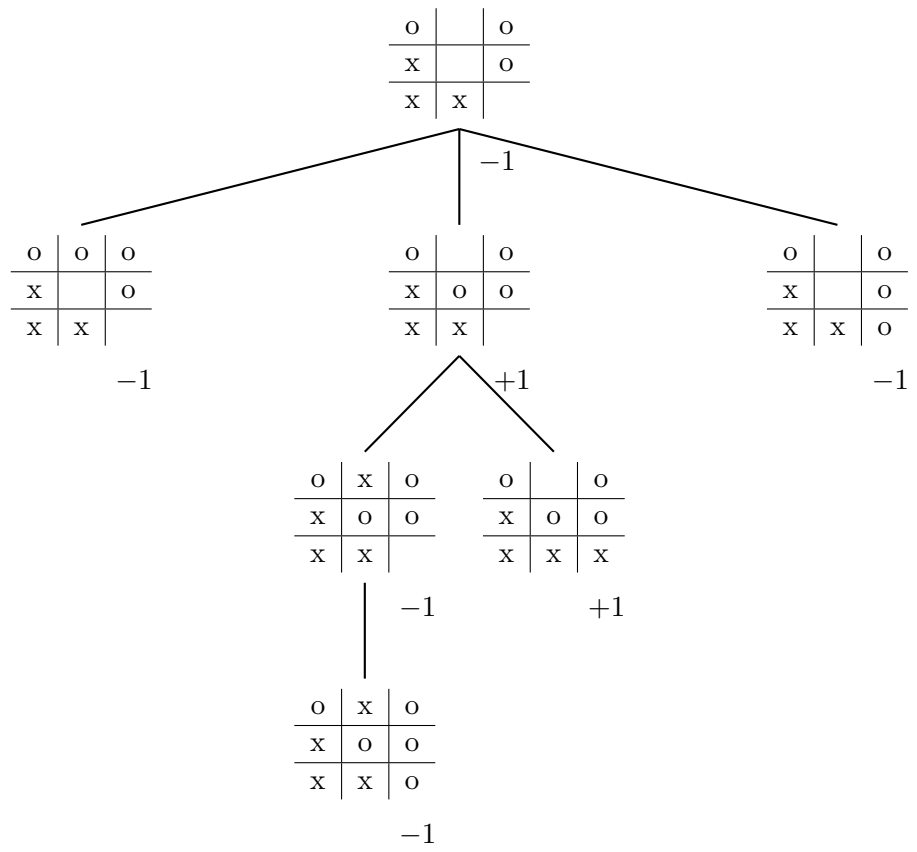


Figure 3: Continuation of Figure 1.

MIN

MAX

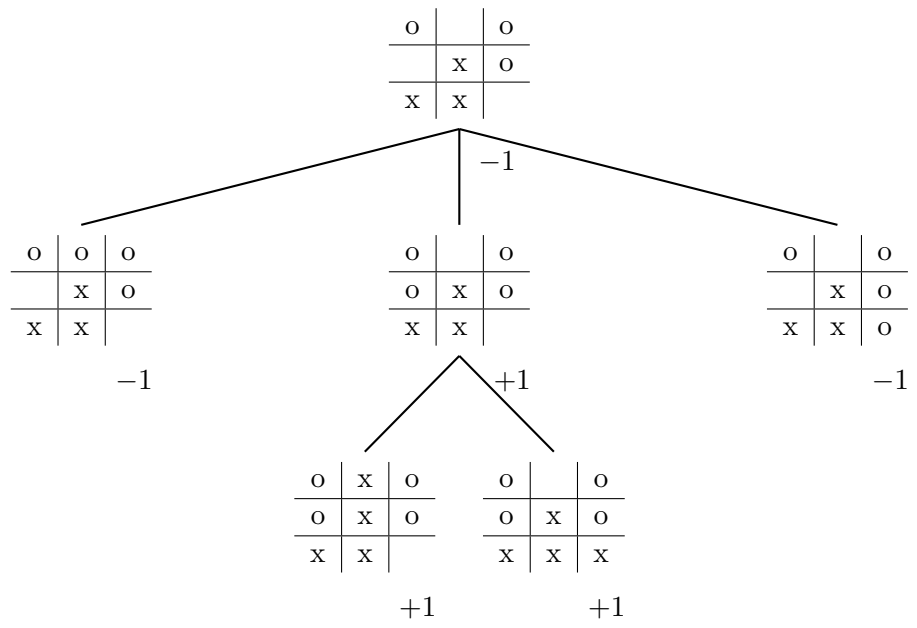


Figure 4: Continuation of Figure 1.

2. See Figure 5.

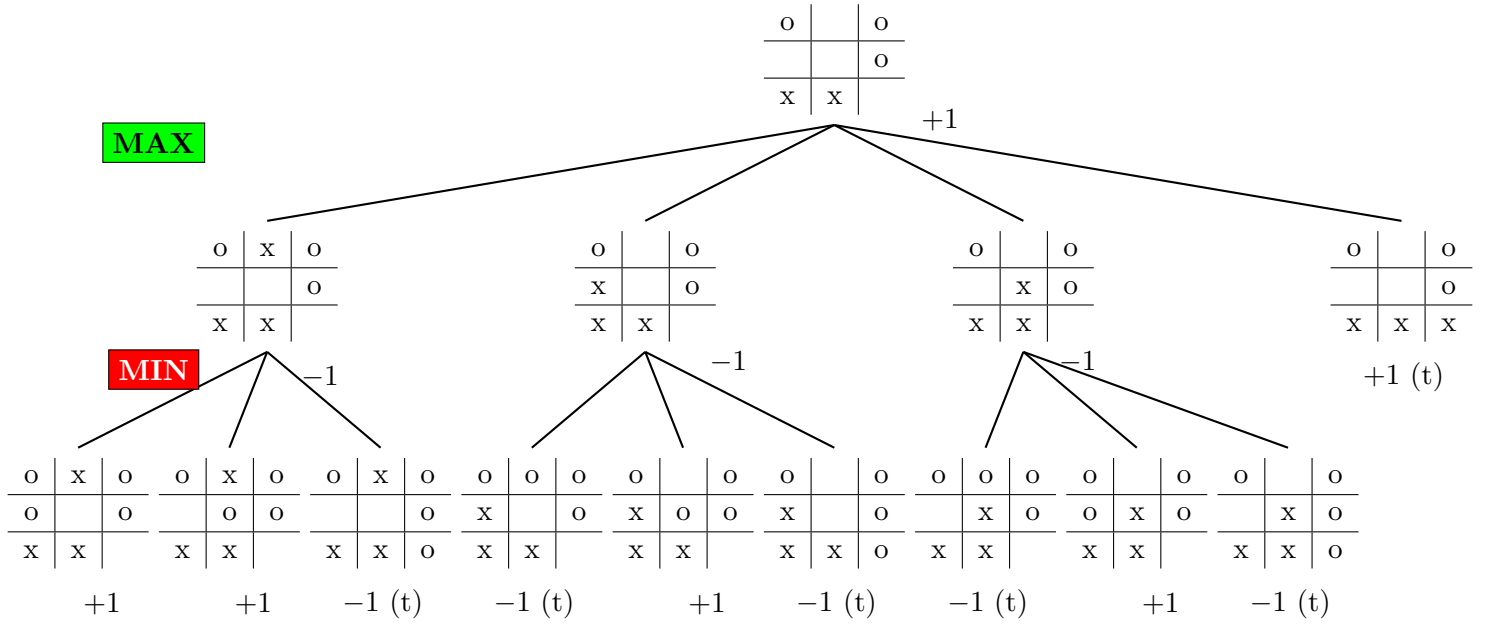
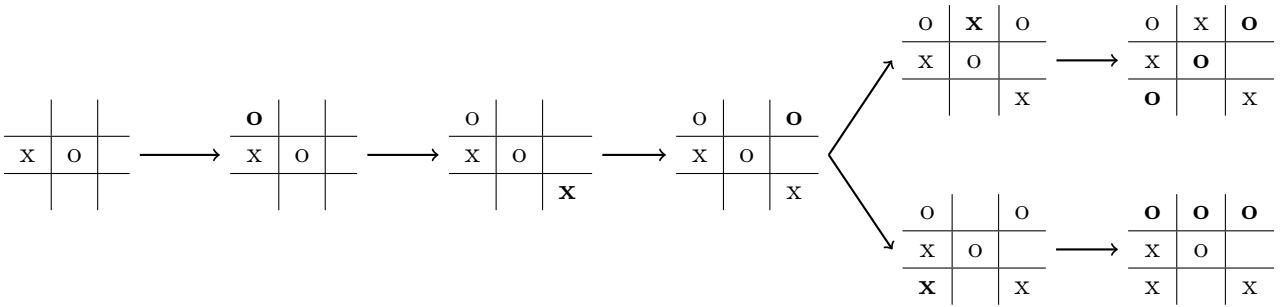


Figure 5: Solution of Exercise 1 (b). Nodes marked with (t) are terminal nodes.

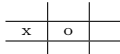
3. The claim is false. We prove this on the basis of the following counterexample.

Let us assume we are playing as Max and our opponent Minnie has begun the game by placing an O in the centre. Up to symmetry, there are only two ways in which we can respond: We can place an X in a corner or we can place the X in the middle of a row or column. If we do not mark a corner, Minnie has a winning strategy, which is depicted below.



Thus, to avoid to risk of being defeated, Minimax search must recognize this failing move. In our case this means that the utility of placing an X in the corner must be strictly greater than the one for failing to do so. As we cannot win the game within

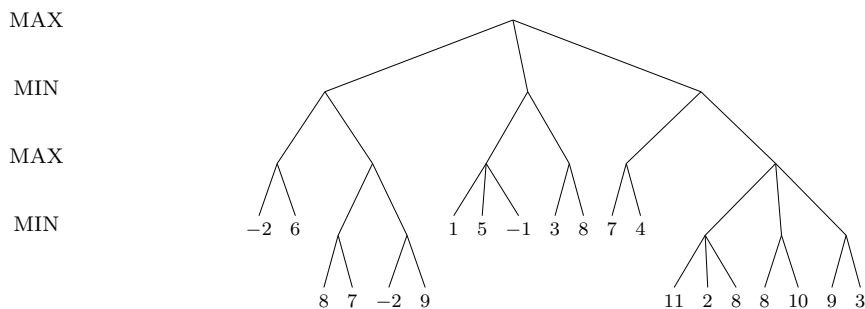
four (half-)moves (because we need 3 own marks), the utility value for putting an X in the corner is ≤ 0 . Therefore the utility value for putting the X somewhere else must be -1 .

However, it is easy to see that Minimax search starting in  with a depth limit of 2 (because we have already made the first move) will produce a utility value of 0. This is because, no matter where Minnie puts her next O, there is always at most one line with 2 Os and an empty field after the next step. Thus Max can force a state in which g evaluates to 0 simply by putting an X in that respective line.

Exercise 2: Alpha-Beta pruning

11 Points

Consider the following game tree corresponding to a two-player zero-sum game as specified in the lecture. As usual, **Max is to start in the initial state (i.e., the root of the tree)**. For the following algorithms, the expansion order is **from left to right, i.e., in each node the left-most branch is expanded first**.

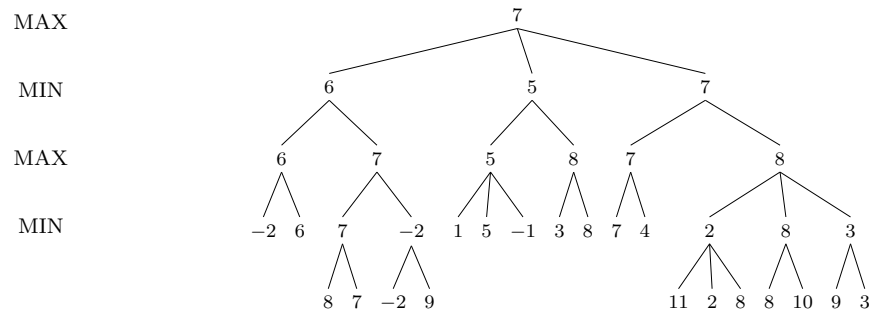


1. Perform Minimax search, i.e., annotate all internal nodes with the correct Minimax value. Which move does Max choose?
2. Perform Alpha-Beta search. Annotate all internal nodes (that are not pruned) with the value that will be propagated to the parent node as well as the final $[\alpha, \beta]$ window before propagating the value to the parent.
Mark which edges will be pruned. How many leaf nodes are pruned?
3. There are move orderings that yield more effective pruning. Reorder the nodes of the tree such that as many leaf nodes as possible are pruned. How many leaf nodes are pruned? A node counts as pruned in this context if it is not expanded or evaluated. Perform Alpha-Beta search on this tree as in the previous exercise.

4. Develop a general recipe to reorder the nodes of a Minimax tree such that a maximal number of leaf nodes is pruned in Alpha-Beta pruning. No formal algorithm is required.

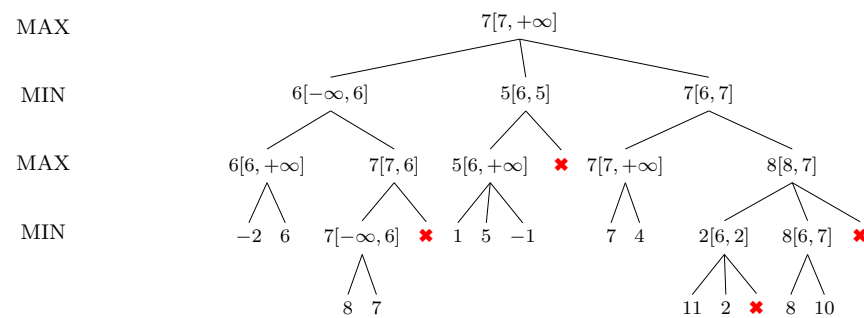
Solution:

1. The Minimax tree:

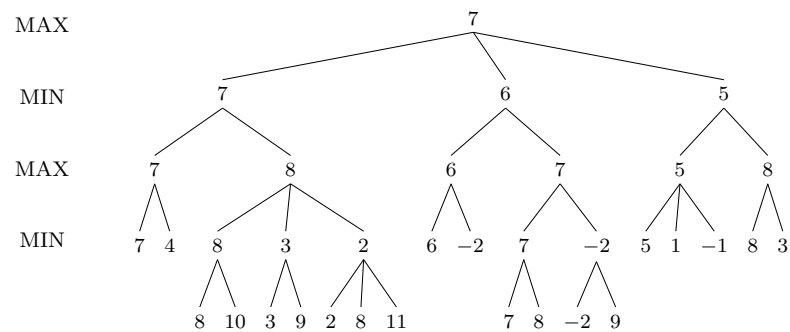


Max chooses the right-most action towards the child node with value 7.

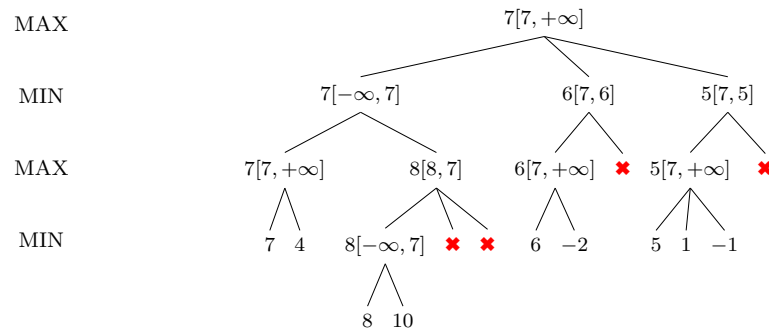
2. The Alpha-Beta tree (7 leaf nodes are pruned):



3. The optimally reordered Minimax tree:



The Alpha-Beta tree with 11 pruned leaf nodes:



4. To prune a maximum number of leaf nodes with Alpha-Beta pruning reorder the nodes in the following way:
- Place the optimal goal value on the leftmost branch.
 - Place the children of Min-nodes in order of increasing Minimax value.
 - Place the children of Max-nodes in order of decreasing Minimax value.

Exercise 3: True or False

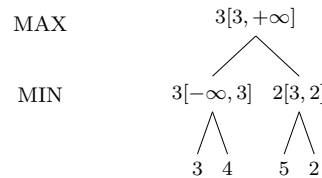
5 Points

Please decide for each of the following statements whether it is true or false and justify your answer (1–3 sentences per statement).

1. In Alpha-Beta pruning, all unexplored successors of a Min-node n will be pruned if one of the explored successors of n has a higher utility score than the lowest Min-node utility already found on the path to n .
2. There exists a Minimax tree in which Alpha-Beta pruning will not prune any leaf nodes.
3. For every Minimax tree with at least 42 nodes there is a move ordering such that Alpha-Beta pruning will prune at least one leaf node.
4. Full Minimax search always yields the best possible outcome in terms of its utility, no matter how the opponent plays.
5. A two-player zero-sum game has exactly three possible outcomes in terms of its utility function.

Solution:

1. False. The unexplored successors of Min-node n get pruned if one of the explored successors of n has a **lower or equal** utility score than the **highest Max-node** utility already found on the path to n .
2. True. See the following counterexample for a Minimax tree in which no leaf node is pruned:



3. False. Assume that every node has exactly one successor. Then there is only one leaf node, which will not be pruned.
4. False. It yields the best possible outcome assuming the opponent plays perfectly.
5. False. In some two-player zero sum games there is a larger variety of possible outcomes (e.g. the payoffs in backgammon range from +192 to -192).

Submission Instructions

Solutions need to be packaged into a **.zip** file and uploaded in the AI CMS. The **.zip** file has to contain a single folder with name:

AI2020_TE5_mat1_mat2_mat3 where **mat1**, **mat2**, **mat3** are the matriculation numbers of the students who submit together. This folder must contain the following files:

- **authors.txt** listing the names and matriculation numbers of all students who submit together. Use one line per student and no spaces: Name;Matriculation number.
- The **.pdf** file containing your solutions.

Do not add any other folder or sub folder, this means place all files directly into **AI2020_TE5_mat1_mat2_mat3**. Do not place any file outside of **AI2020_TE5_mat1_mat2_mat3**.

Only one student of each group needs to do the submission! Remember that this sheet can be submitted in groups of up to three members (all members of the group must however be assigned to the same tutorial). If you are still looking for submission partners, it is recommended to ask in the forum of the CMS in the category “Student Room”.