ISBN 978-981-11-0008-6
Proceedings of 2016 6th International Workshop on Computer Science and Engineering (WCSE 2016)
Tokyo, Japan, 17-19 June, 2016, pp. 18 -22
doi: 10.18178/wcse.2016.06.004

Sudoku as a Problem Domain in Studying Multiagent Systems

Cecilia E. Nugraheni[+], Luciana Abednego and Albert Kamord

Informatics Dept., Parahyangan Catholic University, Bandung, Indonesia

Abstract. Multiagent System is one of system development paradigms. In particular, this paradigm is useful in the development of a system that is open and distributed, like the internet. With the increasing number of systems that are open and distributed, it is clear that a good understanding on the basic concepts of multiagent systems should be delivered to computer science students. This paper investigates the use of Sudoku as a problem domain in studying Multiagent Systems. Some results from our previous work related to Multiagent Systems are reported, including the specification, verification, design, implementation, testing, and experiments. Some tools developed for supporting teaching/learning activity are also presented.

Keywords: Sudoku, multiagent system, block world problem, artificial intelligence.

1. Introduction

Software systems today, such as the internet, operate in the open and distributed environments. Agent-based technology promised as a new paradigm for conceptualizing, designing, and implementing software systems, particularly for creating software that operates in environments that are distributed and open [1,2]. Therefore, understanding the Multiagents Systems thoroughly is important for Computer Science students.

Unfortunately, in our department, there is no Multiagent Systems course. Multiagent Systems is still a topic of Artificial Intelligence course. We are interested in offering a course that dedicated to Multiagent Systems. For this reason, as preparation, we conducted some research related to Multiagent Systems. The results of the research will be used as materials for this course.

Inspired by the game-based learning approach, we are interested in using a game as the problem domain. We have chosen Sudoku, a famous puzzle game, as a problem domain. Sudoku is a kind of puzzle games. This puzzle consists of a 9×9-cells board and a set of numbers from 1 to 9. The board is divided into nine 3×3-cells sub-boards. The objective of this game is to fill every cell with a number from 1 to 9 such that every row, every column, and every sub-board contains all of the numbers from 1 to 9. Fig. 1 shows an example of Sudoku puzzle and its solution.

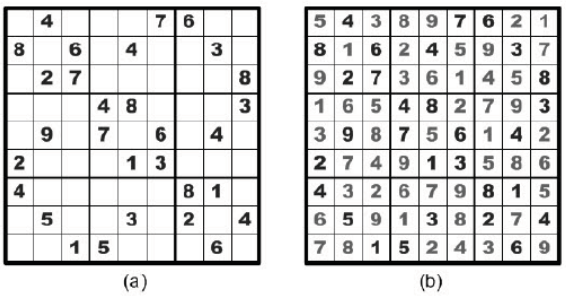

Fig. 1: An example of Sudoku (a) and its solution (b).

[+] Corresponding author. Tel.: +62 813 221 00 411.
E-mail address: cheni@unpar.ac.id.

Solving Sudoku puzzle has been an interesting topic for many researchers. A variety algorithms, such as integer programming, genetic algorithm, SAT, heuristics, etc. have been proposed for solving Sudoku puzzles. Sudoku has been used also as a learning material for many courses as reported by many sites in the internet. One example is the use of Sudoku as problem domain in teaching Artificial Intelligence [3].

This paper presents the results of our work on some aspects of Multiagent Systems [4]-[8], in particular on Sudoku as Multiagent Systems. Our work include almost all the steps of software development, which are specification, design, implementation, verification, testing, and experiments.

The rest of this paper is structured as follows. Section 2 explains how to model Sudoku as Multiagent Systems. Section 3 describes how to solve the Sudoku as Multiagent Systems. Section 4 presents some tools that have been developed. Section 5 concludes the paper.

2. Sudoku as a Multiagent System

In [4] we have introduced a class of Multiagent Systems called Parameterized Multiagent Systems and used Block World Problem for a case study. Block World Problem is one of the simple and famous planning domains in artificial intelligence. Given a set of blocks on a table, two robots are responsible to change the arrangement of the blocks, from initial arrangement to a new different arrangement. At a time only one block that may be moved by a robot. The first robot can only free one block from a top of another and put it on the table, whereas the second robot can only move one block from the table and put it on top of another. Fig. 2. shows an example of Block World Problem.

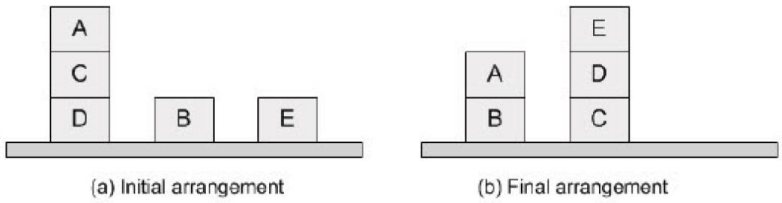

Fig. 2: An example of block world problem.

In [4] we have given the formal specification of Multiagent Systems in TLA notation. The focus of the paper is how to describe or to specify Multiagent Systems and how to verify them. We have restricted on Multiagent Systems that are interleaving, which means that at any time there will never be more than one active agent.

By modifying some settings of Block World Problem, we have shown that Sudoku puzzles can be regarded as a variant of Block World Problems [4,5]. A Sudoku is a Block World Problem with the following settings: There 9×9 blocks are on the table. Each block is labelled with a number from 1 to 9. For every number, there are 9 blocks labelled with the number. There is a special part of the table called board consisting of 3×3 grids. Each grid is called sub-board and consists of 3 × 3 smaller grids called cells. Initially some of the blocks are already placed on those cells and the rest are outside the board. The blocks that are outside the board are organized in nine stacks according to their labels. The objective of the problem is to modify the blocks arrangement from the initial condition to an arrangement satisfying the Sudoku rules. The arrangement modification is done by using some robots with particular capabilities.

3. Solving Sudoku

We now explain the process of solving a Sudoku after modelling the puzzle as a Block World Problem. Regarding a Sudoku as a Block World Problem, the objective of the game as stated in section 2 can be rephrased as changing the arrangement of the blocks into an arrangement that satisfies the Sudoku rules. Searching the solution is done by controlling the work of the robots.

Four algorithms or models for finding solutions have been proposed [5,6]. There are two differences between the first model with other models, which are the number of robots used and the ability or task of each robot.

Very similar to the original Block World Problem, the first model uses two robots with different capabilities or tasks. The first robot's task is to take a block outside the board and move the block to the cell on the board without violating Sudoku's rules. The first robot will continue working as long as it can find a block outside the board to be put to the board. If at any time the first robot could not find any block, the turn is given to the second robot. The second robot's task is to select a block that is already on the board, pick the block, and return the block to its corresponding stack outside the board. The turn then is given back to the first robot. This is done repeatedly until all blocks are on the board or no more blocks outside the board can be moved to the board. Given a valid Sudoku puzzle, the first model will successfully find the solution.

Unlike the first model, the other models use nine robots and use fixed-point principle, instead of backtracking principle. Every robot has a number between 1 and 9 and is called by the number. The solution searching process is done by making iterations until a termination condition is reached. At any iteration, every robot *i* tries to move a block with label *i* from the stack outside the board to a valid cell on the board. The robots work by turns. After all robots get their turns, it will be decided whether a new iteration should be made or not. Differs from the first model, there is no guarantee that the solution for a Sudoku puzzle will be found.

In deciding whether a cell is a correct or a valid cell for a particular block, there are some rules that must be followed by the robots. There are two types of rules: the general rules and the specific rules. All models apply the general rules. A robot *i* will call a cell at position (*x*,*y*) on the board satisfies the general rules if the following conditions hold: 1) It is empty, 2) The row *x* does not contain any blocks with label *i*, 3) The column *y* does not contain any blocks with label *i*, and 4) The sub-board where (*x*,*y*) is located does not contain any blocks with label *i*.

Every model adds two more rules that are specific for each model. The specific rules for every model are given in Table. 1. We may say that those rules reflect the intelligence of the robots. Fig. 3(a) and Fig. 3(b) shows a valid condition for a robot 4 of model 3 and a valid condition for robot 5 of model 4, respectively.

Table 1: Specific Rules

Model	**Specific Rules**
1	a. For every other row *r* in the same sub-board there is a block with label *i* b. For every other column *c* in the same sub-board there is a block with label *i*
2	a. For every other row r in the same sub-board there is a block with label *i* or the cell at the position (*r*, *y*) is not empty. b. For every other column *c* in the same sub-board there is a block with number *i* or the cell at the position (*x*, *c*) is not empty.
3	a. For every other row *r* in the same sub-board there is a block with label *i* or the cell at the position (*r*, *y*) is not empty or all cells in row *r* in the same sub-board with the cell (*x*, *y*) are not empty. b. For every other column *c* in the same sub-board there is a block with label *i* or the cell at the position (*x*,*c*) is not empty or all cells in column c in the same sub-board with the cell (*x*,*y*) are not empty.

4. Tools

In [6] we reported the development of a prototype of Sudoku Solver implementing all the models. In our work, a Multiagent System is defined as an interleaving system [5,6], which means that at any time there is only maximum one agent that active. As a consequence of that definition, the robots should work in turns, one robot at a time. To control the activities of the robots, a token-based scheduling is used. The robot that has the token has the right to perform its task. After completing its activity, the robot gives the token to another robot. There are two algorithms or scenarios for choosing which robot that should be given the token: ring-based or random. Communication between two robots, in this case is the passing token from one robot to the other robot, is done indirectly, i.e. via a special agent called controller. The controller is responsible on the robot scheduling. The architecture of the solver is given in Fig. 4.

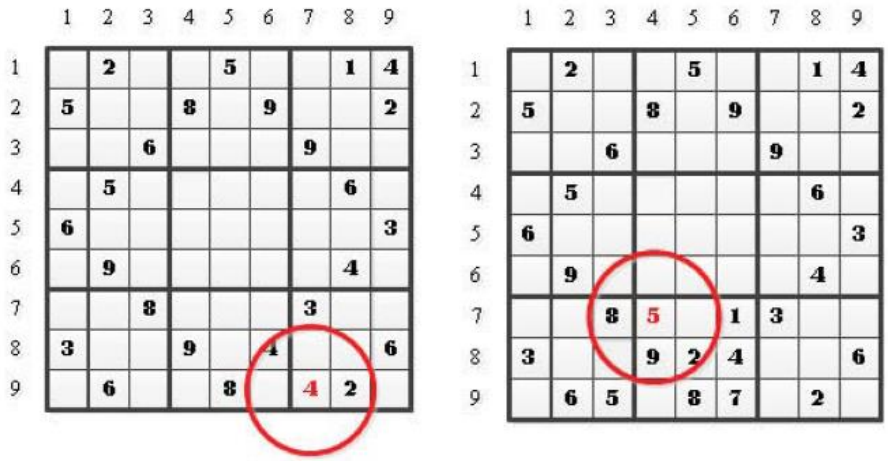

Fig. 3: An example of a valid position for model 3 (a) and an example of a valid position for model 4 (b).

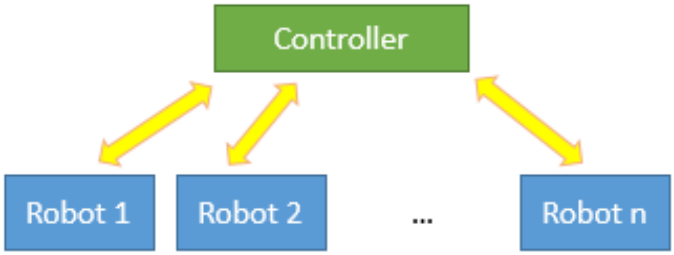

Fig. 4: Architecture of the solver.

Some experiments have been conducted using this prototype. The performance of each model can be concluded based on the experimental results. We have also studied the effect of agent scheduling scenarios on the performance of every model [7]. The prototype has been extended and improved so that it can be used for supporting the teaching/learning activity. This solver is not only can be used to solve the puzzle but also display the solution searching process, step by step. The users can choose the puzzle to be solved, the model, and the scheduling scenario. In order to give better visualization a 3D version of the solver has also been developed. In 3D version, the robots are represented as helipads. Fig. 5 and Fig. 6 presents the screenshot of 2D version and 3D version of Sudoku Solver, respectively.

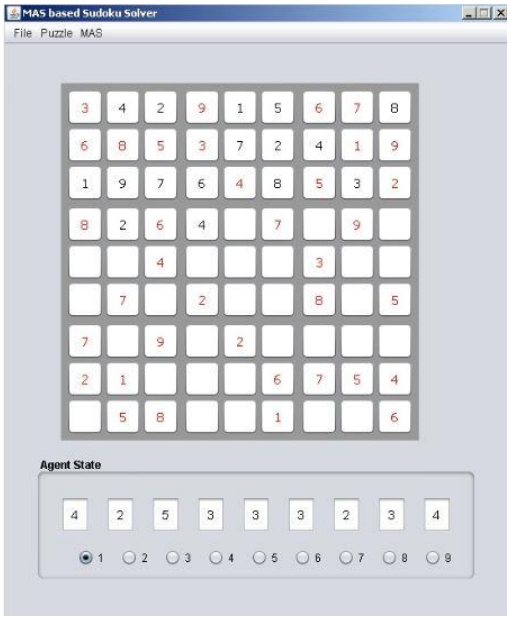

Fig. 5: 2D Block world problem Sudoku solver.

Fig. 6: 3D Block world problem Sudoku solver.

5. Conclusion

We have presented the results of our research that related to Multiagent Systems, especially about Sudoku as Multiagent Systems. Our work cover a range of issues related to software development, namely specification, verification, design, implementation, testing, and experiment. The results of these researches will be used as materials for Multiagent Systems course.

Until now we only considered some aspects of Multiagent Systems, which are the architecture, agent communication and coordination, and scheduling scenarios. Moreover, Multiagent Systems we have considered is only restricted to the interleaving Multiagent Systems. In the future we plan to consider parallel Multiagents Systems and more aspects of Multiagent Systems.

6. References

[1] K.P. Sycara. Multiagent Systems. AI Magazine, Vol 19, No 2, pp. 79-92, Summer 1998.

[2] Woolridge. An Introduction to MultiAgent Systems, 2nd Edition. Wiley. 2009.

[3] Jeffrey O. Pfaffmann and William J. Collins. Teaching Artificial Intelligence Across the Computer Science Curriculum using Sudoku as a Problem Domain. https://www.aaai.org/Papers/FLAIRS/2007/Flairs07-066.pdf

[4] Cecilia E. Nugraheni. Formal Verification of Parameterized Multi-Agent Systems using Predicate Diagrams*. *Proc. of Computation Tools* 2011, pp. 19-24.

[5] C.E. Nugraheni and L. Abednego. Modelling Sudoku Puzzles as Block-world Problems. International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:7, No:8, 2013.

[6] L. Abednego and C.E. Nugraheni. A Block World Problem Based Sudoku Solver. International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:8, No:8, 2014.

[7] C.E. Nugraheni and L. Abednego. A Study on Agent Scheduling Scenarios in Multi-Agent Systems. *Journal of Theoretical and Applied Information Technology* (JATIT) 76 (2). 2015.

[8] Albert Kamord. 3D Visualization of Block World Problem Based Sudoku Solver. Final Project. Informatics Dept. Parahyangan Catholic University. 2016.