Erciyes Üniversitesi
İktisadi ve İdari Bilimler Fakültesi Dergisi
e-ISSN: 2630-6409

Araştırma Makalesi | Research Article
Yayın Tarihi: 30 Ağustos 2025 | Published: August 30, 2025
doi: 10.18070/erciyesiibd.1666419

***Sorumlu Yazar/ Corresponding Author:**
Banu SUNGUR,
bitgenb@erciyes.edu.tr

JEL:
C60, C61, C69

***Geliş:** 26 Mart 2025*
***Received:** March 26, 2025*
***Kabul:** 10 Haziran 2025*
***Accepted:** June 10, 2025*
***Yayın:** 30 Ağustos 2025*
***Published:** August 30, 2025*

Atıf / Cited as (APA):
Sungur, B. & Madenoğlu, F. S. (2025), An Integer Programming Formulation for Generating and Solving Survo Puzzle, Erciyes Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 71, 41-44, doi: 10.18070/erciyesiibd.1666419

AN INTEGER PROGRAMMING FORMULATION FOR GENERATING AND SOLVING SURVO PUZZLE

BANU SUNGUR [1*] ***FATMA SELEN MADENOĞLU [2]***

[1] Doç. Dr., Erciyes Üniversitesi, İktisadi ve İdari Bilimler Fakültesi, bitgenb@erciyes.edu.tr
[2] Doç. Dr., Abdullah Gül Üniversitesi, Yönetim Bilimleri Fakültesi, selen.madenoglu@agu.edu.tr

ABSTRACT

Puzzles are often generated for entertainment but also mathematical or logical problems. Every puzzle has its logic and mathematics. Puzzles become more understandable when we can grasp and model the underlying logic. For this reason, puzzles constitute a research area of interest to scientists. One of the logic puzzles is the Survo puzzle. We have formulated an integer linear programming model to generate and solve this puzzle. Illustrative examples have been given to show the validity of the formulation. The model's effectiveness has been tested by solving Survo puzzles available on the internet. The solutions have been obtained in short CPU times. Then, the effectiveness of the model has been analyzed using experimental computations. The computational results have been obtained from a number of Survo puzzle instances. The proposed mathematical model has generated puzzles up to 50x50 size in short CPU times, a maximum of 254 seconds. Puzzles up to size 15x15 have been solved.

Keywords: Survo, Puzzle, Mathematical formulation, Integer programming

SURVO BULMACASININ OLUŞTURULMASI VE ÇÖZÜLMESİ İÇİN TAMSAYILI PROGRAMLAMA FORMÜLASYONU

ÖZ

Bulmacalar genellikle eğlence için üretilir ancak aynı zamanda matematiksel veya mantıksal problemlerdir. Her bulmacanın kendine özgü bir mantığı ve matematiği vardır. Altta yatan mantığı kavrayıp modelleyebildiğimizde bulmacalar daha anlaşılır hale gelir. Bu nedenle bulmacalar bilim insanlarının ilgisini çeken bir araştırma alanı oluşturmaktadır. Mantık bulmacalarından biri de Survo bulmacasıdır. Bu bulmacayı oluşturmak ve çözmek için bir tamsayılı doğrusal programlama modeli formüle edilmiştir. Formülasyonun geçerliliğini göstermek için açıklayıcı örnekler verilmiştir. Modelin etkinliği internette bulunan Survo bulmacaları çözülerek test edilmiştir. Çözümler kısa işlemci sürelerinde elde edilmiştir. Daha sonra, modellerin etkinliği deneysel hesaplamalar kullanılarak analiz edilmiştir. Hesaplama sonuçları bir dizi Survo bulmaca örneği üzerinden elde edilmiştir. Önerilen matematiksel model, 50x50 boyutuna kadar bulmacaları kısa CPU sürelerinde, maksimum 254 saniyede üretmiştir. 15x15 boyutuna kadar olan bulmacalar çözülmüştür.

Anahtar Kelimeler: Survo, Bulmaca, Matematiksel programlama, Tamsayılı programlama

INTRODUCTION

Survo is a kind of logic-based puzzle presented by Mustonen in 2006. In this puzzle, the goal is to fill *mxn* grid of cells with integers (1,2,...,*mxn*) so that no integer is repeated in the grid. These integers are placed in each cell so as to satisfy row and column sums. Some integers are given readily in the table to guarantee uniqueness and/or to facilitate the solution. Survo puzzle's challenging level might vary based on factors like the size of the grid and the number of empty cells (Mustonen, 2007). The number of missing values is represented as *k*. There are *k!* ways of ordering *k* numbers (Jones et al., 2011). Thus, the overall complexity of this puzzle is (*k!*). This is a non-polynomial function. This complexity shows that this puzzle is NP. In computational complexity theory, NP is a complexity class that describes certain types of decision problems.

Figure 1 shows the 3 x 4 example of the Survo puzzle and its solution. Numbers outside the grid indicate the total of cells in the corresponding row or column. The sum of the cells in the first, second , and third rows must be 30, 18, and 30, respectively. 27, 16, 10, and 25 are the sums of the integers to be written in the relevant columns from the first to the fourth column, respectively. For this example integers 3, 6 and 8 are given readily. There are nine blank cells (*k*=9) in the grid. Then we are asked to fill these nine empty cells with 1, 2, 4, 5, 7, 9, 10, 11, 12. These integers are written in blank cells, providing row and column totals.

FIGURE 1| Survo puzzle example (3x4)

	6			30
8				18
		3		30
27	16	10	25	

12	6	2	10	30
8	1	5	4	18
7	9	3	11	30
27	16	10	25	

In this paper, we have formulated an integer linear programming (ILP) model to generate and solve Survo puzzles.The remainder of the paper is organized as follows. In Section 2, a literature review is presented. Section 3 is devoted to the mathematical formulation of Survo puzzle. The computational experiments are presented in Section 4. Finally, concluding remarks are given in Section 5.

1. Literature Review

Puzzles have been studied both conceptually and mathematically in the literature. Mathematical models have been proposed for different puzzles. Hurliman (2015) has written a book that is a comprehensive guide to mathematical modeling for puzzles and games. The author has proposed mathematical models that can be adapted to a wide range of puzzle types and game scenarios, emphasizing the role of mathematical optimization in recreational problem solving. Sudoku has the most attention in the scientific community. Various mathematical models have been formulated to generate and solve them. Bartlett et al. (2008) have presented a binary integer linear programming (BILP) model to solve Sudoku puzzles. Their model formulates the placement of digits in each cell using binary variables and expresses the puzzle's rules through linear constraints. Coelho and Laporte (2014) have described two mathematical programming models and four enumerative algorithms for Sudoku. They have conducted a comparative analysis of these algorithms regarding solution performance, providing valuable insights into the effectiveness of different solving techniques. Yu et al. (2016) have proposed a BILP model to solve Odd/Even Sudoku puzzles. Their model incorporates standard Sudoku rules and constraints requiring certain cells to contain only odd or even digits. Reddy et al. (2016) have presented a BILP model to solve Sudoku puzzles, demonstrating the applicability of binary programming approaches to this widely studied problem. Ates and Cavdur (2025) have proposed a new mathematical programming formulation for generating Sudoku puzzles. They have discussed how to ensure the uniqueness of a solution for a puzzle instance generated by a hybrid approach that integrates the mathematical program with a heuristic algorithm. Beyond Sudoku, other logic-based puzzles have also begun to receive attention in the puzzle literature. Hinz et al. (2009) have proposed a mathematical model and computer tool to solve the Tower of Hanoi and Tower of London puzzles. Often used in cognitive science research, these puzzles have been modeled analytically and their solution spaces have been systematically explored. Meuffels and Hertog (2010) have solved a logical puzzle, the Battleship, by ILP model. Their approach has involved representing ship sizes, orientations, and valid placements through a set of linear constraints, ensuring compliance with the game's rules. Trindade et al. (2011) have presented a mixed integer linear programming (MILP) model for solving the Shisen-Sho puzzle, a tile-matching game of Chinese origin. The model captures the puzzle's matching and connection rules through mathematical formulations. Chlond (2015) has presented an ILP model for solving Wijuko and ABC logic puzzles. Puzzles' rules have been encoded as linear inequalities, allowing the derivation of optimal solutions using exact methods. Keçeci (2021) has proposed a MILP formulation for solving and generating Smashed Sums puzzles. Sungur (2022) has developed an ILP model to solve the Futoshiki puzzles. The model enforces the uniqueness of numbers in each row and column and the satisfaction of inequality conditions between adjacent cells. Burkardt and Garvie (2023) have developed an ILP model to solve the Eternity Puzzle, a notoriously complex tiling puzzle. Their work highlights the potential of ILP approaches in handling puzzles with large solution spaces and intricate placement constraints.

The Survo puzzle has not been studied much in the literature. Mustonen (2006) has suggested algorithms for solving Survo puzzles. Vehkalahti and Sund (2015) have formulated a Survo puzzle as a combinatorial matrix problem and developed a computational approach with binary matrices to solve Survo puzzles. They noted that this method, while not computationally efficient, is an interesting way to solve Survo puzzles.

To our knowledge, the ILP model has never been considered in the Survo puzzle literature so far. In the paper, we have presented an ILP model for generating and solving Survo puzzles for the first time.

2. Mathematical Formulation for Survo Puzzle

This section presents a mathematical formulation for the Survo puzzle. First, we give the notation which will be used throughout the section:

i rows ($\forall i \in I$)
j columns ($j \in J$)
s integer numbers ($s \in S$)
m number of rows
n number of columns
I set of rows
J set of columns
S set of integers
R_i the sum of row i ($\forall i \in I$)
C_j the sum of column j ($j \in J$)
A dummy variable
x_{ijs} 1, if integer number s is assigned to cell(i,j); 0, otherwise ($\forall i \in I, \forall j \in J, \forall s \in S$)
y_i the sum of row i ($\forall i \in I$)
z_j the sum of column j ($\forall j \in J$)

The sets of I={1,⋯,*m*} and J={1,⋯,*n*} denote the rows and columns, respectively. The set S={1,⋯,*mxn*} denotes the integers assigned to the cells. The puzzle consists of *mxn* integers, each of which can be assigned to only one cell.

The integer linear programming model to generate Survo puzzles can be formulated as follows:

$$Min\, C = A \quad (1)$$

subject to:

$$\sum_s x_{ijs} = 1, \forall i \in I, \forall j \in J \quad (2)$$

$$\sum_i \sum_j x_{ijs} = 1, \forall s \in S \quad (3)$$

$$\sum_j \sum_s sx_{ijs} = y_i, \forall i \in I \tag{4}$$

$$\sum_i \sum_s sx_{ijn} = z_j, \forall j \in J \tag{5}$$

$$x_{ijs} \in \{0,1\}, \forall i \in I, \forall j \in J, \forall s \in S \tag{6}$$

$$y_i = 0,1,2 \ldots, \forall i \in I \tag{7}$$

$$z_j = 0,1,2 \ldots, \forall j \in J \tag{8}$$

The objective function (1) minimizes the dummy variable A. The constraints are as follows: (2) places only one integer s in each cell (i,j). (3) ensures that each integer s must be placed only in one cell. (4) determines the total sum of row i (y_i). (5) determines the total sum of column j (z_j). (6), (7) and (8) define restrictions on the variables.

The model defined by (1)–(8) has $[(mxn)^2+m+n]$ variables and $[(2xmxn)+m+n]$ linear constraints.

The model formulated to generate Survo puzzles can also solve these puzzles with a little modification. The right-hand sides of constraints (4) and (5) are replaced by parameters R_i and C_j respectively and these two constraints are rewritten into the model as follows:

$$\sum_j \sum_s sx_{ijs} = R_i, \forall i \in I \tag{9}$$

$$\sum_i \sum_s sx_{ijs} = C_j, \forall j \in J \tag{10}$$

Constraint (9) ensures that the total value of the integer numbers assigned to the row i is equal to the given row sum R_i. Constraint (10) ensures that the total value of the integer numbers assigned to the column j is equal to the given column sum C_j. This model has $[(mxn)^2]$ variables and $[(2xmxn)+m+n]$ linear constraints.

The proposed model has been illustrated on the examples to first generate and then to solve the Survo puzzles. We can examine how the parameters and decision variables take values according to these illustrative puzzles.

The model has been first applied to generate a 5x5 Survo puzzle with m=5,n=5. There are 25 integers (s=1,2,...,25) to be placed in the table. The optimal solution is given in the Table 1.

TABLE 1 | The solution for the puzzle generation example

y_i		z_j		x_{ijs}				x_{ijs}				x_{ijs}				x_{ijs}				x_{ijs}			
i	Value	*j*	Value	*i*	*j*	*s*	Value	*i*	*j*	*s*	Value	*i*	*j*	*s*	Value	*i*	*j*	*s*	Value	*i*	*j*	*s*	Value
1	48	1	90	1	1	9	1	2	1	21	1	3	1	23	1	4	1	24	1	5	1	13	1
2	73	2	46	1	2	6	1	2	2	7	1	3	2	11	1	4	2	5	1	5	2	17	1
3	89	3	38	1	3	3	1	2	3	1	1	3	3	22	1	4	3	10	1	5	3	2	1
4	66	4	83	1	4	14	1	2	4	20	1	3	4	18	1	4	4	19	1	5	4	12	1
5	48	5	67	1	5	6	1	2	5	25	1	3	5	15	1	4	5	8	1	5	5	4	1

A 5x5 Survo puzzle has been generated with its solution as seen in Figure 2. For instance, the sum of the integers placed in the first row is 48, that's why y_1=48. Similarly z_1=90 for column 1. The number 7 is placed into the cell (2,2), that's why x_{227}=1 . Similarly x_{3322}=1 means 22 is placed in cell (3,3).

FIGURE 2 | Survo puzzle example (5x5

9	6	3	14	16	48
21	7	1	20	25	73
23	11	22	18	15	89
24	5	10	19	8	66
13	17	2	12	4	48
90	46	38	83	6	

The integer programming model has been also applied to solve the illustrative example in Figure 1. In this example: $m=3$, $n=4$, $R_1=30$, $R_2=18$, $R_3=30$, $C_1=27$, $C_2=16$, $C_3=10$, $C_4=25$.

In the illustrative puzzle, integers '*1*' to '*12*' (s=1,2,...,12) should be placed in cells considering row and column sums. Three integers (3,6,8) are given in advance. The integer 6 has been written in the 2nd column of the *1st row*, that's why x_{126}=1. Similarly x_{218}=1 and x_{333}=1. These three variable values have been written as constraints to the model. Other cells are blank. When the model is solved, the remaining integers (1,2,4,5,7,9,10,11,12) are placed in the puzzle as shown in the solution below:

x_{1112}=1, x_{132}=2, x_{1410}=1, x_{221}=1, x_{235}=1, x_{244}=1, x_{317}=1, x_{329} = 1, x_{3411}=1.

For instance, x_{1112} =1 that's why the integer 12 is written in the *1st* column of the *1st* row cell.

3. Computational Experiments

The integer programming model has been coded in AIMMS. All computations are done on computer with Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz and 8,00 GB Ram. The experimental study is done for solving the given puzzles are available on the web site https://www.survo.fi/puzzles/. A total of 377 puzzles of similar sizes were shared between 2006 and 2016 on the web site. Among these puzzles, 24 puzzles shared in the last two years have been chosen. These puzzles have been solved by the proposed model. Puzzles and solutions are given in Table 2. The first six columns present information about puzzles and the last column expresses the puzzle's solutions. The problem numbers and difficulty levels of the puzzles specified on the website are in the first and third columns, respectively. Optimal solutions have been obtained for all puzzles. As seen in Table 2, the CPU times are short in duration.

TABLE 2 | Solution of Puzzles on the Internet

Puzzle No	Puzzle size	Difficulty degrees	*k*	Row Sums	Column Sums	CPU Time (sec)
354/2015	4x4	1000	16	40,30,18,48	51,16,45,24	0.03
355/2015	4x4	540	15	33,31,53,19	37,27,54,18	0.03
356/2015	4x4	400	15	46,23,31,36	13,54,45,24	0.05
357/2015	3x5	600	15	22,37,61	24,28,37,19,12	0.03
358/2015	3x5	250	14	21,45,54	32,25,20,35,8	0.03
359/2015	3x5	300	14	55,18,47	25,18,12,29,36	0.03
360/2015	4x4	260	14	32,35,43,26	44,25,56,11	0.08
361/2015	4x4	1000	16	41,47,19,29	17,24,55,40	0.03
362/2015	4x4	1000	16	37,24,43,32	43,11,26,56	0.03
363/2015	3x4	100	12	17,34,27	9,16,33,20	0.09
364/2015	3x5	500	15	31,42,47	26,42,13,8,31	0.03
365/2015	3x6	550	17	54,26,91	30,22,19,26,31,43	0.05
366/2015	3x5	165	14	36,62,22	19,30,12,24,35	0.03
367/2015	3x5	215	14	39,59,22	25,36,30,10,19	0.03
368/2015	3x6	2600	18	28,66,77	31,19,47,14,23,37	0.03
369/2015	4x4	650	15	31,24,44,37	43,57,19,17	0.03
370/2015	4x4	350	15	37,24,44,31	42,56,11,27	0.03
371/2015	4x4	550	16	50,37,17,32	19,27,34,56	0.03
372/2015	3x4	80	11	17,24,37	15,25,29,9	0.01
373/2015	3x5	270	14	23,38,59	18,11,25,38,28	0.01
374/2015	3x6	1550	18	32,50,89	22,13,43,30,37,26	0.05
375/2016	4x4	40	12	40,21,42,33	55,43,26,12	0.01
376/2016	4x4	120	12	33,44,20,39	24,14,43,55	0.02
377/2016	4x4	355	15	37,49,21,29	15,54,42,25	0.03

The puzzles found on the internet are small-sized puzzles. Survo puzzles becomes more difficult for larger i and j combinations. For that reason, we have generated the larger size puzzles by using proposed mathematical model. These generated puzzles are given in Table 3. The first column presents the problem size from 5x5 to 50x50. The second column shows the number of constraints, the third column shows the number of the variables. As seen in the last column (CPU times), all puzzles have been generated in a short time. The proposed mathematical model has generated puzzles up to 50x50 size. Larger-sized puzzles have not been generated. Solution process has been terminated by the solver.

TABLE 3 | Generated puzzles

Problem size	# Const	# Variables	CPU Time	Problem size	# Const	# Variables	CPU Time
5x5	60	635	0.032	30x25	1555	562555	3.672
5x10	115	2515	0.031	30x30	1860	810060	5.359
5x15	170	5645	0.094	30x35	2165	1102565	7.250
10x5	115	2515	0.016	30x40	2470	1440070	7.703
10x10	220	10020	0.047	35x25	1810	765685	4.969
10x15	325	22525	0.125	35x30	2165	1102565	7.625
10x20	430	40030	0.203	35x35	2520	1500695	10.328
15x5	170	5645	0.047	35x40	2875	1960075	14.516
15x10	325	22525	0.156	35x45	3230	2480705	123.922
15x15	480	50655	0.297	40x30	2470	1440070	10.281
15x20	635	90035	0.532	40x35	2875	1960075	13.906
15x25	790	140665	0.641	40x40	3280	2560080	156.453
20x10	430	40030	0.219	40x45	3685	3240085	212.141
20x15	635	90035	0.532	40x50	4090	4000090	253.625
20x20	840	160040	0.938	45x35	3230	2480705	124.281
20x25	1045	250045	1.578	45x40	3685	3240085	212.015
20x30	1250	360050	2.313	45x45	4140	4100715	36.234
25x15	790	140665	0.828	45x50	4595	5062595	47.938
25x20	1045	250045	1.594	45x55	5050	6125725	55.515
25x25	1300	390675	2.500	50x40	4090	4000090	250.609
25x30	1555	562555	3.641	50x45	4595	5062595	46.187
30x20	1250	360050	2.109	50x50	5100	6250100	67.781

We have solved newly generated larger size puzzles by proposed ILP model. Solutions are given in Table 4. In 15x15 puzzle, the optimum solution has been found when 12 integers (5% of the total integers) were given readily. Except for tthe 15x15 puzzle, no integers have been given readily in the other dimensional puzzles. The optimum solutions have not been found for other dimensional puzzles not included in Table 4.

TABLE 4 | Solution of the generated puzzles

Problem size	# Constraints	# Variables	CPU Time(sec)
5x5	60	625	0.16
5x10	115	2500	1.27
5x15	170	5625	4.55
10x5	115	2500	0.91
10x10	220	10000	9.47
10x15	325	22500	878.97
10x20	430	40000	198,89
15x5	170	5625	895.83
15x10	325	22500	522.44
15x15	491	50625	4170.22

Concluding Remarks

Mathematical models exist in a wide variety of puzzle applications, however, it has not been considered in the Survo puzzle literature so far. Survo puzzle is considered in this paper. This paper contributes to the literature by introducing, for the first time, an ILP model for Survo puzzle. We have formulated an ILP model for generating and solving Survo puzzles.

A computational study has been conducted to depict the capability of the mathematical model. The complexity of the corresponding formulation is analyzed computationally. The puzzles on the related website are solved using the proposed ILP model. In addition to these puzzles, new puzzles up to 50x50 have been generated with the proposed model. Of these newly generated puzzles, puzzles up to the size of 15x15 have been solved.

REFERENCES

Bartlett, A., Chartier, T. P., Langville, A. N., & Rankin, T. D. (2008). An integer programming model for the Sudoku problem. *Journal of Online Mathematics and its Applications, 8*(1), 1798.

Burkardt, J., & Garvie, M. R. (2023). An integer linear programming approach to solving the Eternity Puzzle. *Theoretical Computer Science,* 975, 114138. https://doi.org/10.1016/j.tcs.2023.114138 0

Chlond, M. J. (2015). Puzzle—IP in the i. *INFORMS Transactions on Education, 16*(1), 39-41. https:// doi.org/10.1287/ited.2015.0142

Coelho, L. C., & Laporte, G. (2014). A comparison of several enumerative algorithms for Sudoku. *Journal of the Operational Research Society, 65*(10), 1602-1610. : https://doi.org/10.1057/jors.2013.114

Hinz, A. M., Kostov, A., Kneißl, F., Sürer, F., & Danek, A. (2009). A mathematical model and a computer tool for the Tower of Hanoi and Tower of London puzzles. *Information Sciences, 179*(17), 2934-2947. https://doi.org/10.1016/j.ins.2009.04.010

Hurliman, T. (2015). *Puzzles and games: a mathematical modeling approach.* Lulu. com.

Jones, S. K., Roach, P. A., & Perkins, S. (2011). Sudoku puzzle complexity. In *Proceedings of 6th Research Student Workshop* (pp. 19-24).

Keçeci, B. (2021). A mixed integer programming formulation for Smashed Sums puzzle: Generating and solving problem instances. *Entertainment Computing, 36,* 100386. https://doi.org/10.1016/j.entcom.2020.100386

Meuffels, W. J. M., & den Hertog, D. (2010). Puzzle—Solving the Battleship puzzle as an integer programming problem. *Informs Transactions on Education, 10*(3), 156-162. https://doi.org/10.1287/ited.1100.0047

Mustonen, S. (2006, June). *On certain cross sum puzzles.* http://www.survo.fi/papers/puzzles.pdf

Mustonen, S. (2007). *Enumeration of uniquely solvable open Survo puzzles.* Available from: http://www.survo.fi/papers/enum_survo_puzzles.pdf

Reddy, C. B., Reddy, K. K., & Upendra, D. (2016). An Integer Programming Model for the Sudoku Problem. *International Journal of Mathematics Trends and Technology, 40*(2), 164-169.

Sungur, B. (2022). An Integer Programming Formulation for the Futoshiki Puzzle. *International Review of Economics and Management, 10*(2), 38-49. http://dx.doi.org/10.18825/iremjournal.1149837

Ates, T., & Cavdur, F. (2025). Sudoku Puzzle Generation Using Mathematical Programming and Heuristics: Puzzle Construction and Game Development. *Expert Systems with Applications,* 282, 127710. https://doi.org/10.1016/j.eswa.2025.127710

Trindade, R. S., Dhein, G., Muller, F. M., & de Araujo, O. C. B. (2011, August). Mixed Integer Linear Programming Models to Solve the Shisen-Sho Puzzle. In 2011 Workshop-School on Theoretical Computer Science (pp. 108-112). IEEE.

Vehkalahti, K., & Sund, R. (2015). Solving Survo puzzles using matrix combinatorial products. *Journal of Statistical Computation and Simulation, 85*(13), 2666-2681. http://dx.doi.org/10.1080/00949655.2014.899363

Yu, H., Tang, Y., & Zong, C. (2016, August). Solving odd even sudoku puzzles by binary integer linear programming. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)* (pp. 2226-2230). IEEE.