AI report

# 1.1 Introduction

Sudoku is a logic-based puzzle which you have to fill in numbers from 1 to 9. It comprises of 9×9 grids so that the numbers 1 to 9 appear only one time in each row, each column, and each 3×3 sub grids (blocks more generally). Each Sudoku has only one unique solution, so there is no second way to fill the 81 grids.

Our aim of doing this project is to create a Sudoku solver using the artificial intelligence algorithm such as search algorithms and first-order logic. As Lisp and Prolog are taught in laboratory, so we are using these two programming languages to create a small program that can solve every kind of Sudoku puzzle. Besides, Lisp and Prolog are powerful languages in Artificial Intelligence (AI) aspect because they handle list and perform logical reasoning extremely outstanding, respectively.

# 1.2 Objectives

The following is our objectives this project:

* Understanding the logic and the algorithms used behind Sudoku puzzle
* Figure out the algorithms that solve every Sudoku puzzles using search
* Learn more about Lisp and Prolog.

# 1.3 Problem Statement

People solve Sudoku puzzle using logic reasoning, but what AI performs to solve it like a human being? As AI major students, we are very curious about and wish to figure out the ways or algorithms that machines use to solve the Sudoku puzzle.

# 2.1 Background

Before going further, let us know where is the origin of Sudoku and what it means “Sudoku”? The origin of Sudoku is from Switzerland (By a Mathematician Leonhard Euler) and then travels to Japan by way of America. “Su” is the Japanese character with the meaning of number and “doku” means single. In another word, it called single number which means that for each column, each row, and each sub grid, it can only contain the digit 1-9 without repetition.

In this project, we need to consider the AI search algorithms, such as breath-first-search, depth-first-search and A\* heuristic search. After figuring the concept of the search algorithms, we have to create a program that can solve the Sudoku puzzle.

Sudoku is a game that suits every generation of the people. There can be many level of difficulty in Sudoku puzzles.

# 2.2 Literature Reviews

## 2.2.1 The Science behind Sudoku

Sudoku is a game of numbers, but it does not have anything to do with mathematics such as addition or multiplication. A Sudoku grid is a special kind of Latin square that named by an 18th century mathematician-Leonhard Euler. Latin square is an n x n matrices that are filled in with n symbols in a way that the same symbols appear exactly once in same row and column.

The number of valid Sudoku grids is 6,670,903,752,021,072,936,960 which proved by the use of logic and computers. The minimum numbers of hints that a 9 x 9 Sudoku puzzle can start with and the solution can still remain unique seem to be 17. This mean for any Sudoku which is having the hints that less than 17 will make it cannot guarantee a unique solution.

When discovering the algorithms, we have found two existing Sudoku solver online which is Peter Norvig’s Sudoku solver implementation and Nick Smallbone's Sudoku solver implementation.

## 2.2.2 Peter Norvig’s Sudoku Solver Implementation

Peter Norvig has created two way to solve the Sudoku puzzles, one is Constraint Propagation and another is Search. The code is written in Python. After reading all the codes, we figure out that he is using dictionary (a hash table in Java) to store every nodes’ possible numbers.

In constraint propagation, he defines a function called eliminate() that eliminates the duplicate number from the dictionary if the peers have already contained the number. The peers including the cells in the same row, same column and same block. The solver can solve the easy Sudoku puzzle rapidly. However, when it comes to hard puzzle, unfortunately, that will not the case.

In Search, he uses the depth-first search algorithm to recursively search the possible answers. To avoid bookkeeping complications, he creates a new copy of values for each recursive call to search. So each search tree is independent and does not confuse another branch. Besides, he also uses backtracking search that keep track of each change to values and undo the change when hit a dead end. Instead of trying a number that having 6 possible numbers, it will try for the least possible numbers, for example, 2. Hence, the possibility of correctly guessing the number will be higher. Surprisingly, this method solves every Sudoku puzzle, including hard level.

## 2.2.3 Nick Smallbone's Sudoku solver implementation

The algorithm is to solve almost any Sudoku puzzle. We went across the code in order to understand the idea of his implementation.

First thing to do is to go through all empty cells and check for all possibility numbers from 1 to 9 that can be placed on it. The algorithm uses an array to have all the possibilities for each cell. Looking at the code, it looks like the code implements a depth first search and best first to look for a solution. In order to solve the puzzle, the procedure below describes it;

1. The function findMin() is called to find the cell with smallest number of possibility.
2. If a cell has only one valid possibility, then fill this number in the cell.
3. If the cell has more than one possibility, then it calls solve() method which continue recursively until all cells are occupied.
4. Then the solution is found [8].

This algorithm provides a new way to have a list of all possible valid numbers for each cell. As a result, this improves the performance comparing to Guy’s solver that uses an array.

# 3. Detailed Project Description

We are given the title “Solving the Sudoku Problem using State Space Search Techniques” as our assignment. Since our lecturer has taught us Lisp and Prolog in class, we decide to create one or two programs that are written in Lisp or Prolog that can solve every Sudoku games. Besides, we are trying to include one or two of those search in our program to solve Sudoku puzzles, such as uninformed search (breath-first search and depth-first search) and heuristic search (best-first search, greedy search and A\* search).

As mentioned above, Sudoku is a logic-based combinatorial puzzle game. The objective of the game is to take a 9×9 grid and fill in the open spots with numbers from 1 to 9 so that each column and each row of the grid contains each of the numbers only once. Furthermore, each of the nine 3×3 sub-grids that together compose the total 9×9 grid (also called boxes, blocks, regions, and sub-squares) must contain all of the digits from 1 to 9 only once.

We have to create a program that has the following features:

1. takes a Sudoku puzzle game data (may be an array or a list) as input,
2. consists of a search algorithm and apply it,
3. runs smoothly without any bug or anything that causes the program not responding,
4. returns the solution as fast as possible and
5. prints out the solved Sudoku board.

After studying few literature reviews, we discover that normal depth-first search may be the simplest algorithm because it recursively searches for the next possible number. However, this is not enough for solving hard level of Sudoku. We have to add backtracking algorithm and some more constraints to the algorithm so that it will not search blindly. This will make the search process more efficiently and take less time for solving the puzzle.