Advances in Social Science, Education and Humanities Research, volume 410
1st International Multidisciplinary Conference on Education, Technology, and Engineering (IMCETE 2019)

Software Design Completion of Sudoku Game with Branch and Bound Algorithm

1st Rustam Effendi
Program Studi Tehnik Informatika,
Fakultas Ilmu Komputer
Universitas Banten Jaya
Banten, Indonesia
rustameffendi@unbaja.ac.id

2nd Indra Gunawan
Program Studi Tehnik Informatika,
Fakultas Ilmu Komputer
Universitas Banten Jaya
Banten, Indonesia

3rd Yasin Efendi
Program Studi Tehnik Informatika
Sekolah Tinggi Ilmu Informatika
NIIT
Indonesia

***Abstract*—In our life there are many of games. The uniqueness game of numbers makes this game very exciting, and at the same time can be used to train the intelligence. Sudoku is a puzzle game of logic-based numbers. Rules of the game are simple, fill all the matrix to the brim, with notes, for each column, row, or sub matrix only be filled with the numbers 1 through 9, as each of the figures. Branching algorithm and a restriction in the search algorithm in a systematic solution space, so it is very effective and efficient to be implemented in the problem. For it is necessary to build learning applications of artificial intelligence in particular to the material tracking system so that students can use as a learning tool and facilitate faculty in presenting the material. Subjects in this study is the application of Sudoku system as a medium of learning artificial intelligence to the material tracking system especially on Branch and Bound algorithm. The results of this research is the application of Sudoku settlement system as a medium of learning Artificial Intelligence on material tracking system for Branch and Bound student of Computer Science at the University of Banten Jaya which is based on the results of these trials can be concluded that the application of this study can help in the learning process for students understand the material and can be used as a tool to support teacher learning Artificial Intelligence.**

***Keywords:* game, Sudoku, system algorithm, learning, tracking system, Branch and Bound**

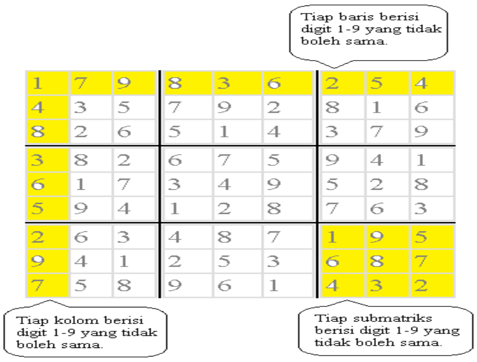
I. INTRODUCTION

Sudoku is a logic puzzle game composed of rows and columns. Sudoku itself is widely circulated in the form of books, newspaper columns, game consoles, and games on the computer. The word sudoku is an abbreviation of the phrase "Su-ji wa dokushin ni kagiru" in Japanese means the number must be single. The aim of this game is to fill the numbers into the available rows and columns, provided that each number entered appears only once in each row and column. However, in some cases sudoku becomes very difficult to solve even if it takes too much time, so an idea arises to build an application to solve sudoku puzzles appropriately and with fast time. As information technology develops, several algorithms emerge that can be applied to solve the sudoku game, one of which is the Branch and Bound algorithm, an algorithmic technique that specifically learns how to reduce the Search Tree to the smallest possible size. As the name implies, this method consists of 2 steps namely: (1). Branch, which means building all the branches of a tree that might lead to a solution. (2). Bound which means counting which nodes are active nodes (E-nodes) and which nodes are dead nodes (D-nodes) using the constraint boundary conditions (constraints).

II. METHOD

A. Sudoku Game Analysis

Sudoku is a logic-based number puzzle game. The word "Sudoku" itself is an abbreviation of a Japanese sentence phrase, "suji wa dokushin ni kagiru", which means the numbers must remain single or unique. Based on the matrix type of order used in this Sudoku program is to use square or square matrices i.e. the matrix with the order or the number of rows equal to the number of columns. Examples of Sudoku sized 9 9 can be seen in the following image.



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

Tiap baris berisi digit 1-9 yang tidak boleh sama.
Tiap kolom berisi digit 1-9 yang tidak boleh sama.
Tiap submatriks berisi digit 1-9 yang tidak boleh sama.

Fig.1. Examples of Sudoku sized

The rules of the game are simple, fill all the matrices to the full, with notes, for each column, row and submatrix measuring 3 3 () may only be filled with numbers 1 to 9 which are each one number. According to the name, then there should not be any same number or digits. This is called the "principle of uniqueness".

The difficulty of Sudoku's game level, apart from being measured by the size of its matrix (), is usually measured from the problem of the formulation of the initial numbers given, of course it will be increasingly difficult. The method used in writing the results of this study is a descriptive method that is the method carried out by analyzing, explain and design a system of available materials.

B. Implementation of Branch and Bound Algorithms

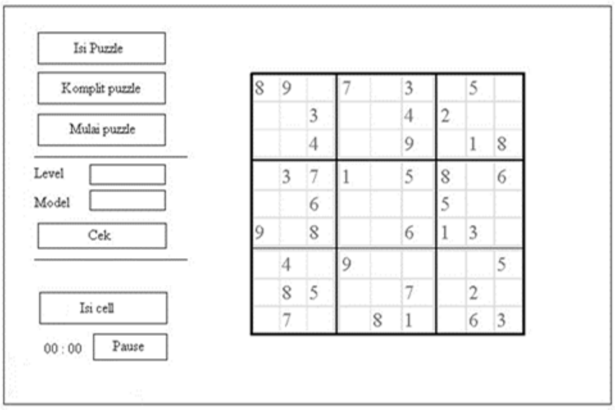
In the problem of formulating Sudoku initial numbers using the Branch and Bound algorithm, it can be divided into two main steps or procedures.

The first step is to fill the grid or submatrix randomly, based on the limiting function. This means that in the process of filling in the numeric boxes, they still adhere to the principle of the uniqueness of Sudoku, that there should not be the same digits in each column, row or submatrix.

The second step, delete the numbers formed one by one, also randomly. If after one number has been deleted and the problem already has a unique solution, then expand to the next node, if not, remove that node then replace the number at its original position, and delete the other numbers (expansion to the next node). In this way, the formulation of the Sudoku problem can prevent many possible solutions. The advantage of the Branch and Boundini method is that there are no digits trapped in an infinite loop, so they can certainly create a solution space.

III. Results and Discussion

In this software design a software will be made using Dreamweaver CS5 software. This software will have menus that can be selected to start, fill, and stop playing time. The design of the software can be seen in Figure 2.



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

Fig.2. The design of the software

The software also includes a button to fill all or one by one blank Sudoku puzzle. Each button has uses, namely:1). "Fill puzzle" button This button is used to direct the user by selecting each empty sub-box sequentially on each sub sudoku matrix to be filled by the user. 2). "Complete puzzle" button This button is used to fill in all the numbers on the sudoku board, if the user presses this button then the software will fill all the empty boxes on the sudoku board.3). "Start puzzle" button. This button is used to start a new game.4). "Level" button This button is used to select the level of difficulty of the game.5). "Model" button This button is used to select the types of sudoku display boards.6). "Check" button This button is used to check the user's error in completing the game.7). "Fill cell" button This button is used to fill empty boxes. .8). "Pause" button This button is used to stop playing time.

Here is a Flowchart diagram of making a Sudoku system:

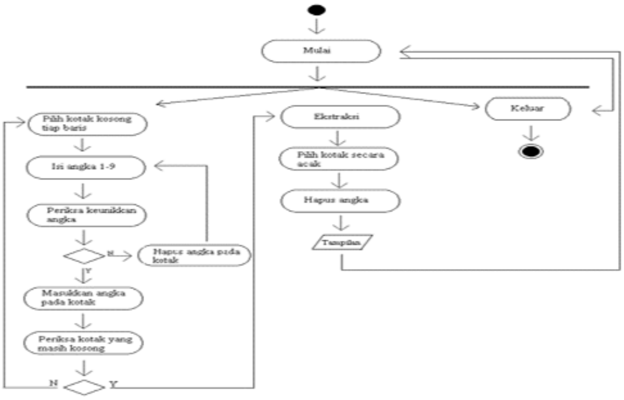
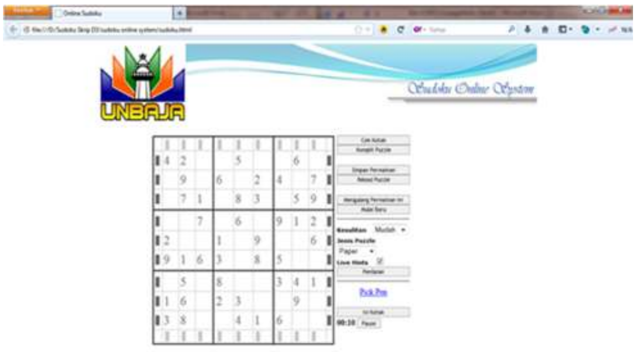


Fig.3. Flowchart diagram of making a Sudoku system

Algorithm of the Program

```
// algorithm forGridKomplit ()
Function forGridKomplit (output: Boolean)
// empty grid initialization process
Dictionary:
Empty column = integer [1. . 9];
// list of empty columns in the entire matrix column
fill in the box = integer [1. . 9]
// list of boxes filled in the line
algorithm:
for (digit=1 to n) (
        integer repetisi = 0;
        clear isi kotak;
        inisialisasi isikotak dengan nilai 1-n
        //cari posisi di baris, untuk tempat meletakkan current digit
        boolean ulangi=true;
        while (ulangi) {
        if (isempty (kolomkosong)) then
        hapus kotak lain yang berisi digit dan
        mulai lagi dari baris=0
        else {
        pilih secara acak dari array kolomKosong
        dan ambil nilai kolom tersebut, kolom=kol;
        if (matriks[baris,kolom] !=empty or
        isExist (digit)) then
                hapus (row, col) dari kolomKosong;
                ulangi=true ;
                if (ulangi) then continue;
                jika
        }//end else
        }//end while
//jika keluar dari loop while dan mulai dari baris awal lagi
        If (ulangi) then continue;
Matriks (baris, kolom) ke dalam isi Kotak
}//endfor loop2
}//endfor loop1
```



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

Fig.4.Starting Game

Starting Game: Previously, it was explained that in filling sudoku numbers using the branch and bound algorithm there are two main procedures. To start a new game in the sudoku program, then first press the "Start Puzzle" button, where the program works by filling in all the numbers on the sudoku board then the program will create a grid of questions to be worked on. Here is the source code for the "Start Puzzle" button:

```
function newGame() {
chosen = Math.floor((Math.random()*10)/(10/puzzles.length));
//'chosen' puzzle should be random
this_puzzle = makeNewOrder(puzzles[chosen]);
show("");
//Clear the display area.
//Clear the existing numbers first
compleated = 0;
clearer();
var last_box_ended_at = 0;
var extra_number_count = 0;
default_number_per_box = $("difficulty").value;
for(var i=0;i<9;i++) {
//Initialisations
var b=0,location_of_fixed_number=0;
var arr_b = new Array();
//Get the numbers for this box from the 'this_puzzle' varaible
var limit = 9;
var this_box = "";
var dot_count = 0;
for(var j=last_box_ended_at; j<last_box_ended_at+limit; j++) {
if(this_puzzle.charAt(j) == ".") {
limit++;
dot_count++;
}
this_box = this_box + this_puzzle.charAt(j);
}
last_box_ended_at = last_box_ended_at + limit;
//Decide how much numbers must appear in this box
number_of_numbers = default_number_per_box - extra_number_count;
extra_number_count = 0;
// 'extra_number_count' is used for reducing the number of populated cells in the next box if
// the current box has more than 4 numbers.
//If the number of dots are more than 4, use it
if (dot_count > number_of_numbers) {
if(rand() > 5) {
extra_number_count = dot_count - number_of_numbers;
}
number_of_numbers = dot_count;
}
//Empty the array.
arr_b = new Array();
//Get the positions in the box and insert the numbers there
for(b=0;b<number_of_numbers;b++) {
location_of_fixed_number = this_box.indexOf(".");
if(location_of_fixed_number + 1) {//If there are dots...
//Remove this dot
this_box = this_box.substring(0,location_of_fixed_number) + this_box.substring(location_of_fixed_number+1,this_box.length);
} else { //No more dots - get some random locations
location_of_fixed_number = uniqueRand(arr_b);
//Give random position for the numbers
location_of_fixed_number--;
//uniqueRand gives 1-9. We need it from 0
arr_b.push(location_of_fixed_number+1);
//Put the number into the don't repeat array
//Get the numbers that should be inserted
insertion_number = this_box.charAt(location_of_fixed_number);
location_of_fixed_number++;
//We need this to start from 1 - not from 0
id = "c" + (i+1) + location_of_fixed_number
insert(id,insertion_number); //Show the numbers
}
}
orginal_game = pos2str(2);
//Save the game before beginning - for restarting the puzzle
//Initialize the timer.
stopTimer();
timer_seconds = 0;
$("timer").innerHTML="00:00";
//Start the timer
timer();
timer_started = 1;
```

System Implementation

The process of charging numbers on the sudoku board can be done by the user or the program itself. When the "Fill Cell" button is pressed, the result number will appear automatically in the box.

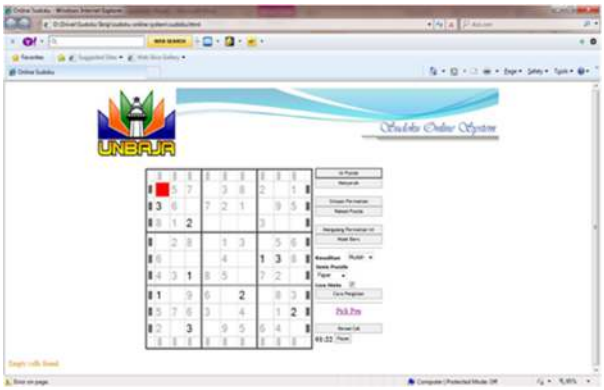
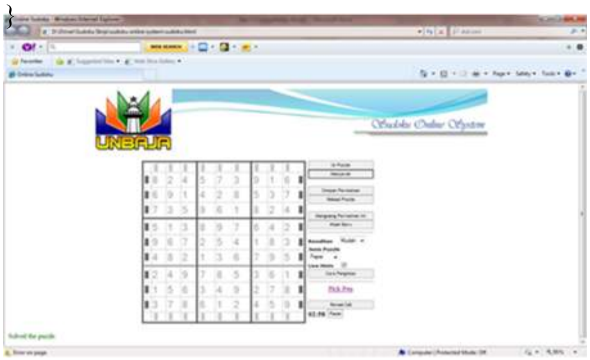


Fig.5.System Implementation

source code untuk tombol “isi cell”:

```
//Show the number of the selected cell.
function revealCell() {
        if(!selected_cell) return;
        var str = this_puzzle.replace(/\./g,"");
//Remove the '.' in the active puzzle.
        var c_xy = selected_cell.id.split("");
//The id will be split - if the id is 'c13' the resulting array will be c,1,3
        var box = c_xy[1]-1;
        var cell= c_xy[2]-1;
        var rest = str.substr(box*9);//Remove the numbers of all the boxes before it.
        var number = rest.charAt(cell); //Now get the cell.
        selected_cell.value = number;
        fadeColor(selected_cell.id,"#ffffff","#000000",10,100,"c");
}
```



```
//Solve the game
function solve() {
show("Finding the solution to the game...");
if(confirm("This will automaticaly solve the puzzle for you.\nAre you sure you want to do this?\n")) {
        str2pos(this_puzzle);
        compleated = 1;
        show("Solved the puzzle.");
        stopTimer(); } }
```

Sudoku Table Inspection Process With Branch and Bound Functions ,The branch and bound method is a search method used in solving sudoku games in this software. In the previous section it was explained that in filling in sudoku numbers using the branch and bound algorithm there are two main procedures. Here is the source code or algorithm to check the sudoku table that has been done using the branch and bound method.

```
function checker() {
        var found = 0;
        show("Cek puzzle...","#a84efa");
        loop:
        for(var a=0;a<9;a++) {
                for(var b=0;b<9;b++) {
                        var ele = cells[a][b];
                var id = "c" + (a+1) + (b+1);
                var value = ele.value
        if(isNaN(value) || value<1 || value>9) {

        if(value == "") {
        show("Empty cells found.","#d78601");
                } else {
        show("Invalid entries found.","#d78601");
                                }
                found = 1;
                ele.style.background = "red";
//Problem Cells found here.
setTimeout("discolorCells('"+id+"')",2000);
                break loop;
                }
                else if(value) {
                if(!checkForUnique(a,b,false)) {
                ele.style.background = "red";
//Repated number found here.
document.getElementById(repeated_at).style.background = "red"; //and here.
        show("Repeated numbers were found.","#d78601");
        setTimeout("discolorCells('"+repeated_at+"','"+id+"')",2000);//Change background back to white after 2 secs
found++;
break loop;
}
} else {
show("Empty cells were found. Please complete the puzzle.","#d78601");
ele.style.background = "red"; //Empty
setTimeout("discolorCells('"+id+"')",2000);
found = 1;
break loop;
                        }
                }
        }
if(!found) {
```

```
if(compleated)
alert("Sorry - you can't win after giving up. But it is solved.");
else
alert("Congratulations - You have completed the puzzle.");
          stopTimer();
          show("Game Over","#000000");
          }
}
```

IV. Conclusions

Conclusions can be drawn based on test cases:

- Branch and bound search algorithms can be implemented and can complete the sudoku game.
- b). Determination of the number of each box by weighing on the main row, column or submatrix that is formed can meet the principle of the uniqueness of sudoku.

Acknowledgments

To discuss further, it can be used with other search algorithms that are simpler and easier to implement in sudoku games. The matrix used on a sudoku board is not only 9 x 9 in size, it can also be used on larger or smaller boards.

References

[1] Kusumadewi, Sri. 2003. Artificial Intelligence (Teknik dan Aplikasinya). Edisi pertama – Yogyakarta; Penerbit Graha Ilmu, 2003.

[2] Suyanto. 2007. Artificial Intelligence (Searching, Reasoning, Planning and Learning). Penerbit Informatika Bandung. 2007.

[3] Suparman. 1991. Mengenal Artificial Intelligence. Edisi pertama – Yogyakarta; Penerbit ANDI OFFSET.

[4] Setiawan, Sandi. 2003. Artificial Intelligence. Penerbit ANDI OFFSET.

[5] Kristanto, Andri, 2004, Kecerdasan Buatan. Yogyakarta: Graha Ilmu.

[6] Santosa, P. Insap, 2004, Interaksi Manusia dan Komputer, Teori dan Prakteknya. Yogyakarta: Penerbit Andi Offset.

[7] Shalahudin, M, 2011, Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur & Berorientasi Objek). : Penerbit Modula

[8] Ladjamudin, Al-Bahra, 2005, Analisis & Desain Sistem Informasi.