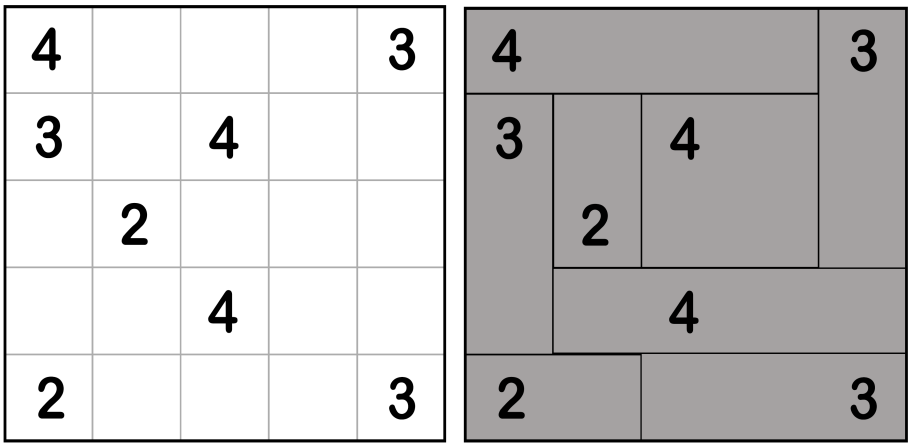
1. Pendahuluan

Shikaku merupakan salah satu permainan *logic puzzle* yang diterbitkan oleh perusahaan Nikoli asal Jepang yang juga menerbitkan permainan Sudoku [1, 2, 3]. Shikaku memiliki peraturan yang sederhana. Papan permainan berbentuk matriks $N \times N$ dan terdapat beberapa angka yang tersebar di atasnya sebagai petunjuk. Setiap angka mewakili ukuran partisi dari angka tersebut. Bentuk partisi yang dibolehkan adalah persegi atau persegi panjang. Setiap partisi hanya memiliki tepat satu angka.

Gambar 1 merupakan contoh Shikaku 5×5 dengan solusi penyelesaiannya. Angka 2, 4, dan 5 pada gambar tersebut adalah petunjuk yang mewakili ukuran partisi. Permainan akan berakhir jika semua angka pada papan permainan sudah mendapatkan partisinya sesuai dengan aturan. Tidak ada pernyataan eksplisit yang menyatakan tingkat kesulitan Shikaku $N \times N$ [4, 5]. Namun, semakin besar ukuran Shikaku, secara otomatis kesulitannya akan bertambah, karena banyaknya partisi dapat bertambah, begitu pula dengan kemungkinan bentuk partisinya.



Gambar 1. Contoh permainan Shikaku 5×5 dengan solusinya.

Permasalahan yang diberikan pada pemainan Shikaku secara umum dapat dikategorikan sebagai masalah NP-*complete* [6]. NP adalah suatu masalah yang dapat diselesaikan dalam waktu polinomial menggunakan *non-deterministic* Turing *machine*. Suatu masalah NP dapat dikategorikan sebagai NP-*complete* jika masalah tersebut termasuk NP-*hard* dan dapat diverifikasi dalam waktu polinomial [7, 8].

Terdapat beberapa penelitian yang sudah dilakukan mengenai pencarian solusi atau pemecahan Shikaku [3, 4, 9, 10]. Adriansyah [9] membahas pendekatan pencarian solusi Shikaku menggunakan kombinasi tiga algoritma, yaitu *greedy*, BFS+*bounding function*, dan runut balik. Namun, pada [9], penulis tidak membahas langkah-langkah pencarian solusi Shikaku dengan sistematis dan tidak menganalisi performansi kombinasi tiga algoritma yang diajukannya secara detail. Lukas et al. [4] membahas pendekatan pencarian solusi Shikaku menggunakan implementasi teknik heuristik dan algoritma genetika. Sama seperti penelitian pada [9], pada [4], penulis juga tidak melakukan analisis komprehensif terkait algoritma yang digunakan. Schrama [3] membahas pembuatan pemecah Shikaku yang efektif menggunakan teknik heuristik. Sama seperti penelitian pada [4, 9], pada [3], penulis juga tidak melakukan analisis komprehensif terkait algoritma yang digunakan. Simonis [10] menjelaskan model untuk Shikaku menggunakan teknik CP, MIP, dan SAT. Sama seperti penelitian pada [3, 4, 9], pada [10], penulis juga tidak melakukan analisis komprehensif terkait algoritma yang digunakan. Pada keempat makalah tersebut belum ada yang melakukan analisis komprehensif terhadap algoritma yang digunakan. Selain itu algoritma-algoritma pada penelitian tersebut tidak didesain untuk mencari semua solusi Shikaku yang mungkin. Kendala lain adalah teknik heuristik, terutama algoritma genetika yang digunakan Lukas et al., belum tentu menghasilkan solusi optimal terhadap suatu masalah NP-*complete*.

Pada penelitian ini penulis menggunakan pendekatan logika proposisi untuk mencari semua solusi yang mungkin ada pada Shikaku $N \times N$. Sejauh pengetahuan penulis, belum ada penelitian terkait pemecahan Shikaku menggunakan pendekatan logika proposisi. Pendekatan logika proposisi dipilih karena dengan bantuan SAT *solver* secara teori dapat ditemukan semua solusi yang mungkin ada pada Shikaku. Sebelumnya, Musa [11] menggunakan pendekatan logika proposisi untuk memecahkan masalah permainan Sudoku.

Untuk mencari solusi Shikaku, pertama aturan-aturan Shikaku ditranslasikan ke dalam formula logika proposisi dalam bentuk normal konjungtif (*conjunctive normal form*, CNF). Selanjutnya aturan-aturan tersebut akan didefinsikan sebagai kendala (*constraint*) yang membatasi ruang pencarian solusi. Seluruh formula yang mewakili aturan Shikaku yang sifatnya esensial akan diproses menggunakan SAT *solver* untuk menemukan semua solusi

yang mungkin pada suatu Shikaku. Kemudian pemecahan Shikaku diterapkan dalam Python. Python dipilih karena memiliki *library* Pycosat dan Satispy untuk menjalankan SAT *solver*. Program dapat menerima masukan berupa dokumen dalam format .txt, kemudian semua solusi akan ditampilkan dalam program.

Availability: Source code program dan data tes tersedia di `https://github.com/Abdusha/Shikaku-Solver`.