## 1. Introduction

Path puzzle is a logic puzzle introduced by Roderick Kimball, a freelance puzzle maker, in his 2013 book [1] and was featured in The *New York Times*'s wordplay blog [2]. This puzzle is proven NP-complete by Bosboom et al. in 2020 [3]. It is played on a rectangular grid of cells with two openings on the edge and constraint numbers on some of the rows and columns.[1] The solution to this puzzle is a line connecting the two openings through the grid cells with each cell can only be passed once. The number of cells passing through a specific row or column must also equal to the constraint numbers on the rows or columns. There are some modifications to the original puzzle to increase the difficulty, such as using a non-rectangular grid and more than two openings. An instance of this puzzle is illustrated in Figure 1.

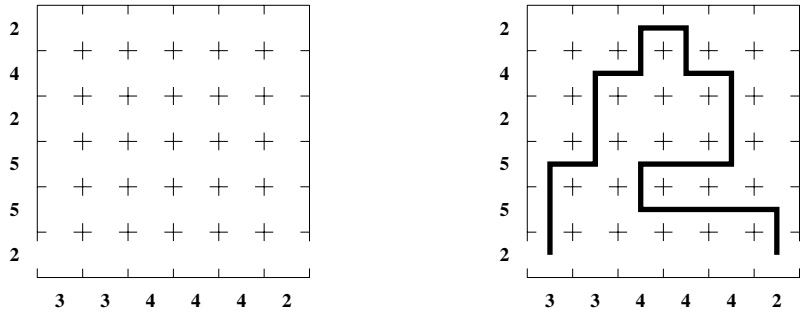


Figure 1. An instance of Path Puzzles (left) and its corresponding solution (right).

Puzzles are a form of play that can improve mood and reduce stress from everyday life, thus mainly done as a form of recreational activity [4]. In addition, puzzles also can help develop some skills, such as logical problem-solving. Many puzzles are related to important computational and combinatorial problems, thus gathering the attention of scientific communities in computing and mathematics [5–8]. There have been many puzzles proven to be NP-complete, such as Country Road (2012) [9], Hashiwokakero (2009) [10], Hiroimono (2007) [11], Juosan (2018) [12, 13], KPlumber (2004) [14], Kurotto [12, 13], Minesweeper (2000) [15], Moon-or-Sun (2022) [16], Nagareru (2022) [16], Nonogram (1996) [17], Nurimeizu (2022) [16], Nurikabe (2003) [18, 19], Shikaku (2013) [20], Slither Link (2000) [21], Sudoku (2003) [22], Suguru (2022) [23], Tatamibari (2020) [24], Tilepaint (2022) [25], Yajilin (2012) [9], Yin-Yang (2021) [26], and ZHED (2022) [27].

The NP-completeness of the Path puzzle means a polynomial time verifier algorithm exists for verifying the Path puzzle's solution. This also means an exponential time solver exists to find the solution to arbitrary Path puzzles. Nevertheless, to the authors' knowledge, no further exploration regarding the algorithms to solve these puzzles has been discussed. There are various approaches to solving NP-complete puzzles, such as using an integer programming model [28, 29], SAT solver [30], and SMT solver [24]. Some studies also have been conducted regarding the elementary technique to solve puzzles, such as an exhaustive search approach for solving the Tatamibari puzzle [31] and the prune-and-search technique for solving Yin-Yang puzzle [32].

This paper presents a method for solving a Path puzzle using the backtracking technique as a straightforward and elementary method of solving a Path puzzle. With backtracking, this paper establishes the upper bound to the time complexity of finding a single solution to the instance and demonstrates that the solution can be obtained in exponential time in terms of the puzzle's size. In addition, this final project also discusses an alternative method of solving a Path puzzle using a SAT-based approach. This involves formulating the rules of Path puzzles as Boolean satisfiability (SAT) problems and utilizing a SAT solver to find the solution. Furthermore, this paper conducts empirical performance tests on both methods and provides a comparative analysis, evaluating their effectiveness and efficiency in solving Path puzzles.

The subsequent investigation of the backtracking and SAT-based approach for solving Path puzzles, along with their related computational problem, is structured into eight parts. Section 2 discusses a comprehensive overview of the key concepts related to Path puzzles. This includes a formal definition of a Path instance, Path configuration, Path solution, an array representation of a Path puzzle, an overview of the NP-completeness of Path puzzles, and a brief introduction to propositional logic, Conjunctive Normal Form (CNF), Boolean satisfiability (SAT) problem, and SAT solver. Some findings on conditions where a Path puzzle has no solution are discussed in Section 3. Section 4 discusses an algorithm to verify whether a Path configuration is a Path solution in polynomial time. The backtracking-based algorithm for solving an arbitrary Path instance of size $m \times n$ in $O(3^{mn})$ time is discussed in Section 5. Section 6 covers the proposed encoding to translate the rules of Path puzzles to propositional formulas and an analysis of the number of variables and clauses required to encode an arbitrary $m \times n$ Path puzzle. Section

[1] See `https://www.enigami.fun` for details.

7 demonstrates that a Path puzzle whose rows and columns constraints are undefined is polynomially solvable. Section 8 discusses the experiments conducted related to the proposed methods and their results. Lastly, Section 9 gives the summary and conclusion of the paper as well as some potential future works.