

Note on Algorithmic Investigations of Juosan Puzzles

Muhammad Tsaqif Ammar¹, Muhammad Arzaki², Gia Septiana Wulandari² *(leave it blank for the first submission)*

¹ Undergraduate Student, Computing Laboratory, School of Computing, Telkom University, Bandung (40257)

² Computing Laboratory, School of Computing, Telkom University, Bandung (40257)
(left it blank for first submission)



Email:author@address.com (leave it blank for the first submission)






Abstract

We investigate several algorithmic and mathematical aspects of the Juosan puzzle— a one-player pencil-and-paper puzzle introduced in 2014 and proven NP-complete in 2018. We introduce an optimized backtracking technique for solving this puzzle by considering some invalid subgrid configurations and show that this algorithm can solve an arbitrary Juosan instance of size $m \times n$ in $O(2^{mn})$ time. A C++ implementation of this algorithm successfully found the solution to all Juosan instances with no more than 300 cells in less than 15 seconds. We also discuss the special cases of Juosan puzzles of size $m \times n$ where either m or n is less than 3. We show that these types of puzzles are solvable in linear time in terms of the puzzle size and establish the upper bound for the number of solutions to the Juosan puzzle of size $1 \times n$.


Keywords: backtracking, Juosan puzzle, puzzle solver, tractable sub-problems

1. Introduction

Juosan is a puzzle developed by Nikoli, a Japanese publisher that specializes in pencil-and-paper logic puzzles. The puzzle—introduced in 2014 [1]—has been proven NP-complete in 2018 by Iwamoto and Ibusuki [2]. Later in 2020, it was shown that the 3SAT problem is polynomial-time reducible to Juosan puzzles [3]. An algorithmic investigation regarding the card-based zero-knowledge proof of Juosan puzzles is discussed in [4]. This puzzle is played on an $m \times n$ grid of cells that are divided into several rectangular territories enclosed by bold lines. The goal is to fill each cell of the grid with a  (vertical bar) or  (horizontal bar) symbol following a set of rules, namely [5]:

- 1) if a territory contains a number, then the number of either  or  symbols in it must be equal to that number; if a territory is unnumbered, then it may have any number of  or  symbols;
- 2) the symbol  can extend for more than three cells vertically but not more than two cells

horizontally;

- 3) the symbol  can extend for more than three cells horizontally but not more than two cells vertically.

Puzzles have a long history and are found in diverse societies, revealing the inherent connection between human intelligence and playful imagination. They serve not only as recreational activities but also as cognitively stimulating exercises [6]. Interestingly, puzzles have gained significant attention from the scientific community in recent decades due to their deep connection with important problems in mathematics and computation [7]. Studies regarding this include algorithmic investigations and computational complexity analysis of various puzzles (see [7–10] for extensive surveys). In the case of pencil-and-paper puzzles, many have been proven to be NP-complete, such as (in chronological order): Nonogram (1996) [11], Sudoku (2003) [12], Nurikabe (2004) [13], Heyawake (2007) [14], Country Road and Yajilin (2012) [15], Kurodoko (2012) [16], Sto-Stone (2018) [17], Usowan (2018) [18], Kurotto and Juosan (2018 and 2020) [2, 3], Tatamibari (2020)

[19], Yin-Yang (2021) [20], Nurimeizu (2022) [21], and Tilepaint (2022) [22]. Furthermore, elementary algorithmic investigations have also been conducted on puzzles like Yin Yang and Tatamibari [23, 24].

The NP-completeness of Juosan puzzles implies the existence of a polynomial-time solution verifier and an exponential-time algorithm for solving such puzzles. However, there has been limited formal algorithmic investigation of Juosan solvers because the puzzle is relatively new and has only been recently proven NP-complete. Regardless of various approaches for solving NP-complete puzzles, such as using MIP (mixed integer programming) solvers [25] or SAT-based solvers [26], this paper discusses a relatively elementary and straightforward technique: the backtracking method with pruning optimizations. We demonstrate that this technique can solve an arbitrary Juosan puzzle in exponential time relative to the puzzle's size. We also discuss some special tractable cases of the Juosan puzzle and pertinent mathematical analyses regarding the number of solutions. Investigation of tractable sub-problems and tractable variants of NP-complete problems are particularly important in theoretical computer science (see, e.g., [27, 28]). Moreover, counting the number of solutions to a particular computational problem is also interesting from mathematical and computational perspectives, especially in counting complexity theory [29, 30].

We further organize the rest of the paper into the following sections. In Section 2, we discuss some theoretical aspects of the Juosan puzzles and derive an additional rule concerning the non-existence of some particular subgrids. In Section 3, we introduce an $O(mn)$ time algorithm to verify an $m \times n$ Juosan solution. Section 4 discusses our optimized backtracking algorithm for solving an arbitrary $m \times n$ Juosan puzzle in $O(2^{mn})$ time. The investigation of tractable variants of the Juosan puzzle and the upper bound on their number of solutions based on mathematical analysis is discussed in Section 5. Additionally, we present computational experiments of our algorithms in Section 6. The last portion of the paper, Section 7, summarizes and concludes our findings.







2. Preliminaries, Related Works, and Important Observation

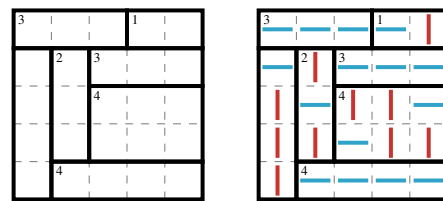
In this paper, we use one-based indexing for all arrays. For a one-dimensional array A of length n , the notation A_i or $A[i]$ ($1 \leq i \leq n$) denotes the i -th entry of A . For a two-dimensional array B of m rows and n columns, the notation $B_{i,j}$ denotes

the entry in i -th row and j -th column of B where $1 \leq i \leq m$ and $1 \leq j \leq n$.

2.1. Formal Definition and Data Structure Representation of Juosan Puzzles

Informally, a Juosan *instance* is the initial incomplete puzzle. A Juosan *configuration* is a puzzle where all cells are filled, but it may not satisfy the rules of the Juosan puzzle. A Juosan *solution* is a Juosan configuration that complies with the rules of the Juosan puzzle. To discuss the formal algorithmic aspects of the Juosan puzzle, we introduce the formal definition of Juosan instance, configuration, and solution in Definition 1. For more illustration about the Juosan puzzle, see Figure 1.

Definition 1. An *instance* of a Juosan puzzle of size $m \times n$ is a rectangular grid of m rows and n columns consisting of mn empty cells partitioned into one or more rectangular territories. A territory may contain a positive integer between 1 and the number of cells in such a territory. A *configuration* for a Juosan instance (or a Juosan configuration) is a Juosan instance whose cells are filled with either the  or  symbol. A *solution* to a Juosan instance is a Juosan configuration that satisfies the rule of Juosan puzzles, namely: (1) if a territory contains a number k , then the number of either  or  symbols in it must be equal to k ; (2) the symbol  cannot extend horizontally to more than two consecutive cells; and (3) the symbol  cannot extend vertically to more than two consecutive cells.



(a) An instance of a Juosan puzzle. (b) One possible solution for an instance in Figure 1a.

Figure 1. An example of a Juosan instance (left) and its solution (right).

In this paper, a Juosan instance of size $m \times n$ is represented using a two-dimensional array R of the same size where $R_{i,j}$ denotes the label of the territory containing cell (i, j) . This label is an integer between 1 and r (inclusive), where r denotes the number of territories in the instance. We also use a one-dimensional array N of length r to represent the numbers within each territory (if such numbers

exist). For each $1 \leq i \leq r$, we define N_i as the number occurring in territory i . If there is no information about this number for a particular territory i , we define $N_i = -1$. The solution to a Juosan instance R of size $m \times n$ is represented using a two-dimensional array S of the same size where $S_{i,j}$ is either \square or \blacksquare .

2.2. Overview of the NP-Completeness of Juosan Puzzles

Juosan puzzles have been proven NP-complete by Iwamoto and Ibusuki [2, 3]. The authors presented a polynomial-time reduction from the PLANAR 3SAT problem to Juosan puzzles [3]. The PLANAR 3SAT problem was proven NP-complete by Lichtenstein [31], thus establishing the NP-completeness of Juosan puzzles.

From every 3SAT problem, an incidence graph can be constructed by creating nodes for each variable and clause, and edges connecting each variable to the clause it appears in. The PLANAR 3SAT problem is a subset of 3SAT in which the constructed incidence graph is planar [31]. A graph is considered planar if it can be drawn on a plane such that no two edges intersect except on its vertices [32].

The earlier stated polynomial-time reduction presented by Iwamoto and Ibusuki uses constant-size partial instances of Juosan puzzles, generally recognized as *gadgets*, to represent objects in the PLANAR 3SAT problem, such as variable gadgets, clause gadgets, and not gadgets. According to Iwamoto and Ibusuki [3], any instance of the PLANAR 3SAT problem can be represented by a corresponding Juosan puzzle constructed using the aforementioned gadgets. The constructed Juosan puzzle has a solution if and only if the PLANAR 3SAT instance is satisfiable, thus completing the polynomial-time reduction. Furthermore, as Juosan puzzles can be verified in polynomial time, they are considered NP-complete. In this paper, a polynomial-time verification algorithm for checking Juosan configurations is also discussed in Section 3.

2.3. Important Observation: Invalid 2×3 and 3×2 Subgrids

In a Juosan puzzle, it is possible to have a configuration containing a 2×3 subgrid with all cells filled with \square symbols or a 3×2 subgrid with all cells filled with \blacksquare symbols. The following definition formally characterizes such subgrids. An illustration of these subgrids is given in Figure 2.

Definition 2. A 2×3 subgrid of \square is a collection of six adjacent cells (r, c) , $(r, c+1)$, $(r, c+2)$, $(r+1, c)$,

$(r+1, c+1)$, and $(r+1, c+2)$ such that these cells are filled with \square symbols. A 3×2 subgrid of \blacksquare is a collection of six adjacent cells (r, c) , $(r, c+1)$, $(r+1, c)$, $(r+1, c+1)$, $(r+2, c)$, $(r+2, c+1)$ such that these cells are filled with \blacksquare symbols.

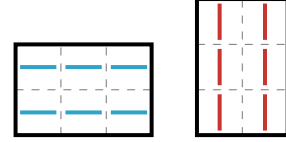


Figure 2. A 2×3 subgrid of \square and a 3×2 subgrid of \blacksquare .

In the following theorem, we prove the non-existence of Juosan solutions for particular Juosan puzzles. We show that any configuration with more than two rows containing at least one 2×3 subgrid of \square does not satisfy the puzzle rules. Analogously, this condition also applies to any configuration with more than two columns containing at least one 3×2 subgrid of \blacksquare . These facts are beneficial for backtracking algorithms in eliminating many invalid configurations as the puzzle's size gets bigger.

Theorem 1. Let C be an $m \times n$ Juosan configuration that satisfies one of the following properties:

- 1) $m > 2$ and $n \geq 3$ and C contains a 2×3 subgrid of \square ,
- 2) $m \geq 3$ and $n > 2$ and C contains a 3×2 subgrid of \blacksquare ,

then C does not satisfy Juosan rules.







Proof. We prove the theorem by contradiction. For brevity, we only discuss the proof for the first condition, i.e., if $m > 2$ and $n \geq 3$ and C contains a 2×3 subgrid of \square . If such a condition happens, then there must exist a row above or below such a subgrid, and these rows contain at least three columns. Since we cannot vertically extend the \square symbol for more than two cells, these rows must be filled with \blacksquare symbols. However, this condition implies that the \blacksquare symbols are extended for three cells horizontally, which contradicts the puzzle rules.

The proof for the second condition is analogous to the first one. \square









3. Verifying Juosan Solutions in Polynomial Time

This section briefly discusses a polynomial time algorithm for verifying whether a Juosan configuration is a valid solution.

3.1. Constraint Checking Related to Numbers Within Territories

As previously mentioned in the rules of Juosan, if a territory contains a number k , then the number of either  or  symbols in it must be equal to k . Suppose the cells are grouped into r territories, denoted as t_1, t_2, \dots, t_r , where territory t_i contains a number k_i . To verify if a Juosan configuration satisfies this rule, we first create two variables D_i and P_i for each territory to track the number of  and  symbols within it, respectively. Then, we traverse all mn cells in row-major order and increment D_i if we encounter a  symbol or increment P_i if we encounter a  symbol in territory t_i . Finally, we check if $D_i = k_i$ or $P_i = k_i$ for all territories t_i containing a positive integer k_i .

3.2. Constraint Checking Related to Three Consecutive Cells

A Juosan configuration is a solution if it does not contain  symbols extending horizontally to more than two consecutive cells or  symbols extending vertically to more than two consecutive cells. Notice that it is sufficient to check for only three consecutive cells to verify this rule. To check the horizontal constraint for the  symbol, we examine all cells (i, j) filled with  symbols, where $1 \leq i \leq m$ and $1 \leq j - 2 \leq n$, to see if two cells to the right of it are also filled with  symbols. Similarly, to check the vertical constraint for the  symbol, we examine all cells (i, j) filled with  symbols, where $1 \leq i \leq m - 2$ and $1 \leq j \leq n$, to see if two cells below it are also filled with  symbols. If one of these conditions is met for any cell, then the Juosan configuration is not a solution.












3.3. Main Verification Algorithms

To verify whether a Juosan configuration satisfies the Juosan rules, we perform all the constraint-checking algorithms described in Section 3.1 and Section 3.2. It is worth noting that each checking algorithm involves a simple linear scan of the $m \times n$ grid, making their running time $O(mn)$ each. Therefore, the entire verification algorithm also has a running time of $O(mn)$. This confirms that the Juosan puzzle can be verified in polynomial time, which places it in the NP complexity class. Additionally, this algorithm has an asymptotic space complexity of $O(mn)$ due to the need to store two variables for each territory, which, in the worst case, totals up to mn territories.

4. Backtracking Method for Solving Juosan Puzzles

The backtracking method is a recursive approach to incrementally build and test potential solutions while rejecting those that violate the problem's constraints. A backtracking algorithm can be optimized by adding extra negative constraints that might help eliminate (or "prune") many invalid solutions [33]. This section discusses a backtracking method with some pruning optimizations to solve the puzzle and its complexity analysis.

To solve the Juosan puzzle with the backtracking method, we begin by labeling the territories with unique numbers. Let us denote $R_{i,j}$ as the label of the territory containing cell (i, j) and N_i as the number occurring in territory i . If a particular territory i does not contain a number, we define $N_i = -1$. The backtracking algorithm, with row-major order grid traversal, works as follows:

- 1) We fill the grid in row-major order. To fill each cell, we try both possibilities: filling it with either  or  symbol.
- 2) As we fill in the grid, we also track and update some information to help us decide when to backtrack later. When filling cell (i, j) , four types of information are tracked:
 - a) The number of vertically consecutive  symbols ending at cell (i, j) , denoted as $VH_{i,j}$. To track this, we set $VH_{i,j}$ to $VH_{i-1,j} + 1$ if cell (i, j) is currently filled with . When backtracking, we reset $VH_{i,j}$ to 0.
 - b) The number of horizontally consecutive  symbols ending at cell (i, j) , denoted as $HV_{i,j}$. To track this, we set $HV_{i,j}$ to $HV_{i,j-1} + 1$ if cell (i, j) is currently filled with . When backtracking, we reset $HV_{i,j}$ to 0.
 - c) The number of horizontally consecutive  symbols ending at cell (i, j) , denoted as $HH_{i,j}$. To track this, we set $HH_{i,j}$ to $HH_{i,j-1} + 1$ if cell (i, j) is currently filled with . When backtracking, we reset $HH_{i,j}$ to 0.
 - d) The number of  and  symbols filled in territory $R_{i,j}$ so far. We denote the former with $D[R_{i,j}]$ and the latter with $P[R_{i,j}]$. If cell (i, j) is currently filled with  , we increment $D[R_{i,j}]$; otherwise, we increment $P[R_{i,j}]$. When backtracking, we decrement them correspondingly.

Initially, all of these variables have the value of 0 in all positions (i, j) .

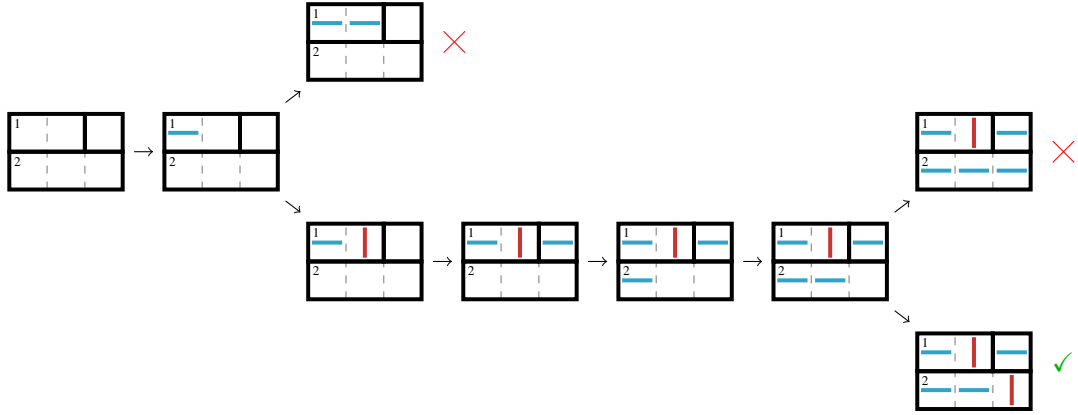


Figure 3. The pruned state space tree generated when the Algorithm 4 is used to solve a Juosan instance of size 2×3 (far left). Grid states that cannot possibly lead to a solution (i.e., meeting one of the conditions in step 3) are pruned and marked with red crosses, while the found solution is marked with a green check mark.

- 3) After filling each cell (i, j) , we decide whether to backtrack, that is, to undo the filling of the last cell and try another possibility if one exists. If not, we backtrack further. We backtrack if any of the following conditions are met:
 - a) We have formed three consecutive \square 's vertically (i.e., $VH_{i,j} = 3$).
 - b) We have formed three consecutive \blacksquare 's horizontally (i.e., $HV_{i,j} = 3$).
 - c) The grid size is larger than 2×3 and a 2×3 subgrid of \square is formed (i.e., $HH_{i,j} \geq 3$ and $HH_{i-1,j} \geq 3$), which results in an invalid grid state as explained in Theorem 1.¹
 - d) Both the number of \square and \blacksquare symbols in $R_{i,j}$ have exceeded $N_{R_{i,j}}$ (i.e., $\min\{D[R_{i,j}], P[R_{i,j}]\} > N_{R_{i,j}}$).
 - e) If we fill in the remaining empty cells in territory $R_{i,j}$ with all \square symbols or \blacksquare symbols, we can never reach $N_{R_{i,j}}$ (i.e., $E[R_{i,j}] + \max\{D[R_{i,j}], P[R_{i,j}]\} < N_{R_{i,j}}$, where $E[R_{i,j}]$ denotes the number of empty cells in $R_{i,j}$).
 - f) In territory $R_{i,j}$, the symbol with the higher frequency, denoted by c , exceeds $N_{R_{i,j}}$ while the number of cells that are not filled with c (including empty cells) is less than $N_{R_{i,j}}$.

Note that we only check for the first three conditions if $N_{R_{i,j}} = -1$ (i.e., the territory $R_{i,j}$ does not contain a number).
- 4) A solution has been found if all cells are

successfully filled. Thus, we can output this solution and terminate the algorithm.

- 5) If there is no more possibility to try and the algorithm has not yet terminated from step 4, then there is no solution for the instance; therefore, we terminate the algorithm.

We divide the backtracking approach into three major procedures, namely: the procedure $\text{FILL}(i, j, s)$ to fill the cell (i, j) with a symbol $s \in \{\square, \blacksquare\}$, the procedure $\text{UNFILL}(i, j)$ to undo the filling of the cell (i, j) , and the procedure $\text{MUST-BACKTRACK}(i, j)$ which returns true if we must backtrack considering the information at cell (i, j) . All these procedures consider the variables *current* and *size* correspondingly denotes the grid state during the filling process and an array signifying the sizes of the territories. Additionally, all these procedures consider the variables VH , HV , and HH as two-dimensional integer arrays of size $m \times n$ and the variables D and P as one-dimensional integer arrays whose lengths are equal to the number of territories.

We summarize the procedure to update the value of VH , HV , HH , D , and P in Algorithm 1. This procedure fills cell (i, j) with either a \square or \blacksquare and updates the value of $VH_{i,j}$, $HV_{i,j}$, $HH_{i,j}$, $D[R_{i,j}]$, and $P[R_{i,j}]$ accordingly.²

The process to undo the backtracking steps is summarized in Algorithm 2. This procedure removes the filling of cell (i, j) and resets the information update in each of the variables $VH_{i,j}$, $HV_{i,j}$, $HH_{i,j}$, $D[R_{i,j}]$, and $P[R_{i,j}]$ accordingly.

¹For row-major order grid traversal, we can omit checking the existence of a 3×2 subgrid of \blacksquare because if one is formed, we must immediately backtrack in the next cell to fill (i.e., besides the bottom right corner of the subgrid). This is because filling it with either symbol violates the rule regarding three consecutive symbols. Therefore, the check is unnecessary.

²We use the notation $D[R_{i,j}]$ and $P[R_{i,j}]$ instead of $D_{R_{i,j}}$ and $P_{R_{i,j}}$ to increase readability and to avoid cumbersome notations.

Algorithm 1 $\text{FILL}(i, j, s)$ fills cell (i, j) with symbol s and updates VH, HV, HH, D , and P .

Require: The cell to be updated, (i, j) , and symbol s to fill the cell with, where $s \in \{\blacksquare, \blacksquare\}$.

Ensure: The procedure fills cell (i, j) in *current* with s and updates the information VH, HV, HH, D , and P accordingly.

```

1:  $\text{current}_{i,j} \leftarrow s$ 
2: if  $s = \blacksquare$  then
3:    $VH_{i,j} \leftarrow VH_{i-1,j} + 1$ 
4:    $HH_{i,j} \leftarrow HH_{i,j-1} + 1$ 
5:    $D[R_{i,j}] \leftarrow D[R_{i,j}] + 1$ 
6: else
7:    $HV_{i,j} \leftarrow HV_{i,j-1} + 1$ 
8:    $P[R_{i,j}] \leftarrow P[R_{i,j}] + 1$ 
9: end if
```

Algorithm 2 $\text{UNFILL}(i, j)$ undo the filling of cell (i, j) for backtracking.

Require: The cell to be undone, (i, j) .

Ensure: The procedure removes the filling of cell (i, j) in *current* and updates the information VH, HV, HH, D , and P accordingly.

```

1:  $\text{assert}(\text{current}_{i,j} \in \{\blacksquare, \blacksquare\})$ 
2: if  $\text{current}_{i,j} = \blacksquare$  then
3:    $VH_{i,j} \leftarrow 0$ 
4:    $HH_{i,j} \leftarrow 0$ 
5:    $D[R_{i,j}] \leftarrow D[R_{i,j}] - 1$ 
6: else
7:    $HV_{i,j} \leftarrow 0$ 
8:    $P[R_{i,j}] \leftarrow P[R_{i,j}] - 1$ 
9: end if
10:  $\text{current}_{i,j} \leftarrow \text{empty}$ 
11:  $\triangleright$  the cell  $(i, j)$  becomes empty
```

Algorithm 3 $\text{MUSTBACKTRACK}(i, j)$ returns true if we must backtrack considering the information at (i, j) .

Require: The cell to be checked, (i, j) .

Ensure: The function returns true if the information at cell (i, j) indicates a violation of Juosan rules or if a backtracking condition is met.

```

1:  $\text{conditionA} \leftarrow VH_{i,j} = 3$   $\triangleright$  there are three consecutive  $\blacksquare$  symbols vertically
2:  $\text{conditionB} \leftarrow HV_{i,j} = 3$   $\triangleright$  there are three consecutive  $\blacksquare$  symbols horizontally
3:  $\text{conditionC} \leftarrow HH_{i,j} \geq 3$  and  $HH_{i-1,j} \geq 3$   $\triangleright$  condition in Theorem 1
4: if  $\text{conditionA}$  or  $\text{conditionB}$  or  $\text{conditionC}$  then
5:   return true
6: end if
7: if  $N_{R_{i,j}} \neq -1$  then
8:    $E \leftarrow \text{size}[R_{i,j}] - (D[R_{i,j}] + P[R_{i,j}])$   $\triangleright$  the number of empty cells in territory  $R_{i,j}$ 
9:    $\text{rem} \leftarrow \text{size}[R_{i,j}] - \max\{D[R_{i,j}], P[R_{i,j}]\}$ 
10:   $\triangleright$  the number of cells in  $R_{i,j}$  that are not filled with a symbol of the higher frequency
11:   $\text{conditionD} \leftarrow \min\{D[R_{i,j}], P[R_{i,j}]\} > N_{R_{i,j}}$ 
12:   $\text{conditionE} \leftarrow E + \max\{D[R_{i,j}], P[R_{i,j}]\} < N_{R_{i,j}}$ 
13:   $\text{conditionF} \leftarrow \max\{D[R_{i,j}], P[R_{i,j}]\} > N_{R_{i,j}}$  and  $\text{rem} < N_{R_{i,j}}$ 
14:  if  $\text{conditionD}$  or  $\text{conditionE}$  or  $\text{conditionF}$  then
15:    return true
16:  end if
17: end if
18: return false
```

Finally, the procedure $\text{MUSTBACKTRACK}(i, j)$ returns true if we must backtrack based on the current information in cell (i, j) . This algorithm returns true if the aforementioned conditions in point 3a-3f are satisfied, and it is summarized in Algorithm 3.

All of the Algorithm 1, Algorithm 2, and Algorithm 3 are used as subroutines in Algorithm 4 as the main entry point of the backtracking algorithm for searching the solution recursively. The invocation of $\text{SEARCH}(1, 1)$ in Algorithm 4 initiates the process of the backtracking approach starting at cell $(1, 1)$.

Figure 3 shows a visualization of this algorithm in a 2×3 grid with three territories (one of them is an unnumbered territory). The analysis of the asymptotic complexity of our proposed backtracking algorithm is discussed in the following theorem.

Theorem 2. *The running time of Algorithm 4 for solving any Juosan instance of size $m \times n$ has an asymptotic upper bound of $O(2^{mn})$.*

Proof. In the worst-case scenario, the algorithm must examine every possible path in its state space

Algorithm 4 SEARCH(i, j) searches for the solutions recursively (i.e., with the backtracking algorithm). Invoking SEARCH(1,1) initiates the process of filling the grid, starting at cell (1, 1).

Require: A cell (i, j) in an $m \times n$ grid of a Juosan puzzle.

Ensure: The procedure recursively searches for a solution (if any).

```

1: if  $i \leq m$  then
2:   for all  $s \in \{\text{blue}, \text{red}\}$  do
3:     FILL( $i, j, s$ )
4:     if not MUSTBACKTRACK( $i, j$ ) then
5:       if  $j + 1 < n$  then
6:         ( $nextCellRow, nextCellCol$ )  $\leftarrow (i, j + 1)$ 
7:       else
8:         ( $nextCellRow, nextCellCol$ )  $\leftarrow (i + 1, 1)$ 
9:       end if
10:      SEARCH( $nextCellRow, nextCellCol$ )
11:    end if
12:    UNFILL( $i, j$ )
13:  end for
14: else
15:    $S \leftarrow current$ 
16:   terminate the algorithm
17: end if
18: if ( $i, j$ ) = (1, 1) then
19:   output("No solution")
20: end if

```

▷ all cells in *current* are filled
▷ set *current* as the solution array *S*

▷ backtracking from (1, 1) as all possibilities have been explored

tree. Since each cell can be filled by one of two symbols, each tree level expands by a maximum of two possibilities starting from the initial grid state. This implies that there are at most 2^i grid states at level i of the state space tree. Given that a solution means that all mn cells must be filled, the depth of the state space tree is at most mn . Thus, the maximum number of grid states is bounded by $1 + 2 + 2^2 + \dots + 2^{mn} = 2^{mn+1} - 1$ which is $O(2^{mn})$. For each grid state, only constant-time operations are performed, including the use of Algorithm 1, Algorithm 2, and Algorithm 3. As a result, the asymptotic upper bound for the running time of this algorithm is $O(2^{mn})$. \square

Furthermore, Algorithm 4 requires several additional grids of size mn to store information, resulting in an asymptotic space complexity of $O(mn)$.

5. Tractability of Particular Juosan Puzzles

Some NP-complete problems may have subproblems or special cases that belong to the P class. For example, a general 3-SAT problem is NP-complete, but 2-SAT problem can be solved in linear time (thus polynomial time) through implication graph [34]. Similarly, although the general Nono-gram puzzle is NP-complete, a certain subclass of

the problem, in which each row or column has only one block of connected cells, can be converted to 2-SAT and solved in polynomial time [35]. Finally, the Yin-Yang puzzle is also an NP-complete problem, but the puzzle with sizes $1 \times n$, $2 \times n$, $m \times 1$, or $m \times 2$ can be solved in polynomial time, including finding all possible solutions [23].

In this section, we show that Juosan puzzles of sizes $1 \times n$, $2 \times n$, $m \times 1$, and $m \times 2$ for any positive integers m and n are solvable in linear time. Since Juosan puzzles of sizes $m \times 1$ and $m \times 2$ can be correspondingly transformed into puzzles of sizes $1 \times m$ and $2 \times m$, we only investigate the case of $1 \times n$ and $2 \times n$ puzzles.

5.1. Tractability of $1 \times n$ Juosan Puzzles

A $1 \times n$ Juosan puzzle is a Juosan puzzle with exactly 1 row and n columns. Here, an instance of this puzzle is a collection of n cells grouped into one or more territories, where territory is a contiguous collection of cells. A $1 \times n$ Juosan instance may contain r territories where $1 \leq r \leq n$. A solution to this instance is obtained by filling each cell with either blue or red symbols without making three consecutive red symbols. A territory may contain a positive integer k where k is between 1 and the number of cells in such a territory. If a territory contains a positive integer k , then k must equal the

number of either \square symbols or \blacksquare symbols in this territory. Moreover, if a territory does not contain k , then the number of \square and \blacksquare symbols can be any number. In this case, we can fill a territory with an alternating configuration of \square and \blacksquare symbols. In other words, if k is not defined for a particular territory, every two adjacent cells in this territory are filled with different symbols. We formally define the alternating configuration of a $1 \times n$ Juosan instance in Definition 3.

Definition 3. Suppose we consider a $1 \times n$ Juosan instance. An alternating configuration of this instance is obtained by filling the first cell with \square , the second cell with \blacksquare , and so on alternatingly until the last cell. That is, the odd-indexed cell is filled with \square while the even-indexed cell is filled with \blacksquare .

Notice that in Definition 3, we can swap the position of \square and \blacksquare symbols. Furthermore, if a territory does not contain a number k , then the alternating territory in Definition 3 also complies with the Juosan rules. This alternating configuration is essential for constructing the solution for any Juosan instance of size $1 \times n$. We discuss the general construction of the solution to any $1 \times n$ Juosan puzzle in the proof of the following theorem.

Theorem 3. Any instance of $1 \times n$ Juosan puzzles can be solved in $O(n)$ time.

Proof. Suppose we consider a $1 \times n$ Juosan instance where the cells are grouped into r territories t_1, t_2, \dots, t_r . Each territory may or may not contain a positive integer k_i . If a territory t_i does not contain a positive integer k_i , then we simply fill t_i with alternating symbols of \square and \blacksquare in its cells as described in Definition 3. Such a process clearly can be done in linear time in terms of the number of cells in t_i . If a territory t_i contains a positive integer k_i , then we firstly fill each cell in t_i with alternating symbols of \square and \blacksquare as in Definition 3. Now, let us respectively denote D_i and P_i as the number of \square symbols and \blacksquare symbols occurring in t_i . Initially, we have $D_i = \lceil n_i/2 \rceil$ and $P_i = \lfloor n_i/2 \rfloor$ where n_i is the number of cells in t_i . Observe that replacing some \blacksquare symbols with \square symbols decreases the value of D_i while simultaneously increases the value of P_i . Our goal is to set $k_i = D_i$ or $k_i = P_i$. If either $k_i = D_i$ or $k_i = P_i$, then we are done. Otherwise, we perform the following steps:

- 1) If $k_i > D_i$, then we change $k_i - D_i$ symbols of \blacksquare with \square symbols so that $k_i = D_i$ and there are no three consecutive \blacksquare symbols. That is, we increase the value of D_i to exactly k_i by replacing some \blacksquare symbols with \square symbols.

Observe that the number of \square symbols after the replacement is $D_i + (k_i - D_i) = k_i$.

- 2) If $k_i < D_i$, then we change $P_i - k_i$ symbols of \blacksquare with \square symbols so that $k_i = P_i$ and there are no three consecutive \blacksquare symbols. That is, we decrease the value of P_i to exactly k_i by replacing some \blacksquare symbols with \square symbols. Observe that the number of \square symbols after this replacement is $D_i + (P_i - k_i) = n_i - k_i$, hence the number of \blacksquare symbols after this replacement is $n_i - (n_i - k_i) = k_i$.

Notice that the replacement takes linear time in terms of the number of cells in t_i for any territory $t_i \in \{t_1, t_2, \dots, t_r\}$. As a result, the replacement for the cells in a $1 \times n$ Juosan instance takes at most $O(n)$ time. \square

The steps in the proof of Theorem 3 can be modified for constructing the solution of any $m \times 1$ Juosan puzzle in $O(m)$ by swapping the \square symbols with the \blacksquare symbols. We illustrate the construction steps for a 1×7 Juosan puzzle in the following example.

Example 1. Consider a Juosan instance of size 1×7 with only one territory containing a positive integer k . We obtain a solution for such an instance by first filling each cell with alternating \square and \blacksquare symbols as in Definition 3. At this point, we have $D = \lceil 7/2 \rceil = 4$ symbols of \square and $P = \lfloor 7/2 \rfloor = 3$ symbols of \blacksquare . If $k = 6$, then $k > D$, which means we replace $k - D = 6 - 4 = 2$ symbols of \blacksquare with \square symbols resulting in D increased to exactly k . Similarly, if $k = 2$, then $k < D$, which means we replace $P - k = 3 - 2 = 1$ symbol of \blacksquare with \square symbol resulting in P decreased to exactly k . See Figure 4 for more illustration.

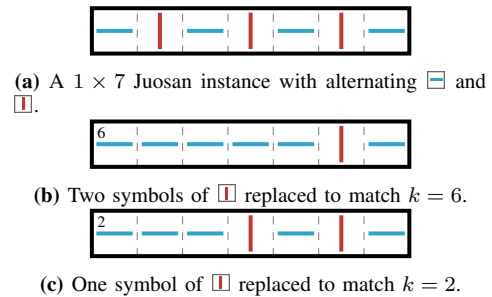


Figure 4. Solving a 1×7 Juosan instance containing a positive integer $k = 6$ or $k = 2$.

The method described in the proof of Theorem 3 gives us a linear time algorithm for solving any $1 \times n$ Juosan instance. However, this algorithm does not recover all possible solutions to a Juosan instance of

size $1 \times n$. In some puzzles, such as Yin-Yang puzzles of size $1 \times n$ and $2 \times n$, the number of solutions to these puzzles is bounded by $O(n)$ (see [23, Theorem 2 and Theorem 3]), and thus discovering all solutions to these instances can be done in polynomial time. Nevertheless, in the subsequent analysis, we shall show that the number of solutions to any arbitrary Juosan instance of size $1 \times n$ is exponential in terms of n . This is unsurprising because some easy decision problems have exponential time results for their corresponding counting problems [30].

In the subsequent analysis, suppose we consider a Juosan instance of size $1 \times n$ divided into r territories t_1, t_2, \dots, t_r . Each territory may or may not contain a number k_i signifying the number of either \square or \blacksquare symbols within such territory. Firstly, we discuss the number of possible Juosan solutions in a territory that does not contain a number. We observe the following lemma.

Lemma 1. *Suppose we consider a territory t_i containing $p \leq n$ cells within a $1 \times n$ Juosan puzzle. Suppose $s(p)$ describes the number of solutions to such a territory. If this territory does not contain a number, then $s(p)$ satisfies the recurrence $s(p) = s(p-1) + s(p-2) + s(p-3)$, with initial conditions $s(1) = 2$, $s(2) = 4$, and $s(3) = 7$.*

Proof. Since this territory does not contain a number, the number of \square and \blacksquare can be any number, so long as there are no three consecutive \blacksquare symbols. Thus, $s(p)$ can be considered as the number of strings of length p whose characters are taken from the set $\{\square, \blacksquare\}$ that contains no three consecutive \blacksquare symbols. Clearly $s(1) = 2$, $s(2) = 4$, and $s(3) = 7$. Moreover, such strings have three types of endings, i.e., either they end with \square , $\square\blacksquare$, or $\blacksquare\blacksquare$. In other words, such strings have the form of $\alpha\square$, $\alpha\square\blacksquare$, or $\alpha\blacksquare\blacksquare$ where α is again a string whose characters are taken from the set $\{\square, \blacksquare\}$ that contains no three consecutive \blacksquare symbols of length $p-1$, $p-2$, and $p-3$, respectively. Any such α results in a valid string. Thus, we have a one-to-one correspondence since the three cases are disjoint, and the result follows. \square

The exact solution of the recurrence relation in Lemma 1 is discussed in the following lemma.

Lemma 2. *The exact solution of the recurrence relation in Lemma 1 is $s(p) = \left\lceil \frac{3\delta \left(\frac{1}{3}(\gamma_+ + \gamma_- + 1)\right)^p}{\delta^2 - 2\delta + 4} \right\rceil$ where $\gamma_{\pm} = \sqrt[3]{19 \pm 3\sqrt{33}}$ and $\delta = \sqrt[3]{586 + 102\sqrt{33}}$. Here, the function $\lceil \cdot \rceil$ denotes the nearest integer rounding function.*

Proof. Notice that the sequence $s(p)$ is similar to the tribonacci sequence defined by $t(1) = 1$, $t(2) = 1$, $t(3) = 2$, and the recurrence relation $t(p) = t(p-1) + t(p-2) + t(p-3)$ for $p \geq 4$ [36]. The difference between the two is their initial conditions, where $s(1) = 2$, $s(2) = 4$, and $s(3) = 7$, resulting in $s(p)$ being equivalent to the tribonacci sequence, but shifted forward by two terms. In other words, $s(p) = t(p+2)$. According to [37], the explicit formula for the tribonacci sequence is $t(p) = \left\lceil \frac{3\delta \left(\frac{1}{3}(\gamma_+ + \gamma_- + 1)\right)^p}{\delta^2 - 2\delta + 4} \right\rceil$ where $\gamma_{\pm} = \sqrt[3]{19 \pm 3\sqrt{33}}$ and $\delta = \sqrt[3]{586 + 102\sqrt{33}}$. Since $s(p) = t(p+2)$, the theorem is proven. \square

Secondly, we consider the case of a territory that contains a number. We observe the following lemma.

Lemma 3. *Suppose we consider a territory t_i containing p cells within a $1 \times n$ Juosan puzzle that contains a number k . Suppose $s(p, k)$ describes the number of solutions to such a territory. Then, we have the following:*

$$s(p, k) = \begin{cases} F(p/2, p/2), & \text{if } p \text{ is even and } k = p/2 \\ F(k, p-k) + F(p-k, k), & \text{otherwise} \end{cases}$$

where $F(a, b)$ denotes the number of solutions to a territory containing $a + b$ cells with exactly a cells filled with \blacksquare and b cells filled with \square without three consecutive \blacksquare symbols. Here, $F(a, b)$ follows the recurrence relation:

$$F(a, b) = \begin{cases} 0 & \text{if } b = 0, a > 2 \\ 1 & \text{if } a = 0 \text{ or } (a, b) = (2, 0) \\ b + 1 & \text{if } a = 1 \\ F(a, b-1) & \\ + F(a-1, b-1) & \\ + F(a-2, b-1) & \text{if } a \geq 2, b \geq 1 \end{cases} \quad (1)$$

Proof. The proof of this lemma is similar to the proof of Lemma 1. Since a territory of size p containing a number k must either have exactly k number of \blacksquare or \square symbols in it, then $s(p, k) = F(k, p-k) + F(p-k, k)$. However, if p is even and $k = p/2$, then $s(p, k) = F(p/2, p/2)$ as $k = p-k = p/2$ and to avoid over-counting. Furthermore, by definition, the value $F(a, b)$ is equal to the number of strings of length $a + b$ whose characters are taken from the set $\{\square, \blacksquare\}$ that consist of exactly a characters of \blacksquare and b characters of \square without three consecutive \blacksquare symbols. Clearly, $F(a, b)$ holds for the aforementioned initial conditions. Moreover, such strings with no three consecutive \blacksquare characters also have the form of $\alpha\square$, $\alpha\square\blacksquare$, or $\alpha\blacksquare\blacksquare$ where α is again a string whose characters are taken from

the set $\{\square, \blacksquare\}$ that contains no three consecutive \blacksquare symbols. Each type of ending corresponds to a pattern of extra symbols of \blacksquare and \square following α . The extra symbols for the ending types $\alpha\square$, $\alpha\square\square$, and $\alpha\square\blacksquare\blacksquare$ correspond respectively to the relations of $F(a, b)$ with $F(a, b-1)$, $F(a-1, b-1)$, and $F(a-2, b-1)$. Thus, we have a one-to-one correspondence since the three cases are disjoint, and the result follows \square

In the following analysis, we use bivariate generating function methods to find the explicit solution of the recurrence relation (1) as in [32]. The following lemma discusses the derivation of this generating function.

Lemma 4. *The corresponding (ordinary) bivariate generating function for the recurrence relation $F(a, b)$ in (1) is*

$$\sum_{a \geq 0, b \geq 0} F(a, b) x^a y^b = \frac{x^2 + x + 1}{1 - y(x^2 + x + 1)}$$

Proof. Let us define an ordinary bivariate generating function $G(x, y) = \sum_{a \geq 0, b \geq 0} F(a, b) x^a y^b$. Based on $F(a, b)$ for the general case $a \geq 2, b \geq 1$, we have:

$$\begin{aligned} & \sum_{a \geq 2, b \geq 1} F(a, b) x^a y^b \\ &= \sum_{a \geq 2, b \geq 1} F(a-2, b-1) x^a y^b \\ &+ \sum_{a \geq 2, b \geq 1} F(a-1, b-1) x^a y^b \\ &+ \sum_{a \geq 2, b \geq 1} F(a, b-1) x^a y^b \end{aligned} \quad (2)$$

In (2), we have four summands (three of them are on the right-hand side of the equal sign). Let us break down each one in the following equations:

$$\begin{aligned} & \sum_{a \geq 2, b \geq 1} F(a, b) x^a y^b \\ &= G(x, y) - \left(\sum_{b \geq 0} F(0, b) y^b \right) \\ &- \left(\sum_{b \geq 0} F(1, b) x y^b \right) - F(2, 0) x^2 \\ &= G(x, y) - \frac{1}{1-y} - \frac{x}{(1-y)^2} - x^2 \end{aligned} \quad (3)$$

$$\begin{aligned} & \sum_{a \geq 2, b \geq 1} F(a-2, b-1) x^a y^b \\ &= x^2 y \sum_{a \geq 2, b \geq 1} F(a-2, b-1) x^{a-2} y^{b-1} \\ &= x^2 y \sum_{p \geq 0, q \geq 0} F(p, q) x^p y^q \\ &= x^2 y \cdot G(x, y) \end{aligned} \quad (4)$$

$$\begin{aligned} & \sum_{a \geq 2, b \geq 1} F(a-1, b-1) x^a y^b \\ &= x y \sum_{a \geq 2, b \geq 1} F(a-1, b-1) x^{a-1} y^{b-1} \\ &= x y \sum_{p \geq 1, q \geq 0} F(p, q) x^p y^q \\ &= x y \left(G(x, y) - \sum_{q \geq 0} F(0, q) y^q \right) \\ &= x y \left(G(x, y) - \frac{1}{1-y} \right) \end{aligned} \quad (5)$$

$$\begin{aligned} & \sum_{a \geq 2, b \geq 1} F(a, b-1) x^a y^b \\ &= y \sum_{a \geq 2, b \geq 1} F(a, b-1) x^a y^{b-1} \\ &= y \sum_{p \geq 2, q \geq 0} F(p, q) x^p y^q \\ &= y \left(G(x, y) - \sum_{q \geq 0} F(0, q) y^q - \sum_{q \geq 0} F(1, q) x y^q \right) \\ &= y \left(G(x, y) - \frac{1}{1-y} - \frac{x}{(1-y)^2} \right) \end{aligned} \quad (6)$$

We substitute these results to the previous equation (2) to obtain the following expression:

$$\begin{aligned} & G(x, y) - \frac{1}{1-y} - \frac{x}{(1-y)^2} - x^2 \\ &= x^2 y \cdot G(x, y) \\ &+ x y \left(G(x, y) - \frac{1}{1-y} \right) \\ &+ y \left(G(x, y) - \frac{1}{1-y} - \frac{x}{(1-y)^2} \right) \end{aligned}$$

It is easy to obtain $G(x, y) = \frac{x^2 + x + 1}{1 - y(x^2 + x + 1)}$ with basic algebra. This completes the proof for the lemma. \square

We derive the explicit solution for $s(p, k)$ in Lemma 3 in the following lemma.

Lemma 5. Suppose we consider a territory t_i containing p cells within a $1 \times n$ Juosan puzzle that contains a number k . Suppose $s(p, k)$ describes the number of solutions to such a territory. Then, we have the following:

$$\tilde{F}(a, b) = \sum_{j=0}^{\min\{b+1, \lfloor a/2 \rfloor\}} \binom{b+1}{j} \binom{b+1-j}{a-2j}$$

$$s(p, k) = \begin{cases} \tilde{F}(p/2, p/2), & \text{if } p \text{ is even and } k = p/2 \\ \tilde{F}(k, p-k) + \tilde{F}(p-k, k), & \text{otherwise} \end{cases}$$

Proof. We derive the explicit formula, $\tilde{F}(a, b)$, for the recurrence relation $F(a, b)$ in Lemma 3 by using the generating function in Lemma 4 and extract the coefficient from it. In other words, $\tilde{F}(a, b) = [x^a y^b] \frac{x^2 + x + 1}{1 - y(x^2 + x + 1)}$ (the notation $[x^a y^b]$ denotes the *coefficient operator* to denote the coefficient of $x^a y^b$ in a series). To extract this coefficient, [38] outlines a calculation that shows $\tilde{F}(a, b) = \sum_{j=0}^{\min\{b+1, \lfloor a/2 \rfloor\}} \binom{b+1}{j} \binom{b+1-j}{a-2j}$. Finally, the value $s(p, k)$ can be obtained analogously to the one mentioned in the proof of Lemma 3. \square

Finding the closed-form expression of $\tilde{F}(a, b)$ in Lemma 5 is particularly challenging. Instead of finding such an expression, we provide the upper bound of $s(p, k)$ based on the solution in Lemma 5 in the following Lemma 6.

Lemma 6. The asymptotic upper bound on the number of solutions for territory with p cells in a $1 \times n$ Juosan puzzle, whether numbered or unnumbered, is $O((\frac{1}{3}(\gamma_+ + \gamma_- + 1))^p)$ where $\gamma_{\pm} = \sqrt[3]{19 \pm 3\sqrt{33}}$ or roughly $O(1.8392^p)$.

Proof. This bound clearly holds for unnumbered territories, as their explicit formula (already described in Lemma 2) is simply $(\frac{1}{3}(\gamma_+ + \gamma_- + 1))^p$ multiplied by a constant. Furthermore, note that unnumbered territories do not have constraints on the number of \square 's or \blacksquare 's that can be placed within them. This implies that the number of solutions for the unnumbered case effectively sums up the number of solutions for the numbered case with $k = 1, 2, \dots, p$. Since the solutions for the numbered case form a subset of the solutions for the unnumbered case, the same upper bound applies. Thus, the proof is complete. \square

Finally, the asymptotic upper bound on the number of solutions to a $1 \times n$ Juosan puzzle is discussed in the following theorem.

Theorem 4. The asymptotic upper bound on the number of solutions to a $1 \times n$ Juosan puzzle is

$O((\frac{1}{3}(\gamma_+ + \gamma_- + 1))^n)$ where $\gamma_{\pm} = \sqrt[3]{19 \pm 3\sqrt{33}}$ or roughly $O(1.8392^n)$.

Proof. Suppose a $1 \times n$ Juosan puzzle is comprised of r territories t_1, t_2, \dots, t_r with respective sizes s_1, s_2, \dots, s_r . According to Lemma 6, the number of solutions for each territory t_i , when considering it in isolation from all other territories, is bounded by $O((\frac{1}{3}(\gamma_+ + \gamma_- + 1))^{s_i})$. To ease our analysis, let us consider a hypothetical case where we can color the territories independently. In such a scenario, the asymptotic upper bound for the number of solutions for the entire $1 \times n$ puzzle can be obtained by multiplying the upper bounds for each territory, resulting in $O((\frac{1}{3}(\gamma_+ + \gamma_- + 1))^{s_1 + \dots + s_r}) = O((\frac{1}{3}(\gamma_+ + \gamma_- + 1))^n)$. However, since filling the territories independently is impossible, the number of solutions for a $1 \times n$ Juosan puzzle must be lower. Hence, the upper bound is still valid. \square

The result in Theorem 4 provides an insight that the counting complexity related to the number of solutions for a $1 \times n$ Juosan puzzle is exponential with respect to n , although such a puzzle is solvable in linear time. Nevertheless, we suspected that finding the closed form or at least the tight bound for the expression in Lemma 5 is challenging.

5.2. Tractability of $2 \times n$ Juosan Puzzles

A $2 \times n$ Juosan puzzle is a Juosan puzzle with exactly 2 rows and n columns. An instance of this puzzle is a collection of $2n$ cells grouped into one or more rectangular territories. This instance may contain r territories where $1 \leq r \leq 2n$. As in the $1 \times n$ Juosan puzzle, the solution to an instance of $2 \times n$ Juosan puzzle is obtained by putting either \square or \blacksquare symbols to each cell without creating three consecutive \blacksquare symbols. In addition, every territory may contain a positive integer k denoting the number of either \square or \blacksquare symbols in it. If the integer k is not defined for a particular territory, then we can put any number of \square or \blacksquare symbols in such territory. Furthermore, this territory has at least one trivial solution, a *checkered-like pattern* where orthogonally adjacent cells are filled with distinct symbols. We formally define such a configuration in Definition 4.

Definition 4. Suppose we consider a $2 \times n$ Juosan puzzle. A *checkered-like configuration* of this instance is obtained as follows:

- 1) For the first row, the first cell is filled with \square , the second cell is filled with \blacksquare , and so on alternatingly until the last cell.
- 2) For the second row, the first cell is filled with \blacksquare , the second cell is filled with \square , and so on alternatingly until the last cell.

In other words, two orthogonally adjacent cells contain different symbols.

Notice that in Definition 4, we can swap the position of \blacksquare and \blacksquare symbols. Moreover, the configuration in Definition 4 complies Juosan rule if a territory does not contain a number k . As the alternating configuration in Definition 3, the checkered-like configuration in Definition 4 is important for constructing the solution to any Juosan instance of size $2 \times n$. The general construction of the solution to any $2 \times n$ Juosan puzzle is discussed in the proof of the following theorem.

Theorem 5. *Any instance of $2 \times n$ Juosan puzzle can be solved in $O(n)$ time.*

Proof. The proof of this theorem is similar to the proof of Theorem 3. One key difference is the shape of a territory is a rectangle with either one row or two rows. For a territory that does not contain a positive integer, we fill this territory with the alternating configuration as in Definition 3 if it has one row or the checkered-like configuration as in Definition 4 if it has two rows, and we are done. This process can be done in linear time in terms of the number of cells in the territory. For a territory that contains a positive integer, we perform symbol replacements analogous to those described in the proof of Theorem 3 if necessary. Notice that replacing \blacksquare symbols with \blacksquare symbols does not violate the puzzle's rule regarding the number of vertically consecutive \blacksquare symbols since we consider an instance with two rows. Consequently, the replacement for the cells in a $2 \times n$ Juosan puzzle takes at most $O(n)$ time. \square

The construction steps in the proof of Theorem 5 can be adapted for constructing the solution of any $m \times 2$ Juosan puzzle in $O(m)$ by swapping the \blacksquare symbols with the \blacksquare symbols.

6. Experimental Results

The following section delves into the experiments conducted to evaluate the running time of the proposed algorithm. The experiments were performed on a 64-bit Windows 11 system using the C++ programming language and g++ compiler of version 12.2.0. The programming language C++ was chosen as it is relatively faster than other commonly used programming languages like Java or Python [39]. The system used for the experiment also included an Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz with 4 GB of RAM. Interested readers can access the source codes, test cases, and other relevant documents related to the experiment at <https://github.com/tsaqifammar/juosan-backtracking>.

The experiment tested the C++ implementation of the backtracking algorithm described in Section 4, using test cases collected from [40]. The test cases consist of various Juosan puzzles of various dimensions, ranging from 6×6 to 30×45 , but each is guaranteed to have a unique solution. There are seventy test cases, with the majority (fifty-eight) being 10×10 in size. The goal was to determine the average running time of three runs needed for the algorithm to solve a given instance.

When running the experiment, we were limited by our computational device, which prevented us from evaluating some of the larger test cases. From the test cases in [40], we successfully solved all test cases—except those of sizes 15×24 , 25×40 , and 30×45 —in less than fifteen seconds. The running times of the algorithm for solving the test cases that were able to be evaluated are summarized in Table 1, grouped based on their sizes. The running time is measured in milliseconds up to three decimal places except for the Juosan instance of size 15×24 . Note that the algorithm's running time is not only determined by the instance's size. Other factors like the number and arrangement of territories also influence the resulting running time. Therefore, two test cases of equal size may produce significantly different outcomes, and larger test cases might have a faster running time than the smaller ones.

Table 1. The resulting running times taken (in milliseconds) for the backtracking algorithm to solve the test cases, grouped on their sizes.

Size	#	Max.	Min.	Avg.
6×6	2	0.018	0.004	0.011
10×10	58	23.407	0.036	1.214
12×12	1	8.709	8.709	8.709
10×18	1	128.912	128.912	128.912
15×15	3	13537.167	617.870	4951.601
16×16	1	2850.790	2850.790	2850.790
12×25	1	9298.723	9298.723	9298.723
15×24	1	348315755	348315755	348315755

7. Concluding Remarks

In this study, we presented an algorithm to verify a Juosan solution of size $m \times n$ in $O(mn)$ time and proposed a backtracking algorithm that has an asymptotic running time of $O(2^{mn})$ to solve an arbitrary Juosan puzzle. Despite the exponential-time upper bound, each of the test cases from [40], except those of sizes 15×24 , 25×40 , and 30×45 , were solved successfully in less than fifteen seconds.

Some NP-complete problems may have sub-problems or special cases that belong to the P class. In the case of Juosan puzzles, Theorem 3 and Theorem 5 shows that puzzles of sizes $1 \times n$,

$2 \times n$, $m \times 1$, and $m \times 2$ for any positive integers m and n are solvable in linear time, making them tractable. We also mathematically investigated the number of solutions for Juosan puzzles of size $1 \times n$, which has an upper bound of $O((\frac{1}{3}(\gamma_+ + \gamma_- + 1))^n)$ where $\gamma_{\pm} = \sqrt[3]{19 \pm 3\sqrt{33}}$ or roughly $O(1.8392^n)$ as shown in Theorem 4. This outcome is not unexpected given that some easy decision problems exhibit exponential time complexity for their corresponding counting problems [30]. Notably, our current upper bound for the number of solutions of a numbered territory within a $1 \times n$ Juosan puzzle is not very tight, as finding a closed form or a tight upper bound for the found formula is challenging. This could potentially serve as an interesting direction for future research. Furthermore, exploring the complexity class for finding all possible solutions to a Juosan instance could be an interesting route for further investigation. We conjecture that it belongs to the #P-complete class.

As a final suggestion, we propose exploring the use of SAT-based solvers to solve Juosan puzzles. The use of SAT-based solvers for solving NP-complete problems has been shown to be highly effective, such as Sudoku [26], and could be adapted to Juosan puzzles. In addition, the solver can be extended to search for all possible solutions to a Juosan puzzle, which can facilitate the discovery of new insights about the structure of Juosan puzzles and interesting puzzle variations.

References

- [1] Nikori, *Nikori no penpa*. 2015, Nov. 2014.
- [2] C. Iwamoto and T. Ibusuki, “Kurotto and Juosan are NP-complete,” in *The 21st Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCG3 2018)*, 2018, pp. 46–48. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-90048-9_14
- [3] C. Iwamoto and T. Ibusuki, “Polynomial-time reductions from 3SAT to Kurotto and Juosan puzzles,” *IEICE Transactions on Information and Systems*, vol. 103, no. 3, pp. 500–505, 2020. [Online]. Available: https://www.jstage.jst.go.jp/article/transinf/E103.D/3/E103.D_2019FCP0004/_pdf
- [4] D. Miyahara, L. Robert, P. Lafourcade, S. Takeshige, T. Mizuki, K. Shinagawa, A. Nagao, and H. Sone, “Card-based ZKP protocols for Takuzu and Juosan,” in *10th International Conference on Fun with Algorithms (FUN 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- [5] Nikoli, “Juosan - nikoli,” <https://www.nikoli.co.jp/en/puzzles/juosan/>, Nov. 2022, accessed: 2022-11-30.
- [6] M. Danesi, *An anthropology of puzzles: The role of puzzles in the origins and evolution of mind and culture*. Taylor & Francis, 2018.
- [7] R. A. Hearn and E. D. Demaine, *Games, puzzles, and computation*. CRC Press, 2009.
- [8] E. D. Demaine, “Playing games with algorithms: Algorithmic combinatorial game theory,” in *International Symposium on Mathematical Foundations of Computer Science*. Springer, 2001, pp. 18–33.
- [9] G. Kendall, A. Parkes, and K. Spoerer, “A survey of NP-complete puzzles,” *ICGA Journal*, vol. 31, no. 1, pp. 13–34, 2008.
- [10] R. Uehara, “Computational complexity of puzzles and related topics,” *Interdisciplinary Information Sciences*, pp. 1–22, 2023.
- [11] N. Ueda and T. Nagao, “NP-completeness results for Nonogram via parsimonious reductions,” Department of Computer Science, Tokyo Institute of Technology, Tech. Rep., 1996.
- [12] T. Yato and T. Seta, “Complexity and completeness of finding another solution and its application to puzzles,” *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 86, no. 5, pp. 1052–1060, 2003.
- [13] M. Holzer, A. Klein, and M. Kutrib, “On the NP-completeness of the Nurikabe pencil puzzle and variants thereof,” in *Proceedings of the 3rd International Conference on FUN with Algorithms*. Citeseer, 2004, pp. 77–89.
- [14] M. Holzer and O. Ruepp, “The troubles of interior design—a complexity analysis of the game Heyawake,” in *International Conference on Fun with Algorithms*. Springer, 2007, pp. 198–212.
- [15] A. Ishibashi, Y. Sato, and S. Iwata, “NP-completeness of two pencil puzzles: Yajilin and Country Road,” *Utilitas Mathematica*, vol. 88, pp. 237–246, 2012.
- [16] J. Kölker, “Kurodoko is NP-complete,” *Information and Media Technologies*, vol. 7, no. 3, pp. 1000–1012, 2012.
- [17] A. Allen and A. Williams, “Sto-Stone is NP-Complete,” in *CCCG*, 2018, pp. 28–34.
- [18] C. Iwamoto and M. Haruishi, “Computational complexity of Usowan puzzles,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 101, no. 9, pp. 1537–1540, 2018.
- [19] A. Adler, J. Bosboom, E. D. Demaine, M. L. Demaine, Q. C. Liu, and J. Lynch, “Tatamibari

- is NP-Complete,” in *10th International Conference on Fun with Algorithms (FUN 2021)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), M. Farach-Colton, G. Prencipe, and R. Uehara, Eds., vol. 157. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 1:1–1:24. [Online]. Available: <https://drops.dagstuhl.de/opus/volltexte/2020/12762>
- [20] E. D. Demaine, J. Lynch, M. Rudoy, and Y. Uno, “Yin-Yang Puzzles are NP-complete,” in *33rd Canadian Conference on Computational Geometry (CCCG) 2021*, 2021.
- [21] C. Iwamoto and T. Ide, “Moon-or-Sun, Nagareru, and Nurimeizu are NP-complete,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, pp. 1187–1194, 2022. [Online]. Available: https://www.jstage.jst.go.jp/article/transfun/advpub/0/advpub_2021DMP0006/_article/-char/ja/
- [22] C. Iwamoto and T. Ide, “Five Cells and Tilepaint are NP-Complete,” *IEICE Transactions on Information and Systems*, vol. 105, no. 3, pp. 508–516, 2022. [Online]. Available: https://www.jstage.jst.go.jp/article/transinf/E105.D/3/E105.D_2021FCP0001/_pdf
- [23] M. I. Putra, M. Arzaki, and G. S. Wulandari, “Solving Yin-Yang Puzzles Using Exhaustive Search and Prune-and-Search Algorithms,” *(IJCSAM) International Journal of Computing Science and Applied Mathematics*, vol. 8, no. 2, pp. 52–65, 2022.
- [24] E. C. Reinhard, M. Arzaki, and G. S. Wulandari, “Solving Tatamibari Puzzle Using Exhaustive Search Approach,” *Indonesia Journal on Computing (Indo-JC)*, vol. 7, no. 3, pp. 53–80, Dec. 2022. [Online]. Available: <https://socj.telkomuniversity.ac.id/ojs/index.php/indojc/article/view/675>
- [25] M. Banbara, K. Hashimoto, T. Horiyama, S.-i. Minato, K. Nakamura, M. Nishino, M. Sakai, R. Uehara, Y. Uno, and N. Yasuda, “Solving rep-tile by computers: Performance of solvers and analyses of solutions,” *arXiv preprint arXiv:2110.05184*, 2021.
- [26] C. Bright, J. Gerhard, I. Kotsireas, and V. Ganesh, “Effective problem solving using SAT solvers,” in *Maple Conference*. Springer, 2019, pp. 205–219.
- [27] C. Bessiere, C. Carbonnel, E. Hebrard, G. Katsirelos, and T. Walsh, “Detecting and exploiting subproblem tractability,” in *IJCAI: International Joint Conference on Artificial Intelligence*, 2013, pp. 468–474.
- [28] J. Dreier, S. Ordyniak, and S. Szeider, “CSP Beyond Tractable Constraint Languages,” in *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022.
- [29] L. G. Valiant, “The complexity of computing the permanent,” *Theoretical computer science*, vol. 8, no. 2, pp. 189–201, 1979.
- [30] A. Antonopoulos, E. Bakali, A. Chalki, A. Pagourtzis, P. Pantavos, and S. Zachos, “Completeness, approximability and exponential time results for counting problems with easy decision version,” *Theoretical Computer Science*, vol. 915, pp. 55–73, 2022.
- [31] D. Lichtenstein, “Planar Formulae and Their Uses,” *SIAM Journal on Computing*, vol. 11, no. 2, pp. 329–343, 1982. [Online]. Available: <https://doi.org/10.1137/0211025>
- [32] T. Koshy, *Discrete mathematics with applications*. Elsevier, 2004.
- [33] R. Neapolitan and K. Naimipour, *Foundations of algorithms*. Jones & Bartlett Publishers, 2010.
- [34] B. Aspvall, M. F. Plass, and R. E. Tarjan, “A linear-time algorithm for testing the truth of certain quantified boolean formulas,” *Information processing letters*, vol. 8, no. 3, pp. 121–123, 1979.
- [35] S. Brunetti and A. Daurat, “An algorithm reconstructing convex lattice sets,” *Theoretical Computer Science*, vol. 304, no. 1, pp. 35–57, 2003. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0304397503000501>
- [36] M. Feinberg, “Fibonacci-tribonacci,” *The Fibonacci Quarterly*, vol. 1, no. 1, pp. 71–74, 1963.
- [37] T. Noe, T. I. Piezas, and E. W. Weisstein, “Tribonacci number. From MathWorld—A Wolfram Web Resource,” last visited on 17/3/2023. [Online]. Available: <https://mathworld.wolfram.com/TribonacciNumber.html>
- [38] M. Scheuer, “Number of binary strings with k ones or k zeros and no three consecutive ones,” *Mathematics Stack Exchange*, last visited on 3/4/2023. [Online]. Available: <https://math.stackexchange.com/q/4617173>
- [39] L. Prechelt, “An empirical comparison of seven programming languages,” *Computer*, vol. 33, no. 10, pp. 23–29, 2000.
- [40] Otto Janko, “Juosan,” <https://www.janko.at/Raetsel/Juosan/index.htm>, Oct. 2022, accessed: 2022-10-11.