УДК 519.83 DOI: https://doi.org/10.32515/2664-262X.2019.2(33).190-201

Yuriy Parkhomenko, Assoc. Prof., PhD. tech. sci., **Mykhailo Parkhomenko,** Assoc. Prof., **Ludmila Rybakova**, Assoc. Prof., **Andriy Bokiy**
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine
e-mail: parhomenkoym@ukr.net

Analysis of the Methods for Solving Game Puzzles such as «Flip-Flop»

There is a variety of popular puzzles having a goal of reducing an arbitrary binary matrix to either all "0" or "1" matrix. In this paper we study methods for solving "Flip-Flop" like puzzles of dimensions 3x3, 3x4, 4x4 applying tools of logical analysis of situations, combinatorics and discrete mathematics. We found that applying the method of sequential analysis of each combination that works well for 3x3 matrices is cumbersome and inefficient for matrices of 4x4 and higher dimensionalities. Therefore, we discovered and analyzed algorithms named trait selection method, stream method and snake method which work better

We concluded that in order to find an optimized solution it is helpful to check if each current combination matches one of the pre-final ones, or to swap «0»s with «1»s and vise versa.

computer puzzle games, logical situation analysis, combinatorics, discrete mathematics, trait method, stream method, snake method

Ю. М. Пархоменко, доц., канд. техн. наук**, М. Д. Пархоменко,** доц., **Л.Рибакова,** доц., **А. Р. Бокий**
Центральноукраинский национальный технический университет, г.Кропивницкий, Украина

Исследование методов решения игр- головоломок типа «Flip-Flop»

Существует множество популярных головоломок, целью которых является сокращение произвольной двоичной матрицы до любой матрицы «0» или «1». В этой статье мы изучаем методы решения компьютерных игр-головоломок типа «Flip-Flop», с матрицами размерностей 3x3, 3x4, 4x4, с использованием инструментов логического анализа ситуаций, комбинаторики и дискретной математики. Мы определили, что применение метода последовательного анализа каждой комбинации, который хорошо работает для матриц 3x3, является громоздким и неэффективным для матриц 4x4 и более высоких размерностей. Поэтому мы нашли и проанализировали алгоритмы, названные методом выбора признаков, методом «ручейка» и методом «змейки» , которые работают лучше.

Мы пришли к выводу, что для нахождения оптимального решения полезно проверить, соответствует ли каждая текущая комбинация одной из предконечных, и своевременно вмешаться в процесс, либо предварительно обработать содержимое исходной матрицы, чтобы оптимизировать количество ячеек с кодами «0» или «1».

компьютерные игры-головоломки, логический анализ ситуаций, комбинаторика, дискретная математика, метод выделения признаков, метод «ручейка», метод «змейки»

Statement of the problem. Mental games like checkers and chess appeared in ancient times as a mental leisure activity. With the rise of educational level they have become more sophisticated, and the circle of their fans has been growing. Gradually, games became not only a type of leisure, but also an object of scientific research. At the turn of the XIX and XX centuries, the Game Theory emerged, and since then it has been constantly evolving finding applications in economics, sociology, biology, industry, military and other fields of human activity. The state of the art in decision making relies on simulation of game situations, behavioral analysis and optimization methods in order to find the best strategy.

Among the growing number of computer games, the so-called puzzle games are of great interest to young people and adults. Solving puzzles requires careful analysis of situations and finding logical and mathematical patterns to determine the right sequence of actions. Some puzzles stimulate theoretical and practical advances. For example, the number of different states of Rubik's cube reaches 43 quintillions of combinations. At the same time, it is known that applying so-called «algorithm of God» allows solving the puzzle in no more than 20 steps from any state. Rubik's cube became not just a toy, but also an object of research for mathematicians and engineers. Even today such puzzles as «Crossbones-Nulks» and Game of Fifteen that people have played for generations have not lost their popularity. At the same time a lot of newcomers, like «Threes!», «2048», «Sudoku» and others have appeared.

The «Flip-Flop» puzzle, that can be found on the internet in different variations caught our attention. The essence of the game is as follows. At the beginning of the game matrix cells 4×4 are filled with «0»s or «1»s randomly. When a matrix cell gets activated, the values of the entire row and column at the intersection of it are changed to the opposite codes, that is, «0» becomes «1» and vice versa. The objective of the game is to bring all matrix cells either to «0»s or «1»s (depending on the given goal) in a finite number of steps. There are several modifications of the game that differ by cell content; instead of «0» and «1»s game designers use flowers, berries ("Fruity Flip Flop") and the like. There are also rule differences when the activation of a cell changes the values of the adjacent cells as opposed to changing the entire row and column.

This paper explores the original version of the game, i.e. when all values of adjacent row and column get changed.

Analysis of recent researches and publications. As the Flip-Flop experience shows, achieving the goal of arriving to the matrix with all 0s (or 1s) from an arbitrary initial combination in a finite number of steps is not an easy problem. It is hard to foresee what combination occurs in two or three steps ahead even in matrix 4×4 because seven out of sixteen cells change their values at each step. In order to solve the puzzle we have developed and analyzed an algorithm that allows us to reach the solution of forming either all-0s or all-1s matrix in minimal number of steps for any **m x n** matrix.

Studying references like Game Theory [1, 2], Discrete Math [3], Artificial Intelligence [4], Combinatorics [5], and experiences with similar puzzles at braingames.ru convinced us to use simulation, formal logic and combinatorics.

Statement of the objective. The aim of the article is to determine the methods for solving the problem of reducing the arbitrary combination of matrix codes, with a minimum number of steps, to one "0" or "1" and to build the algorithms that will provide this process.

The main material.

Terminology. We use the following notation:

- binary numbers 0 or 1 are used to fill matrix cells;

- *(i=1,2,3…m)* denotes <i-th> row of the matrix;

- *(j=1,2,3…m)* denotes <j-th> column of the matrix;

- *(i,j)* denotes i,j matrix cell;

- instead of binary code (of a row) hexadecimal code may be used. Therefore instead of representing a matrix as a collection of cells we may represent it as a list of (hex) codes of its rows;

- activated cell is denoted by either **0.** or **1.**;

- “zero” matrix - matrix consisting of 0s;

- “unit” matrix - matrix consisting of 1s;

- when a cell gets activated, the content of each cells in its row and column gets inverted («0» becomes «1» and vise versa).

- an arbitrary combination of the matrix at the beginning of the game is generated programmatically using a binary random number generator;

- since algorithms for the formation of «zero» and «unit» matrices are similar, we focus on obtaining «zero» matrix.

Analysis of the "Flip-Flop" Solution for 3x3 Matrix.

Before proceeding with the study of a solution for matrices of 3×3 size, we will analyze forming a «zero» 2×2 matrix. In doing so, we will simulate all possible initial combinations and form the final «zero» matrix from them by analyzing the current configuration and determining the strategy for the next step (Fig. 1).

$$1)\begin{matrix}\mathbf{1.} & 0\\ 0 & 1\end{matrix} \Rightarrow \begin{matrix}0 & 1\\ 1 & \mathbf{1.}\end{matrix} \Rightarrow \begin{matrix}0 & 0\\ 0 & 0\end{matrix} \quad 2)\begin{matrix}1 & 1\\ 0 & \mathbf{0.}\end{matrix} \Rightarrow \begin{matrix}1 & 0\\ \mathbf{1.} & 1\end{matrix} \Rightarrow \begin{matrix}0 & 0\\ 0 & 0\end{matrix} \quad 3)\begin{matrix}\mathbf{1.} & 1\\ 1 & 1\end{matrix} \Rightarrow \begin{matrix}0 & \mathbf{0.}\\ 0 & 1\end{matrix} \Rightarrow \begin{matrix}1 & 1\\ 0 & \mathbf{0.}\end{matrix} \Rightarrow \begin{matrix}1 & 0\\ \mathbf{1.} & 1\end{matrix} \Rightarrow \begin{matrix}0 & 0\\ 0 & 0\end{matrix}$$

$$4)\begin{matrix}1 & \mathbf{0.}\\ 0 & 0\end{matrix} \Rightarrow \begin{matrix}0 & 1\\ \mathbf{0.} & 1\end{matrix} \Rightarrow \begin{matrix}\mathbf{1.} & 1\\ 1 & 0\end{matrix} \Rightarrow \begin{matrix}0 & 0\\ 0 & 0\end{matrix} \qquad 5)\begin{matrix}\mathbf{1.} & 0\\ 0 & 0\end{matrix} \Rightarrow \begin{matrix}0 & \mathbf{1.}\\ 1 & 0\end{matrix} \Rightarrow \begin{matrix}1 & 0\\ \mathbf{1.} & 1\end{matrix} \Rightarrow \begin{matrix}0 & 0\\ 0 & 0\end{matrix}$$

Figure 1 – The sequence of formation of «zero» matrix with different initial combinations
Source: author's development

The analysis of the obtained solutions shows:

- in order to reach the «zero» combination, it necessary to arrive to the pre-final combination in which activation of the critical cell results in forming the «zero» matrix;

- all possible pre-final combinations are formed at the intersection of matrix cells, so their number corresponds to the number of the matrix cells (for the matrix $2 \times 2 = 4$);

- the search strategy is to find one of the four final combinations, which makes it easier to solve the problem;

- examples 4 and 5 show that starting from the same input, the sequence of steps for reaching one of the final combinations may be different but the result is the same.

It follows that for the matrix 3×3 there are 9 pre-final combinations. All of them can be expressed number triplets: 7-4-4; 7-1-1; 4-4-7; 1-1-7; 7-2-2; 2-2-7; 4-7-4; 1-7-1; 2-7-2.

1			2			3			4			5			6			7			8			9		
1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	0	0	0	0	1	0	1	0
1	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1	0	0	0	0	1	0	1	0

Figure 2 – Table of final combinations for matrix 3x3
Source: author's development

Searching for a long time for ways to reach one of these combinations did not give positive results. At the same time, the following has been established: the modulo 2 sum of columns and rows of all pre-final combinations is equal (sum $|2| = 7_2$); if the current combination is even (sum$|2| = 0$) then the next combination is odd, so the pre-final combination must necessarily be even; when activating an arbitrary cell, for example (2,3), in the current combination (let's call it combination A) and then the same cell in the resulting combination, then the first combination (i.e. combination A) will appear again; if in the odd combination (2-5-0) sequentially activate cells (1,1) and (1,2), then two new even combinations will be formed, if in the first of them (5-1-4) activate cell (1,2), and in the second (5-7-2) activate cell (1,1), we will also obtain new, but absolutely identical combinations (2-3-6) (Fig. 3); this partly explains why the process may be infinite, that is the sequence of actions results in the same combinations making it cyclical.

Sum |2|	1	1	1		0	0	0		1	1	1	Sum |2|	1	1	1		0	0	0		1	1	1
1	**0.**	1	0	⇨	1	**0.**	1	⇨	0	1	0	2	0	**1.**	0	⇨	**1.**	0	1	⇨	0	1	0
	1	0	1		0	0	1		0	1	1		1	0	1		1	1	1		0	1	1
	0	0	0		1	0	0		1	1	0		0	0	0		0	1	0		1	1	0

Figure 3 – Repeatability of combinations

Source: author's development

Let's get back to the search for combinations which lead to forming the pre-final and final matrices. The analysis has revealed the following:

- activation of the cells with the code «0» in each of the 9 pre-final combinations (Figure 2) result in 6 new types of matrices (Fig. 4, Examples 1-3) having 6 copies of each (4×9=36), which differ from each other by rows with codes 3 5 6 (3-6-5, 6-5-3, 5-3-6, 6-3-5, 3-5-6, 5-6-3);
- activations of the cells with code «1» in each of the 9 pre-final combinations (Fig. 2) result in 18 new matrix types (Fig. 4, Examples 4-9), which differ from each other by rows with codes 0 3 3, 0 5 5 and 0 6 6;
- when performing reverse actions - activation of cells with codes «1» in front of the pre-final matrices with rows having codes 3 5 6 (Fig. 5, examples 1-3) different combinations of the pre-final matrices are formed (Fig. 2);
- when performing reverse actions - activation of cells with codes «1» in front of the pre-final matrices with the rows having codes 0 3 3, 0 5 5 and 0 6 6 (Fig. 5, examples 4-9) different combinations of the pre-final matrices are also formed (Fig. 5, examples 4-9);
- activations of the cells with code «0» in front of the pre-final matrices with rows having codes 3 5 6 result in forming the pre - pre final matrix with combinations of rows having codes 1 2 4 (Fig. 6, Examples 1-3);
- activations of cells with code «1» in front of the pre-final matrices with row codes 0 3 3, 0 5 5 and 0 6 6 result in forming the pre-pre pre-final matrix with row codes 1 2 4 (Fig. 6, examples 4-9);
- when performing reverse actions - activation of cells with codes «1» in front of the pre-pre pre-final matrices with row codes 1 2 4 the pre pre-final matrix is formed with row codes 3 5 6 (Fig. 7, Examples 1-3);
- when performing reverse actions - activation of cells with codes «0» in front of the pre pre pre-final matrices with row codes 1 2 4 the pre pre-final matrix is formed with row codes 0 3 3, 0 5 5 and 0 6 6 (Fig. 7, examples 4-9).

Matrix Forming with row codes 3 5 6 by activation of cells with code «0»																													
1.	(4)	1	**0.**	0	⇨	(3)	0	1	1	2.	(4)	1	0	**0.**	⇨	(3)	0	1	1	3.	(2)	0	1	0	⇨	(6)	1	1	0
	(4)	1	0	0		(6)	1	1	0		(4)	1	0	0		(5)	1	0	1		(7)	1	1	1		(3)	0	1	1
	(7)	1	1	1		(5)	1	0	1		(7)	1	1	1		(6)	1	1	0		(2)	**0.**	1	0		(5)	1	0	1
Matrix Forming with row codes 0 3 3, 0 5 5 та 0 6 6,																													
4.	(4)	**1.**	0	0	⇨	(3)	0	1	1	5.	(4)	1	0	0	⇨	(0)	0	0	0	6.	(4)	1	0	0	⇨	(6)	1	1	0
	(4)	1	0	0		(0)	0	0	0		(4)	**1.**	0	0		(3)	0	1	1		(4)	1	0	0		(6)	1	1	0
	(7)	1	1	1		(3)	0	1	1		(7)	1	1	1		(3)	0	1	1		(7)	1	**1.**	1		(0)	0	0	0
by activation of cells of pre-final matrices with code «1»																													
7.	(7)	1	1	1	⇨	(5)	1	0	1	8.	(7)	1	1	1	⇨	(5)	1	0	1	9.	(7)	**1.**	1	1	⇨	(0)	0	0	0
	(2)	0	**1.**	0		(5)	1	0	1		(2)	0	1	0		(0)	0	0	0		(2)	0	1	0		(6)	1	1	0
	(2)	0	1	0		(0)	0	0	0		(2)	0	**1.**	0		(5)	1	0	1		(2)	0	1.	0		(6)	1	1	0

Figure 4 – Examples of forming matrices with row codes 3 5 6, 0 3 3, 0 5 5, and 0 6 6

Source: author's development

Forming of the Pre-final Matrix with row codes 3 5 6 by activation of cells with code«1»																													
1.	(3)	0	1	**1.**	⇨	(4)	1	0	0	2.	(3)	0	1	1	⇨	(7)	1	1	1	3.	(6)	1	1	0	⇨	(7)	1	1	1
	(6)	1	1	0		(7)	1	1	1		(5)	**1.**	0	1		(2)	0	1	0		(3)	0	1	**1.**		(4)	1	0	0
	(5)	1	0	1		(4)	1	0	0		(6)	1	1	0		(2)	0	1	0		(5)	1	0	1		(4)	1	0	0
Forming of the Pre-final Matrix with row codes 0 3 3, 0 5 5 and 0 6 6,																													
4.	(3)	**0.**	1	1	⇨	(4)	1	0	0	5.	(0)	0	**0.**	0	⇨	(7)	1	1	1	6.	(6)	1	1	**0.**	⇨	(1)	0	0	1
	(0)	0	0	0		(4)	1	0	0		(3)	0	1	1		(1)	0	0	1		(6)	1	1	0		(7)	1	1	1
	(3)	0	1	1		(7)	1	1	1		(3)	0	1	1		(1)	0	0	1		(0)	0	0	0		(1)	0	0	1
by activation of cells with code «0»																													
7.	(5)	1	**0.**	1	⇨	(2)	0	1	0	8.	(5)	1	**0.**	1	⇨	(2)	0	1	0	9.	(6)	1	1	0	⇨	7)	1	1	1
	(5)	1	0	1		(7)	1	1	1		(0)	0	0	0		(2)	0	1	0		(0)	0	0	0		(1)	0	0	1
	(0)	0	0	0		(2)	0	1	0		(5)	1	0	1		(7)	1	1	1		(6)	1	1	**0.**		(1)	0	0	1

Figure 5 – Examples of the formation of final matrices from before the finite
Source: author's development

Forming of the matrices with row codes 1 2 4 by activation of cells having code «0» in the matrices with row codes3 5 6,																													
1.	(3)	**0.**	1	1	⇨	(4)	1	0	0	2.	(3)	0	1	1	⇨	(1)	0	0	1	3.	(6)	1	1	0	⇨	(4)	1	0	0
	(6)	1	1	0		(2)	0	1	0		(5)	1	**0.**	1		(2)	0	1	0		(3)	0	1	1		(1)	0	0	1
	(5)	1	0	1		(1)	0	0	1		(6)	1	1	0		(4)	1	0	0		(5)	1	**0.**	1		(2)	0	1	0
Forming of the matrices with row codes 1 2 4 by activation of cells having code «1» in																													
4.	(3)	0	**1.**	1	⇨	(4)	1	0	0	5.	(0)	0	0	0	⇨	(2)	0	1	0	6.	(6)	1	1	0	⇨	(2)	0	1	0
	(0)	0	0	0		(2)	0	1	0		(3)	0	**1.**	1		(4)	1	0	0		(6)	**1.**	1	0		(1)	0	0	1
	(3)	0	1	1		(1)	0	0	1		(3)	0	1	1		(1)	0	0	1		(0)	0	0	0		(4)	1	0	0
the matrices with row codes 0 3 3, 0 5 5 та 0 6 6.																													
7.	(5)	**1.**	0	1	⇨	(2)	0	1	0	8.	(5)	1	0	1	⇨	(1)	0	0	1	9.	(6)	1	1	0	⇨	(4)	1	0	0
	(5)	1	0	1		(1)	0	0	1		(0)	0	0	0		(4)	1	0	0		(0)	0	0	0		(2)	0	1	0
	(0)	0	0	0		(4)	1	0	0		(5)	**1.**	0	1		(2)	0	1	0		(6)	1	**1.**	0		(1)	0	0	1

Figure 6 – Examples of forming matrices with 1, 2, 4 row codes from the pre pre-final matrices
Source: author's development

Thus, we reached a cycle: in order to output «zero» matrix, it is necessary to get to one of 9 pre-final matrices, however we can get to any of them only from the pre pre-final matrix with row codes 3 5 6, 0 3 3, 0 5 5 and 0 6 6; the pre pre-final matrices are formed from matrices with row codes 1 2 4, and the latter are themselves formed from the pre pre-final matrices with row codes 3 5 6, 0 3 3, 0 5 5 and 0 6 6. Hence, we can conclude that the solution to the problem for 3×3 matrix only exists if the row codes generated by the random number generator match one of the 30 possible codes 1 2 4, 3 5 6, 0 3 3, 0 5 5, and 0 6 6, in all other cases the problem has no solution.

Forming of the matrices with row codes 3 5 6 by activation of cells having code «1» in the matrices with row codes 1 2 4.																													
1.	(4)	**1.**	0	0	⇨	(3)	0	1	1	2.	(1)	0	0	1	⇨	(3)	0	1	1	3.	(4)	1	0	0	⇨	(6)	1	1	0
	(2)	0	1	0		(6)	1	1	0		(2)	0	**1.**	0		(5)	1	0	1		(1)	0	0	1		(3)	0	1	1
	(1)	0	0	1		(5)	1	0	1		(4)	1	0	0		(6)	1	1	0		(2)	0	**1.**	0		(5)	1	0	1
Forming of the matrices with row codes 0 3 3, 0 5 5 and 0 6 6,																													
4.	(4)	1	**0.**	0	⇨	(3)	0	1	1	5.	(4)	1	0	0	⇨	(0)	0	0	0	6.	(2)	0	1	0	⇨	(6)	1	1	0
	(2)	0	1	0		(0)	0	0	0		(4)	**1.**	0	0		(3)	0	1	1		(1)	**0.**	0	1		(6)	1	1	0
	(1)	0	0	1		(3)	0	1	1		(7)	1	1	1		(3)	0	1	1		(4)	1	0	0		(0)	0	0	0
by activation of cells having code «0» in the matrices with row codes 1 2 4.																													
7.	(2)	**0.**	1	0	⇨	(5)	1	0	1	8.	(1)	0	0	1	⇨	(5)	1	0	1	9.	(4)	1	0	0	⇨	(6)	1	1	0
	(1)	0	0	1		(5)	1	0	1		(4)	1	0	0		(0)	0	0	0		(2)	0	1	0		(0)	0	0	0
	(4)	1	0	0		(0)	0	0	0		(2)	**0.**	1	0		(5)	1	0	1		(1)	0	**0.**	1		(6)	1	1	0

Figure 7 – Examples of forming the pre pre-final matrices from matrices with row codes 1 2 4

Source: author's development

Search for algorithms for solving the "Flip-Flop" problem for the 4x4 matrix. Following similar steps as in the previous section, we first determine the pre-final combinations. Their number corresponds to the number of cells in the matrix $N = m \times n = 4 \times 4 = 16$, and the configurations correspond to cells at the intersection of rows and columns with either codes «1» or «0». For each of the 16 pre-final combinations, you can create 15 pre pre-final ones, the total number of which is equal $15 \times 16 = 300$. It is hard to keep them in memory for a regular person. Getting to the pre-final or the pre pre-final combinations by analyzing the current combinations and determining the optimal solution for each subsequent step proved to be quite difficult as at each step the current matrix codes are changed to the opposite in 7 cells out of 16, so tracing the result for 2-3 steps is even harder. Therefore we abandoned attempts to find the best next step at each combination focusing instead at a search for an algorithm capable to find a solution in finite number of steps. The first algorithm was inspired by the following analogy: a reader wants to finish a book quickly and she cares only about her favorite character, so she opens the first page, finds a section talking about her favorite character, reads it and moves to the next page.

Method of sign allocations.. Since the number of steps to obtain «zero» matrix in this method varies and can exceed two dozen, in order to make the explanation less cumbersome and more visible, we are going to expand the whole sequence of matrix combinations in the long rows (Fig. 9), and use the tables only for demonstration of the first few steps (Fig. 8).

1-st step						2-nd step					3-d step					4-th step					5-th step					6-th step				
(8)	**1.**	0	0	0		0	**1.**	1	1		1	0	0	0		0	0	0	0		0	0	0	1		0	1	0	1	
(2)	0	0	1	0	⇨	1	0	1	0	⇨	**1.**	1	1	0	⇨	0	0	0	**1.**	⇨	1	1	1	0	⇨	1	0	1	0	⇨
(1)	0	0	0	1		1	0	0	1		1	1	0	1		0	1	0	1		0	**1.**	0	0		1	0	**1.**	1	
(7)	0	1	1	1		1	1	1	1		1	0	1	1		0	0	1	1		0	0	1	0		0	1	1	0	

Figure 8 – Sequence of actions according to the method of signs

Source: author's development

The algorithm for implementing the sequence of actions by the method of signs is as follows. If you want to get to the «zero» combination, you need to activate the necessary cells with code «1», in the «unit» combination then you need to activate cells with code «0» . Your chosen code serves as a sign. Consider an option to getting to «zero» combination (Fig. 8, 9).

The first step begins with activation of the cell with the code «1» in the input matrix (marked in bold type with a dot).

Note: You can start with an arbitrary cell with code «1» and at an arbitrary step of the game.

When a new combination is detected, going clockwise we skip all cells with code «0» until we detect the first cell with code «1» in the same or next row, but farther from the active cell in the previous combination, and activate it.

We make the next steps following the same rule. If the last activated cell with code «1» was in the last (i.e. fourth) row, and after it in the same row of the new combination there are no cells with code «1» then going clockwise we identify the first cell with code «1» in the first line of this combinations and activate it (see Figure 9 transitions from 7 to 8, from 14 to 15 and from 23 to 24 steps). We continue to perform similar steps until «zero» combination appears. This method is not optimal, but guarantees finding a solution to the problem.

№i/o		1-st row					2-nd row					3-d row					4-th row				16-th row code				
1		**1.**	0	0	0		0	0	1	0		0	0	0	1		0	1	1	1		8	2	1	7
2		0	**1.**	1	1		1	0	1	0		1	0	0	1		1	1	1	1		7	A	9	F
3		1	0	0	0		**1.**	1	1	0		1	1	0	1		1	0	1	1		8	E	D	B
4		0	0	0	0		0	0	0	**1.**		0	1	0	1		0	0	1	1		0	1	5	3
5		0	0	0	1		1	1	1	0		0	**1.**	0	0		0	0	1	0		1	E	4	2
6		0	1	0	1		1	0	1	0		1	0	**1.**	1		0	1	1	0		5	A	B	6
7		0	1	1	1		1	0	0	0		0	1	0	0		0	**1.**	0	0		7	8	4	4
8		0	0	1	1		1	1	0	0		0	0	0	0		1	0	**1.**	1		3	C	0	B
9		0	0	0	**1.**		1	1	1	0		0	0	1	0		0	1	0	0		1	E	2	4
10		1	1	1	0		**1.**	1	1	1		0	0	1	1		0	1	0	1		E	F	3	5
11		0	1	1	0		0	0	0	0		**1.**	0	1	1		1	1	0	1		6	0	B	D
12		1	1	1	1		1	0	0	0		0	**1.**	0	0		0	1	0	1		F	8	4	5
13		1	0	1	0		1	1	0	0		0	1	**1.**	0		1	1	0	0		A	C	6	C
14		1	0	0	0		1	1	1	0		0	1	0	0		0	0	**1.**	1		8	E	4	3
15		**1.**	0	1	0		1	1	0	0		0	1	1	0		1	1	0	0		A	C	6	C
16		0	**1.**	0	1		0	1	0	0		1	1	1	0		0	1	0	0		5	4	E	4
17		1	0	**1.**	0		0	0	0	0		1	0	1	0		0	0	0	0		A	0	A	0
18		0	1	0	**1.**		0	0	1	0		1	0	0	0		0	0	1	0		5	2	8	2
19		1	0	1	0		0	0	**1.**	1		1	0	0	1		0	0	1	1		A	3	9	3
20		1	0	0	0		1	1	0	0		**1.**	0	1	1		0	0	0	1		8	C	B	1
21		0	0	0	0		0	1	0	0		0	**1.**	0	0		1	0	0	1		0	4	4	9
22		0	1	0	0		0	0	0	0		1	0	**1.**	1		1	1	0	1		4	0	B	D
23		0	1	1	0		0	0	1	0		0	1	0	0		**1.**	1	1	1		6	2	4	F
24		**1.**	1	1	0		1	0	1	0		1	1	0	0		0	0	0	0		E	A	C	0
25		0	0	0	**1.**		0	0	1	0		0	1	0	0		1	0	0	0		1	2	4	8
26		1	1	1	0		0	0	**1.**	1		0	1	0	1		1	0	0	1		E	3	5	9
27		1	1	0	0		1	1	0	0		0	**1.**	1	1		1	0	1	1		C	C	7	B
28		1	0	0	0		1	0	0	0		1	0	0	0		**1.**	1	1	1		8	8	8	F
29		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0

Figure 9 – Illustration of the Sign Method Algorithm

Source: author's development

Though the algorithm does not find the optimal solution, its analysis gives us useful insights. While tracing each step of the algorithm, we repeatedly identified combinations which could reliably bring us to the pre-final and final combinations much earlier than the algorithm did. This may be explained by the fact that the player does not analyze the current combination, rather she activates a cell that is provided by the rule, and not the one that is required by the logic. Therefore, when applying this algorithm, it is advisable not to mechanically follow its steps but to analyze each current combination in order to timely detect the pre-final one thus effectively impacting the algorithm flow in order to finish the game earlier. This will significantly reduce the number of steps and make the game much more interesting.

We tried finding more optimal algorithms that are based on the rigid sequence of actions. We called the first of them as the «stream» method. We have identified several modifications and optimization paths for it.

The «stream» method. The essence of this method for the 4x4 matrix is as follows.

All cells with code «0» of the input matrix are numbered from the first row to the last, traversing them clockwise (similar to the flow of the stream) (Fig. 10, 11).

Below we only use addresses of the numbered input cell. We activate the cell with number 1 in the input combination and get a combination with number 1 (see Figure 11). In the resulting combination we activate the cell that is located at the address of the cell with number 2, regardless of its contents. In combination number 2 we activate the cell with number 3 and so on. After activating the last numbered cell of the input matrix, we get a combination (No. 8 in Figure 11), which we set as the basis for further action.

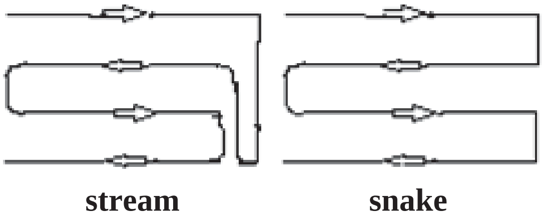

Figure 10 – Cell treversal type

Source: author's development

	Input					1					2					3					4					5				
	1	**0**^{1}**.**	1	0^{2}	⇨	0	1	0	**1.**	⇨	1	0	1	0	⇨	1	0	1	1	⇨	1	0	1	0	⇨	0	0	1	0	⇨
	0^{5}	1	1	0^{3}		0	0	1	0		0	0	1	**1.**		1	1	0	0		**1.**	1	0	1		0	0	1	0	
	1	1	0^{6}	1		1	0	0	1		1	0	0	0		1	0	0	1		1	0	0	0		0	0	**0.**	0	
	0^{8}	1	0^{7}	0^{4}		0	0	0	0		0	0	0	1		0	0	0	**0.**		1	1	1	1		0	1	1	1	
	6					7					Basic					1					2					3				
⇨	0	0	0	0	⇨	0	0	1	0	⇨	**1**^{1}**.**	0	1^{2}	0	⇨	0	1	**0.**	1	⇨	1	0	1	0	⇨	0	0	1	0	⇨
	0	0	0	0		0	0	1	0		1^{3}	0	1^{4}	0		0	0	1	0		**0.**	0	0	0		1	1	**1.**	1	
	1	1	1	1		1	1	0	1		0	1^{5}	0	1^{6}		1	1	0	1		1	1	1	1		0	1	1	1	
	0	1	**0.**	1		**1.**	0	1	0		0	1^{7}	0	1^{8}		1	1	0	1		1	1	1	1		0	1	1	1	
	4					5					6					7					8									
⇨	0	0	0	0	⇨	0	1	0	0	⇨	0	1	0	1	⇨	0	0	0	1	⇨	0	0	0	0		The goal is achieved in 16 steps				
	0	0	0	0		0	1	0	0		0	1	0	1		0	0	0	1		0	0	0	0						
	0	**1.**	0	1		1	0	1	**0.**		0	1	0	1		0	0	0	1		0	0	0	0						
	0	1	0	1		0	0	0	1		0	**0.**	0	0		1	1	1	**1.**		0	0	0	0						

Figure 11 - Algorithm for solving the problem by the stream method

Source: author's development

Now we enumerate its cells which have code «1» by sequentially traversing the rows starting from the beginning of the first row to its end, then from the end of the second row to the beginning of it, again from the beginning of the third row to its end and from the end of the fourth row to its beginning.

This traversal looks like the snake movement (Fig. 10). Starting from the base combination we keep executing the sequence of steps of the described above algorithm by activating the cell with the next sequential number. Again we get combination number 1. We activate its cell with number 2, etc. After 16 steps we get the desired result - the «zero» combination.

The «snake» method differs from the **«stream» method** only by the way of traversing the positions of the input matrix while enumerating the cells. The cells of the input matrix and the following base matrix are numbered identically along the trajectory of the snake movement (Fig. 10). The sequence of actions in both algorithms is the same. In all combinations, namely incoming, base, and current, cells with the number and location of which are determined in the input and base matrices are activated. The method for determining the cell numbers and the traversal algorithm is presented in Fig. 12.

The simulation results showed that in order to reach the zero matrix by the snake method it is possible to start enumeration of the input matrix cells either with code «0» cells or code «1» cells. In any case the activation begins with the first numbered cell of the input matrix. In the base combination, cells with code «1» are enumerated. To reach the «unit» combination the cells with the code «0» of the input and base combinations are enumerated.

If the input matrix has the number of cells with zeros greater than 7, then it is desirable to change them to codes «1» at the first step. To do this, you need to activate a cell at the intersection of a row and a column with the largest number of zero cells. Then the resulted combination shall be taken as the input matrix and you need to perform the above sequence of actions on it. This procedure will reduce the number of steps, and therefore **optimizes the process**. An example of optimizing the exit process from the input matrix given in Fig. 12 to «zero» matrix is shown in Fig. 13. As the figure demonstrates the total number of steps is decreased by 2 in comparison to the example presented in Fig. 12, and the number of steps to reach the "zero" matrix from the simplified input matrix is 12.

	Input					1					2					3					4					5				
	1	**0^1.**	1	0^2	⇨	0	1	0	**1.**	⇨	1	0	1	0	⇨	1	1	1	0	⇨	0	1	1	0	⇨	0	0	1	0	⇨
	1	0^3	1	1		1	1	1	1		1	**1.**	1	0		0	0	0	1		1	0	0	1		1	1	0	1	
	0^4	0^5	0^6	1		0	1	0	1		0	1	0	0		**0.**	0	0	0		1	**1.**	1	1		0	0	**0.**	0	
	0^9	0^8	1	0^7		0	1	1	0		0	0	1	1		0	0	1	1		1	0	1	1		1	1	1	1	
	6					7					8					Basic					1					2				
⇨	0	0	0	0	⇨	0	0	0	1	⇨	0	1	0	1	⇨	**1^1.**	1^2	0	1^3	⇨	0	**0.**	1	0	⇨	1	1	0	**1.**	⇨
	1	1	1	1		1	1	1	0		1	0	1	0		0	0	1^4	0		1	0	1	0		1	1	1	0	
	1	1	1	1		1	1	1	0		1	0	1	0		0	0	1^5	0		1	0	1	0		1	1	1	0	
	1	1	0	**1.**		0	**0.**	1	0		**1,**	1	0	1		0	0	1^6	0		1	0	1	0		1	1	1	0	
	3					4					5					6														
⇨	0	0	1	0	⇨	0	0	0	0	⇨	0	0	1	0	⇨	0	0	0	0							The goal is achieved in 15 steps				
	1	1	**1.**	1		0	0	0	0		0	0	1	0		0	0	0	0											
	1	1	1	1		1	1	**0.**	1		0	0	**1.**	0		0	0	0	0											
	1	1	1	1		1	1	0	1		1	1	1	1		0	0	0	0											

Figure 12 - Algorithm for solving the problem by the snake method

Source: author's development

Analysis of feasibility for solving the "Flip-Flop" problem for the 3x4 matrix. The number of final combinations for the 3x4 matrix is equal to the number of its cells $3 \times 4 = 12$. Since the 12 pre-final combinations can be obtained from $12 \times 11 = 132$ the pre pre-final ones, the probability of solving such problem is high.

Fig. 14 shows an example of solving the problem by using the above-mentioned «snake» method which can find a solution in 10 steps.

						Input					1					2					3					Basic				
	1	0	1	0		1	1	1	**0.**^{1}		0	0	0	1		0	0	0	0		0	0	0	1		**1.**^{1}	0	0	1^{2}	
	1	0	1	1	⇨	1	1	1	1	⇨	1	1	1	0	⇨	1	1	1	1	⇨	1	1	1	0	⇨	0	1^{4}	1^{3}	0	⇨
	0	**0.**	0	1		1	1	1	0^{2}		1	1	1	**1.**		0	0	0	0		0	0	0	1		1^{5}	0	0	1^{6}	
	0	0	1	0		0^{4}	1	1	0^{3}		0	1	1	1		0	1	1	**0.**		**1.**	0	0	1		0	1^{8}	1^{7}	0	
	1					2					3					4					5					6				
	0	1	1	**0.**		1	0	0	1		1	0	1	1		1	1	1	1		0	1	1	1		0	1	1	0	
⇨	1	1	1	0	⇨	1	1	**1.**	1	⇨	0	**0.**	0	0	⇨	1	1	1	1	⇨	0	1	1	1	⇨	0	1	1	0	⇨
	0	0	0	1		0	0	0	0		0	0	1	0		**0.**	1	1	0		1	0	0	**1.**		0	1	1	0	
	1	1	1	0		1	1	1	1		1	1	0	1		1	0	0	1		0	0	0	1		0	0	**0.**	0	
	7					8																								
	0	1	0	0		0	0	0	0																	The goal is achieved in 13 steps				
⇨	0	1	0	0	⇨	0	0	0	0																					
	0	1	0	0		0	0	0	0																					
	1	**1.**	1	1		0	0	0	0																					

Figure 13 – An example of optimizing the solution process using the snake method
Source: author's development

When you get to the basic combination, the question arises if we need to enumerate cells with units or zeros. When enumerating cells with code «1» the process proved to be long, and enumerating of zeros quickly leads to the solution. It required 17 steps to solve the problem by the sign method, however the player has to timely detect the pre pre-final combination and «manually» interfere in the solution process. It is encouraging that the above methods were also suitable for matrices of size $m \times n$. However, solving the problem for matrices of higher dimension were not investigated in detail.

	Input					1					2					3					4					5				
	0.^{1}	1	0^{2}	1		1	0	1	0		0	1	0	1		0	1	1	1		0	0	1	1		0	0	0	1	
	1	1	0^{3}	1	⇨	0	1	0	1	⇨	0	1	1	0	⇨	1	0	0	0	⇨	1	1	0	0	⇨	1	1	1	0	⇨
	1	0^{4}	0^{5}	0^{6}		0	0	0	0		0	0	1	0		0	0	0	0		1	1	1	1		0	0	0	0	
	Basic					7					8					9					10					11				
	0.^{1}	0^{2}	0^{3}	0^{4}		1	**1.**	1	1		0	0	**0.**	0		1	1	1	**1.**		0	0	0	0		The goal is achieved in 10 steps				
⇨	1	1	1	1	⇨	0	1	1	1	⇨	0	0	1	1	⇨	0	0	0	1	⇨	0	0	0	0						
	1	1	1	1		0	1	1	1		0	0	1	1		0	0	0	1		0	0	0	0						

Figure 14 - An example of optimizing the solution process of the snake method
Source: author's development

Conclusions. The results of the study showed that applying logical analysis to each combination in 4x4 and higher dimensionality matrices in order to find the optimal next step is a complex and inefficient process.

To solve the problem at hand it is necessary to find non-standard approaches such as the method of identifying distinct features and using them in the course of the algorithm development; the method for allocating cells with code «1» or «0» in the input matrix along with their enumeration rules and developing on this basis the rules for finding the successful sequence of actions (i.e. «stream» and «snake» methods). When optimizing these algorithms, it is necessary to analyze the current combinations and either timely interfere in the process (like in the sign method) or pre-process the input matrix in order to optimize the number of cells with code «0» or «1» (like in the «snake» method).

Список літератури

1. Раскин М.А. Введение в теорию игр // Летняя школа «Современная математика». Дубна, 2008. URL: https://www.mccme.ru/dubna/2008/courses/raskin.htm (дата обращения: 08.11.2019)
2. Мазалов В.В. Математическая теория игр и приложения. – Санкт-Петербург - Москва - Краснодар: Лань, 2010. 446 с.
3. Яблонский С.В. Введение в дискретную математику. Москва: Высш. шк., 2003. 384 с.
4. Міщенко Н. Штучний інтелект-виклик часу. *Науковий світ*. 2006. № 10. С. 12-13
5. Виленкин Н.Я. Популярная комбинаторика. Москва: Наука, 1975. 208 с.

References

1. Raskin, M.A. (2008). Vvedeniye v teoriyu igr // Letnyaya shkola «Sovremennaya matematika». Dubna, *www.mccme.ru.* Retrieved from: https://www.mccme.ru/dubna/2008/courses/raskin.htm [in Russian].
2. Mazalov, V.V. (2010). *Matematicheskaya teoriya igr i prilozheniy [Mathematical game theory and applications]*. Sankt-Peterburg - Moskva - Krasnodar: Lan' [in Russian].
3. Yablonskiy,S.V. (2003). *Vvedeniye v diskretnuyu matematiku [Introduction to Discrete Mathematics]*. Moskow: Vyssh. shk. [in Russian].
4. Míshchenko, N. (2006). Shtuchniy íntelekt-viklik chasu [Artificial Intelligence Challenge of Time]. *Naukoviy svít, № 10*, 12-13 [in Ukrainian]
5. Vilenkin, N.YA. (1975). *Populyarnaya kombinatorika [Popular combinatorics]*. Moskow: Nauka [in Russian].

Ю.М. Пархоменко, доц., канд. техн. наук, **М.Д. Пархоменко, Л.В. Рибакова, А.Р. Бокій**
Центральноукраїнський національний технічний університет, м.Кропивницький, Україна

Дослідження методів розв’язання ігор-головоломок типу «Flip-Flop»

Метою статті є визначення методів розв’язання задачі зведення довільної комбінації кодів матриці, з мінімальним числом кроків, до одних «0» або «1» та побудова алгоритмів, які забезпечать цей процес.

Як показала практика гри «Flip-Flop», досягнення мети - зведення довільної комбінації кодів матриці, в результаті кінцевого числа кроків, до одних «нулів» або «одиниць», шляхом аналізу ситуацій отриманих після кожного кроку, є задачею не простою. Логіку зміни комбінацій на два, три кроки уперед важко передбачити, так як навіть в матриці 4×4 з кожним кроком одночасно змінюється вміст семи комірок із 16. Тому, без визначення алгоритму послідовності дій, задача розв’язання даної головоломки, навіть при необмеженому числі кроків, не завжди досяжна. З метою визначення методів розв’язання задачі - алгоритмів виконання послідовності дій, які гарантовано або з мінімальною кількістю кроків забезпечать формування «0»-ї або «1»-ї матриці на основі довільно заданої початкової комбінації, було детально о досліджено процес формування комбінацій двійкових кодів матриці розміром m x n при активації окремих її комірок і методів прийняття рішень.

У роботі приведені результати дослідження методів розв’язання комп’ютерних ігор-головоломок типу «Flip-Flop» з матрицями розміром 3×3, 3×4, 4×4 з використанням методів логічного аналізу ситуацій, комбінаторики та дискретної математики. З’ясували, що застосування методу логічного аналізу кожної поточної комбінації, який добре працює в матрицях розміром 3×3, в матрицях 4×4 і вище, для прийняття оптимального рішення при визначенні наступного кроку, є процесом складним і неефективним. Тому вирішили шукати алгоритми у визначенні постійної послідовності дій за певною ознакою. Розглянули та проаналізували методи виділення певних ознак, метод «струмка» та метод «змійки».

Виконані дослідження показали, що для оптимізації кількості кроків, при застосуванні вказаних алгоритмів, необхідно аналізувати поточні комбінації і своєчасно вмішуватися в процес, або попередньо обробити вхідну матрицю, з метою оптимізації кількості комірок з кодами «0» або «1».

комп'ютерні ігри-головоломки, логічний аналіз ситуацій, комбінаторика, дискретна математика метод виділення ознак, метод «струмка», метод «змійки»

Одержано (Received) 11.12.2019 *Прорецензовано (Reviewed) 16.12.2019*
Прийнято до друку (Approved) 23.12.2019

УДК 681.58 DOI: https://doi.org/10.32515/2664-262X.2019.2(33).201-208

В.В. Смирнов, доц., канд. техн. наук, **Н.В. Смирнова**, доц., канд. техн. наук
Центральноукраинский национальный технический университет, г. Кропивницкий, Украина
e-mail: swckntu@gmail.com

Использование статистических методов в системе радиоуправления робототехническим объектом

Приведена реализация системы управления радиоуправляемым объектом в условиях потери связи с оператором. В общий контур радиоуправления объектом введен локальный регулятор, который позволил стабилизировать объект управления в условиях автономного аварийного режима. Для уменьшения влияния случайных возмущений на объект управления в систему управления введен статистический блок, реализующий положения теории принятия статистических гипотез. Уменьшено время реакции системы на изменение значения задающего воздействия. Разделены циклы получения задающего воздействия и циклы управления сервоприводами.

система управления, сервопривод, люфт, объект, статистическая гипотеза

В.В. Смірнов, доц., канд. техн. наук, **Н.В. Смірнова,** доц., канд. техн. наук
Центральноукраїнський національний технічний університет, м Кропивницький, Україна

Використання статистичних методів в системі радіоуправління робототехнічним об'єктом

Наведено реалізація системи управління керованим по радіо об'єктом в умовах втрати зв'язку з оператором. У загальний контур радіоуправління об'єктом введений локальний регулятор, який дозволив стабілізувати об'єкт управління в умовах автономного аварійного режиму. Для зменшення впливу випадкових збурень на об'єкт управління в систему управління введений статистичний блок, який реалізує положення теорії прийняття статистичних гіпотез. Зменшено час реакції системи на зміну значення впливу що задається. Розділені цикли отримання впливу що задається і цикли управління сервоприводами.

система управління, сервопривід, люфт, об'єкт, статистична гіпотеза

Постановка проблемы. В настоящее время существует много разновидностей недорогих систем радиоуправления робототехническими объектами различного назначения. В составе системы радиоуправления находится оператор, который и осуществляет управление объектом при визуальном контакте с ним или по данным телеметрии.

В такой системе управления оператор является задатчиком и регулятором. Оператор передает объекту задающее воздействие и в процессе управления объектом