

Citation: Bitgen Sungur, B. 2022. An Integer Programming Formulation For The Futoshiki Puzzle. *International Review of Economics and Management*, 10(2), 38-49. Doi: <http://dx.doi.org/10.18825/iremjournal.1149837>

AN INTEGER PROGRAMMING FORMULATION FOR THE FUTOSHIKI PUZZLE

Banu BİTGEN SUNGUR¹

Başvuru Tarihi: 27 / 07 / 2022 – Kabul Tarihi: 02 / 11 / 2022

Abstract

This paper is concerned with the problem of solving the Futoshiki puzzle. The Futoshiki, also known as “Unequal,” is a puzzle with an $n \times n$ grid containing inequality signs between the cells. Some digits may have been given at the beginning of the game. The aim is to fill in the empty cells so that each row and column contains the digits ‘1’ to ‘n’ without repeats. We have formulated an integer linear programming model to solve this problem. An illustrative example is given to show the validity of the model. The computational results are obtained and analyzed on some instances.

Keywords: Puzzle, Futoshiki, Mathematical Formulation, Integer Programming Model

JEL Classification: C60, C61, C69

FUTOSHIKI BULMACASI İÇİN BİR TAM SAYILI PROGRAMLAMA MODELİ


Özet

Bu çalışmada, Futoshiki bulmacası ele alınmıştır. "Eşitsiz" olarak da bilinen Futoshiki, hücreler arasında eşitsizlik işaretleri içeren $n \times n$ boyutlu bir bilmecedir. Oyunun başında bazı rakamlar verilmektedir. Amaç, satır ve sütunlarda her bir rakamdan bir tane olacak şekilde, boş hücreleri '1' ile 'n' rakamları ile doldurmaktır. Bu bulmacayı çözmek için tam sayılı bir doğrusal programlama modeli formüle edilmiştir. Modelin işleyişi örnek bir bulmaca üzerinden gösterilmiştir. Ayrıca, test problemleri çözülerek elde edilen sonuçlar analiz edilmiştir.

Anahtar Kelimeler: Bulmaca, Futoshiki, Matematiksel Formülasyon, Tam Sayılı Programlama Modeli

JEL Sınıflandırması: C60, C61, C69

¹ Assoc. Prof. Dr., Erciyes University, Faculty of Economics and Administrative Sciences, Department of Business, bitgenb@erciyes.edu.tr,

 <https://orcid.org/0000-0002-0233-4317>

I. INTRODUCTION

The Futoshiki puzzle is one of the Japanese challenging numeric puzzle and has gained in popularity globally. The puzzle is given as an $n \times n$ grid of cells. Initially, some cells contain numbers. The goal of this puzzle is to fill the empty cells with digits '1' to ' n .' Digits should not be repeated in each row and column. Though the puzzle is very similar to "Sudoku" there is a difference such that it contains inequality signs between some adjacent cells. The digits assigned to these cells must satisfy the inequality signs. Each puzzle has a unique solution.

Figure I. shows the 4×4 grid Futoshiki puzzle and its solution.

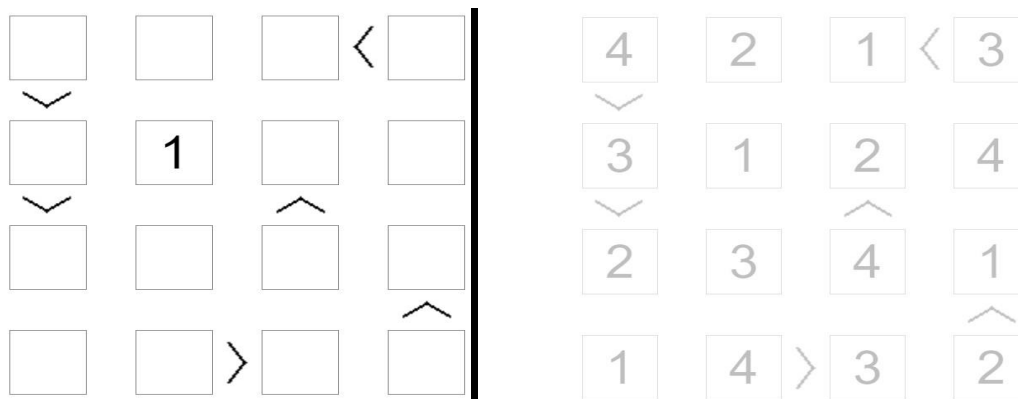


Fig.I. Futoshiki Puzzle Illustrative Example

In the illustrative puzzle, digits '1' to '4' (1, 2, 3, 4) should be written on each row and column. The digit '1' must be written in the cell in the second column of the second row. Signs should also be considered; for example, the digit to be written in the third column of the first row should be smaller than the digit in the first row and fourth column.

Let us explain how to solve the illustrative puzzle above manually. We have started from the second row because the digit '1' has already been given in this row. The digit '4' should be written in the last cell because of less than signs in other cells. When we move to the last column where we wrote the digit '4', we must write digit '1' under '4' because the digits in other cells cannot be the smallest due to the inequality signs. The solution is continued by writing the digit '1' in the last cell of the first column and the first cell of the third column. Considering the inequality signs, the digits '4', '3', and '2' are written in the cells of the first column, from top to bottom, respectively. If we look at the third column, since the digits in the first and fourth cells will be less than the digits next, the digit '4' is

written in the third row. Thus, most of the puzzle has been solved. The digits which will be written in the other cells also appeared obvious.

Puzzles have been attracting the attention of the academic community. Santos-Garcia & Palomino (2007) have considered how a Sudoku puzzle can be solved with rewriting logic. Hinz, Kostov, Kneißl, Sürer & Danek (2009) have proposed a mathematical model for the Towers of Hanoi and London puzzles. Waziri, Saidu & Musa (2010) have presented a mathematical method of solving Hausa puzzles. Maji, Jana & Pal (2013) have developed a heuristic algorithm to solve a Sudoku puzzle. Wang & Tang (2014) have studied the simulated annealing approach to solving multi-solution nonogram puzzles. Coelho & Laporte (2014) have developed two mathematical formulations for Sudoku puzzles. Boreland, Clement & Kunze (2015) have introduced a neural network model that combines set selection with partial memories to solve Sudoku and KenKen puzzles. Chlond (2015) has developed an integer programming model to generate solutions for Wijuko and ABC Logic puzzles. Yu, Tang & Zong (2016) have developed a binary integer linear programming model to solve Sudoku puzzles. Keçeci (2021) has formulated a binary integer linear programming model to solve the Smashed Sums puzzles.

Sudoku has received considerable attention from the scientific community. Various methods have been proposed to solve Sudoku puzzles. Lloyd, Crossley, Sinclair & Amos (2021) has indicated that the Futoshiki puzzle is NP-complete, and there exists relatively little work on its automated solution, although a candidate list strategy is given in Haraguchi (2013) and approximation algorithms are described in Haraguchi & Ono (2015). Haraguchi (2013) has studied how many inequalities there should be in the Futoshiki puzzle. Haraguchi & Ono (2015) have proposed simple generic approximation algorithms for Latin square Completion-Type puzzles, such as Sudoku, Futoshiki, and BlockSum, and have analyzed their approximation ratios. Mahmood (2019) has proposed a random number generator for the Futoshiki puzzle. Piette, Piette, Stephenson, Soemers & Browne (2019) have proposed using the XCSP formalism in order to solve logic puzzles, including the Futoshiki puzzle. Espasa, Gent, Hoffmann, Jefferson & Lynch (2021) have presented a new algorithm for efficiently finding small minimal unsatisfiable subsets to solve pen and paper puzzles, including the Futoshiki puzzle.

In this paper we have developed an integer programming model to solve the Futoshiki puzzle. To our knowledge, an integer linear programming model to solve Futoshiki puzzles has not been formulated in the literature so far.

The paper is arranged as follows. In Section II, the mathematical formulation is presented and model's solution for illustrative example is given. This is followed by computational results in Section 3. The final remarks are given in Section 4.

II. MATHEMATICAL FORMULATION AND ILLUSTRATIVE EXAMPLE

In this section, we present a mathematical formulation for the Futoshiki puzzle. First, we give the notation used in the mathematical formulation.

- i, l rows ($i, l \in T$)
- j, m columns ($j, m \in S$)
- r_{ij} the digit that is assigned to cell (i, j) at the beginning of the puzzle ($(i, j) \in R$)
- n puzzle dimension
- T set of rows
- S set of columns
- A set of adjacent cell pairs $((i, j), (l, m))$ with less than inequality sign between them such that (i, j) is assigned a smaller digit than (l, m)
- R set of non-empty cells (i, j)
- x_{ij} the digit assigned to cell (i, j) ($\forall i \in T, \forall j \in S$)
- y_{ijlm} 1, if the digit in the cell (i, j) is greater than the digit in the cell (l, m) ; 0, otherwise
($\forall i \in T, \forall j \in S, \forall l \in T, \forall m \in S$)($i = l, j \neq m$) or ($i \neq l, j = m$)
- Z the total value of digits assigned to any row or column of an n -dimensional puzzle
- M a large positive integer number

The sets of $T = \{1, \dots, n\}$ and $S = \{1, \dots, n\}$ denote the rows and columns, respectively, where the n is the puzzle dimension. A cell in the i^{th} row, j^{th} column and the l^{th} row, m^{th} column is denoted by (i, j) and (l, m) respectively. The digits (r_{ij}) to be written in some cells $(i, j) \in R$ are given in advance, where $R = \{(i, j) / (i, j) \text{ is non-empty cell}\}$ is the set of non-empty cells. Inequality signs exist between some pairs of two adjacent cells. These cells are given in pairs by set $A = \{(i, j), (l, m) / (i, j) \text{ and } (l, m) \text{ are adjacent and } (i, j) \text{ should be assigned a smaller}$

digit than (l,m). In each row and in each column, each digit should be assigned exactly once. The Futoshiki problem consists in finding the digits to be written in the empty cells of the n dimension puzzle. The integer programming formulation is as follows:

$$\text{Min } Z \tag{1}$$

Subject to:

$$x_{ij} \geq 1, \forall i \in T, \forall j \in S \tag{2}$$

$$x_{ij} \leq n, \forall i \in T, \forall j \in S \tag{3}$$

$$x_{ij} - x_{lm} - My_{ijlm} \leq -1, \forall i, l \in T, \forall j, m \in S, i = l, j \neq m \tag{4}$$

$$x_{ij} - x_{lm} - My_{ijlm} \leq -1, \forall i, l \in T, \forall j, m \in S, i \neq l, j = m \tag{5}$$

$$x_{lm} - x_{ij} + My_{ijlm} \leq M - 1, \forall i, l \in T, \forall j, m \in S, i = l, j \neq m \tag{6}$$

$$x_{lm} - x_{ij} + My_{ijlm} \leq M - 1, \forall i, l \in T, \forall j, m \in S, i \neq l, j = m \tag{7}$$

$$Z - \sum_j x_{ij} \geq 0, \forall i \in T \tag{8}$$

$$x_{ij} = r_{ij}, \forall (i, j) \in R \tag{9}$$

$$x_{ij} \leq x_{lm}, \forall ((i, j), (l, m)) \in A \tag{10}$$

$$x_{ij} \in \{1, 2, 3, \dots, n\}, \forall i \in T, \forall j \in S \tag{11}$$

$$x_{lm} \in \{1, 2, 3, \dots, n\}, \forall l \in T, \forall m \in S \tag{12}$$

$$y_{ijlm} \in \{0, 1\}, \forall i \in T, \forall j \in S, \forall l \in T, \forall m \in S \tag{13}$$

The objective function (1) minimizes the total value of digits assigned to one line (row or column) (Z). Constraint (2) ensures that the assigned digits are greater than or equal to one. Constraint (3) ensures that the largest assigned digits are equal to puzzle size (n). Constraints (4-7) guarantees that the digits to be written in the same row and column are different. Constraint (8) determines the total value of digits assigned to any row or column of an n -dimensional puzzle. This constraint is written for rows only. The sum of the digits assigned to the rows equals the sum of the digits assigned to the columns. Constraint (9) ensures that digits given at the beginning of the puzzle are assigned to the corresponding cells. Constraint (10) is written for the digits to be assigned to the cells, one of which is desired to be smaller

than the other. Constraints (11-12) determine the integer restrictions on the variables. Constrains (13) define the binary restrictions on the variables.

Table I. The Size of Proposed Model

<i>Variable</i>	<i>#Variables</i>	<i>Constraint</i>	<i>#Constraints</i>
Z	1	(2)	n^2
x_{ij}	n^2	(3)	n^2
y_{ijlm}	$2n^2(n - 1)$	(4)	$n^2(n - 1)$
		(5)	$n^2(n - 1)$
		(6)	$n^2(n - 1)$
		(7)	$n^2(n - 1)$
		(8)	n
		(9)	<i>#given digits</i>
		(10)	<i>#signs</i>
<i>Total</i>	$n^2 + 2n^2(n - 1) + 1$		$2n^2 + 4n^2(n - 1) + n + \text{\#given digits} + \text{\#signs}$

The model defined by (1)–(10) contains $[n^2 + 2n^2(n - 1) + 1]$ variables and $[2n^2 + 4n^2(n - 1) + n + (\text{\#given digits}) + (\text{\#signs})]$ linear constraints as seen in Table I.

The model has been illustrated in the puzzle shown in Figure I. The complete model has been formulated with $[4^2 + 2 \cdot 4^2(4 - 1) + 1] = 113$ variables and $[2(4)^2 + 4 \cdot 4^2(4 - 1) + 4 + 1 + 6] = 235$ linear constraints.

The optimum solution is shown in Table II. In the solution of the model, the objective function value was found to be 10 ($Z=1+2+3+4=10$).

Table II. The Solution for the Illustrative Example Problem.

Variable (x_{ij})		Value
i	j	
1	1	4
	2	2
	3	1
	4	3
2	1	3
	2	1
	3	2
	4	4
3	1	2
	2	3
	3	4
	4	1
4	1	1
	2	4
	3	3
	4	2

As seen in Table II, when we solved the puzzle by the developed model, the same result was obtained as when we solved it manually in the previous section.

III. COMPUTATIONAL RESULTS

We have analyzed the computational performance of the developed model on 36 Futoshiki instances. The mathematical model has been applied to puzzles generated from the Futoshiki website (<https://www.futoshiki.org/>). It is possible to obtain puzzles of any dimension from four to nine at this website. In addition, puzzles can be created according to their difficulty levels. There are four levels: trivial, easy, tricky, extreme. The puzzle with the chosen difficulty and dimension is created by the system. We have solved a total of 36 problems created by the system in all dimensions from 4 to 9 and at three difficulty levels (easy, difficult, and extreme). Trivial problems were not included in the study because of their simplicity. Table III summarizes the characteristics of the 36 instances.

Table III. Test Problems Description

Difficulty	Dimension (n)						Total
	4	5	6	7	8	9	
Easy	2	2	2	2	2	2	12
Tricky	2	2	2	2	2	2	12
Extreme	2	2	2	2	2	2	12
Total	6	6	6	6	6	6	36

The mathematical model has been coded in AIMMS 4.13 optimization software. CPLEX 12.5 has been used for solving the problem instances. All computations have been done on a PC with Intel Core i5-3210M CPU 2.50GHz processor and 6.00 GB RAM. We have presented detailed computational results in Table IV.

Table IV. Computational Results for Problem Instances

	Problem instances		Model size		Solution		
	n	# signs	# given digits	# int.	# const.	Z	CPU Time
EASY	4	4	1	113	233	10	0.02
	4	4	1	113	233	10	0.00
	5	8	2	226	465	15	0.02
	5	9	2	226	466	15	0.03
	6	11	3	397	812	21	0.05
	6	11	4	397	813	21	0.03
	7	18	4	638	1303	28	0.06
	7	15	7	638	1303	28	0.08
	8	30	7	961	1965	36	7.42
	8	24	8	961	1896	36	0.08
TRICKY	9	40	4	1378	2807	45	393.66
	9	46	4	1378	2812	45	15.47
	4	4	-	113	232	10	0.05
	4	5	1	113	234	10	0.00

Problem instances			Model size		Solution	
n	# signs	# given digits	# int.	# const.	Z	CPU Time
5	8	1	226	464	15	0.02
5	8	1	226	464	15	0.02
6	11	4	397	813	21	0.05
6	10	4	397	812	21	0.06
7	18	6	638	1305	28	0.17
7	19	4	638	1304	28	0.06
8	21	12	961	1961	36	0.28
8	26	6	961	1961	36	7.63
9	35	8	1378	2806	45	104.23
9	31	10	1378	2804	45	0.53
EXTREME	4	5	113	234	10	0.02
	4	4	113	234	10	0.02
	5	9	226	465	15	0.08
	5	8	226	465	15	0.02
	6	13	397	813	21	0.08
	6	11	397	812	21	0.05
	7	14	638	1302	28	0.50
	7	19	638	1305	28	0.09
	8	19	961	1953	36	1.78
	8	31	961	1964	36	11.20
	9	47	1378	2817	45	4.30
	9	23	1378	2802	45	0.22

Table IV shows problem instances according to their difficulty level. For each problem instance, the properties of the puzzle (dimension, number of signs and given digits), model sizes (number of integer variables and constraints) and solutions (objective function value, Z and CPU times in seconds) are also given in Table IV.

The objective function of the proposed model minimizes the total value of digits assigned. Due to the objective function, it was thought that the largest digit to be assigned to the puzzle would not exceed the puzzle size anyway. For this reason, a lower limit was initially determined for the digits to be assigned (constraint 2), but the upper limit was not determined. Constraint (3) has been added to the model to speed up the solution process; because without this constraint, the model could not find the optimum solutions even though it worked for two hours to solve the 7, 8 and 9-dimensional puzzles.

The dimensions and difficulty factors of the puzzle are analyzed with experimental computations. As seen on the Table IV the size of puzzle increases the solution time of model very little. In 34 of the 36 problem instances optimal solutions are found in less than 1 min. Two 9-dimension problems require more than 1 min: one easy problem with 40 signs and 4 given digits requires 393.66 sec. and one tricky problem with 35 signs and 8 given numbers requires 104.23 sec.

In the easy instances, only one of the 9-dimensional problems was solved in longer time (393.66 s) than the others. When two of the 9-dimensional problems are compared, it is seen that the digits given at the beginning of the puzzle are four for both. The four digits given in the puzzle solved in a short time are 2,8,7,1 ($r_{11} = 2, r_{59} = 8, r_{62} = 7, r_{75} = 1$) while the digits given in the puzzle solved in a long time are 6,7,5,7 ($r_{12} = 6, r_{61} = 7, r_{63} = 5, r_{79} = 7$) and the signs are 46 and 40 respectively. As a result, there is no difference between the two problems that will affect the solution time. The optimal solutions to the other problems were obtained in a much shorter time.

In tricky instances, only one of the 9-dimensional problems has been solved in longer time (104.23 s) than the others. When comparing two 9-dimensional problems, the problem in which the optimum solution is obtained in a longer time has 8 given digits ($r_{13} = 1, r_{14} = 8, r_{29} = 6, r_{42} = 6, r_{77} = 9, r_{79} = 8, r_{84} = 1, r_{86} = 8$) and the other problem has 10 given digits ($r_{11} = 7, r_{16} = 2, r_{33} = 5, r_{34} = 8, r_{35} = 6, r_{42} = 3, r_{44} = 7, r_{71} = 3, r_{87} = 4, r_{99} = 8$). Their initial signs are 35 and 31, respectively. While the number of signs of the problem, whose optimum solution is found in a longer time, is higher, the number of digits given at the beginning is less. The optimal solutions of the other problems were obtained in a much shorter time

The optimum solution of extreme instances has been obtained in a short time. Although the difficulty levels of the puzzles were determined for manual solution, it is obvious that this difficulty levels has no significant impact on the solution times.

IV. CONCLUSIONS

Puzzles have received attention in scientific studies, but there has not been much work in the literature on the solution of the Futoshiki puzzle. In this paper, we have addressed the solution of that puzzle by developing an integer programming model. We have analyzed the computational performance of the developed model on 36 instances, which we have generated from the Futoshiki website. A computational experiment has been carried out to investigate the effects of puzzle dimension and difficulty factors on the CPU times. These problems have been created to solve puzzles of three difficulty levels (easy, tricky, and extreme) and six dimensions (4 to 9). The computational results show that all of the instance problems are solved optimally in a short time.

REFERENCES

- Boreland, B., Clement, G., & Kunze, H. 2015. Set selection dynamical system neural networks with partial memories with applications to Sudoku and KenKen Puzzles. *Neural Networks*, 68: 46–51.
- Chlond, M. J. 2015. Puzzle—IP in the i. *INFORMS Transactions on Education*, 16(1): 39-41.
- Coelho, L.C., & Laporte, G. 2014. A comparison of several enumerative algorithms for Sudoku. *Journal of the Operational Research Society*, 65: 1602–1610.
- Espasa, J., Gent, I.P., Hoffmann, R., Jefferson, C., & Lynch, A.M. 2021. Using small muses to explain how to solve pen and paper puzzles. *ArXiv*, abs/2104.15040.
- Haraguchi, K. 2013. The number of inequality signs in the design of Futoshiki Puzzle. *Journal of Information Processing*, 21(1): 26-32.
- Haraguchi, K., & Ono, H. 2015. How simple algorithms can solve Latinsquare completion-type puzzles approximately. *Journal of Information Processing*, 23(3): 276–283.
- 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: 2934-2947.
- Keçeci, B. 2021. A mixed integer programming formulation for Smashed Sums puzzle: Generating and solving problem instances. *Entertainment Computing*, 36: 1-8.
- Lloyd, H., Crossley, M., Sinclair, M., & Amos, M. 2021. J-POP: Japanese puzzles as optimization problems. *IEEE Transactions on Games*.
- Mahmood, A.S. 2019. Design random number generator utilizing the Futoshiki puzzle. *Journal of Information Hiding and Multimedia Signal Processing*, 10 (1): 178-186.
- Maji, A.K., Jana, S., & Pal, R.K. 2013. An algorithm for generating only desired permutations for solving Sudoku Puzzle. *Procedia Technology*, 10: 392–399.
- Piette, C., Piette, E., Stephenson, M., Soemers, D. J., & Browne, C. 2019. Ludii and XCSP: playing and solving logic puzzles. *IEEE Conference on Games (CoG)*, 1-4.
- Santos-Garcia, G., & Palomino, M. 2007. Solving Sudoku Puzzles with rewriting rules. *Electronic Notes in Theoretical Computer Science*, 176: 79–93.
- Wang, W.L., & Tang, M.H. 2014. Simulated annealing approach to solve Nonogram Puzzles with multiple solutions. *Procedia Computer Science*, 36: 541–548.
- Waziri, M.Y., Saidu, I., & Musa, H. 2010. A mathematical approach on solving Hausa Puzzles in Northern Nigeria. *Procedia Social and Behavioral Sciences*, 8: 694–699.
- Yu, H., Tang, Y., & Zong, C. 2016. Solving Odd Even Sudoku Puzzles by binary integer linear programming. *12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2226-2230.
- Futoshiki online puzzles, <https://www.futoshiki.org/>.