Assignment A3: Team XX

Firstname1 Lastname1 Firstname2 Lastname2 Firstname3 Lastname3

1 Agent Description

Explain, *in your own words*, the workings of your AI agents. (It can also be a single agent; we use the plural in the rest of this template, but you can replace that by singular if that is your situation.) Pay special attention to the following:

- Include a concise description of the basic agent to which you have added new functionality. This is in general your team's A2 agent. You are however free to add or even remove functionality that falls in the general 'minimax' and 'heuristic' categories of the two previous assignments.
- Explain all the techniques that you have chosen to implement. Consider using pseudocode for this, and then also provide an explanation of the pseudocode. (N.B.: The pseudocode does not need to cover all actual code.)
- How do your agents use the (unknown) amount of time available for computing the next move? Can they reasonably be expected to find a better move when given more time? Focus on the new features you have included.
- Explain the evaluation function used for numerically scoring board positions. What is the intuition/reasoning behind it?
- For all of the points above, as a general rule of thumb, consider whether the explanation is clear to your fellow students. Whenever you base your design on existing work/methods, provide references to the appropriate literature.

Do not hesitate to go beyond these points above in case relevant aspects of the design are not captured by any of them.

2 Agent Analysis

In contrast to the earlier assignments, you have complete freedom to choose how you analyse (comparative or absolute) agent performance. The goal is to choose a suite of experiments and analyses that make it possible for you to convincingly discuss the performance of the agents in a wide variety of circumstances (compute-time, board-size, board-type, adversary-type,...).

As before:

- Make sure you clearly explain the setup of your experiments.
- Create plots and/or tables to summarize the experimental results or provide insight that can be gained from them. Basic plots you could use are, for example, those of the average performance (e.g. win/loss/draw statistics) of your agent as a function of compute-time.

Table 1: This is a table. Notice the position of the caption.

left	right
this	is
a	table

In your report, include the interesting plots and tables that you have generated. Also provide some analysis; can you qualitatively summarize the observed performance of your agents on each of these tasks? Moreover, can you think of a possible explanation for these observations?

3 Motivation & Reflection

What do you think are strong points of your agents' design and of your experimental design? Which properties of the problem domain do the agents exploit and which aspects do your experiments test? Is there an inherent strength to the functionalities you chose for playing the game of competitive sudoku?

Conversely, what do you think are weak points of your agents' design and of your experimental design? Are there certain properties of the problem domain for which you think your designs are unsuitable? Which game aspects do your experiments not test? Is there an inherent weakness to the functionalities you chose for playing the game of competitive sudoku?

General Guidelines for Your Report

Please remove this section from your submitted report

This section contains some general guidelines that you should keep in mind when preparing your report:

- The **maximum** length of your report is **7** pages, **excluding** references and the appendix with your included Python code, but **including** all figures and tables.
- Please do not modify the style of the template to squeeze in more content. Make sure all text is legible (not too small).
- Please carefully proofread your report for typos and grammatical errors; use a spellchecker.
- References to literature are best added in the `report_template.bib` BibTeX file, and invoked in your report using a citation command. For example, "Shannon [1950] proposed an approach for ..." or "... as discussed in the literature [Shannon, 1950]".
- Tables can be added through the `\tabular` command, for example Table 1.
- Figures and graphs are best added through the `figure` environment, possibly using the `\includegraphics` command; see e.g. Figure 1.
- Your code should be included in the appendix at the end of this file meant for that. The appendix should not be used for anything else.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Figure 1: Left: Example of a sudoku puzzle with $n = m = 3$. Right: The solved puzzle; original clues are in black, with the values to be entered shown in red.

References

Claude E. Shannon. XXII. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, 1950. doi:10.1080/14786445008521796.

Python files

In this appendix, you must include all your Python files, so that the reviewers can evaluate your code. So this must at least be your `sudokuai.py` file, but might include any support files. Make sure that the file name (and path) is explicitly mentioned, so that it is clear how the code is used. Avoid too-long code lines that need to be broken over multiple lines here.

Code Listing 1: Basic agent: `team00_A2/sudokuai.py`.

```
# (C) Copyright Wieger Wesselink 2021. Distributed under the GPL-3.0-or-later
# Software License, (See accompanying file LICENSE or copy at
# https://www.gnu.org/licenses/gpl-3.0.txt)

import random
import time
from competitive_sudoku.sudoku import GameState, Move, SudokuBoard, TabooMove
import competitive_sudoku.sudokuai


class SudokuAI(competitive_sudoku.sudokuai.SudokuAI):
    """
    Sudoku AI that computes a move for a given sudoku configuration.
    """

    def __init__(self):
        super().__init__()

    # N.B. This is a very naive implementation.
    def compute_best_move(self, game_state: GameState) -> None:
        N = game_state.board.N

        def possible(i, j, value):
            return game_state.board.get(i, j) == SudokuBoard.empty \
                    and not TabooMove(i, j, value) in game_state.taboo_moves

        all_moves = [Move(i, j, value) for i in range(N) for j in range(N)
                     for value in range(1, N+1) if possible(i, j, value)]
        move = random.choice(all_moves)
        self.propose_move(move)
        while True:
            time.sleep(0.2)
            self.propose_move(random.choice(all_moves))
```

Code Listing 2: Agent with this: `team00_A3/sudokuai.py`.

```
# (C) Copyright Wieger Wesselink 2021. Distributed under the GPL-3.0-or-later
# Software License, (See accompanying file LICENSE or copy at
# https://www.gnu.org/licenses/gpl-3.0.txt)

import random
import time
from competitive_sudoku.sudoku import GameState, Move, SudokuBoard, TabooMove
import competitive_sudoku.sudokuai
```

```
11 class SudokuAI(competitive_sudoku.sudokuai.SudokuAI):
12     """
13     Sudoku AI that computes a move for a given sudoku configuration.
14     """

16     def __init__(self):
17         super().__init__()

19     # N.B. This is a very naive implementation.
20     def compute_best_move(self, game_state: GameState) -> None:
21         N = game_state.board.N

23         def possible(i, j, value):
24             return game_state.board.get(i, j) == SudokuBoard.empty \
25                     and not TabooMove(i, j, value) in game_state.taboo_moves

27         all_moves = [Move(i, j, value) for i in range(N) for j in range(N)
28                      for value in range(1, N+1) if possible(i, j, value)]
29         move = random.choice(all_moves)
30         self.propose_move(move)
31         while True:
32             time.sleep(0.2)
33             self.propose_move(random.choice(all_moves))
```

Code Listing 3: Agent with that: `team00_A3_extra1/sudokuai.py`.

```
1 # (C) Copyright Wieger Wesselink 2021. Distributed under the GPL-3.0-or-later
2 # Software License, (See accompanying file LICENSE or copy at
3 # https://www.gnu.org/licenses/gpl-3.0.txt)

5 import random
6 import time
7 from competitive_sudoku.sudoku import GameState, Move, SudokuBoard, TabooMove
8 import competitive_sudoku.sudokuai


11 class SudokuAI(competitive_sudoku.sudokuai.SudokuAI):
12     """
13     Sudoku AI that computes a move for a given sudoku configuration.
14     """

16     def __init__(self):
17         super().__init__()

19     # N.B. This is a very naive implementation.
20     def compute_best_move(self, game_state: GameState) -> None:
21         N = game_state.board.N

23         def possible(i, j, value):
24             return game_state.board.get(i, j) == SudokuBoard.empty \
```

```
25                  and not TabooMove(i, j, value) in game_state.taboo_moves

27          all_moves = [Move(i, j, value) for i in range(N) for j in range(N)
28                       for value in range(1, N+1) if possible(i, j, value)]
29          move = random.choice(all_moves)
30          self.propose_move(move)
31          while True:
32              time.sleep(0.2)
33              self.propose_move(random.choice(all_moves))
```