Implementasi SAT pada Permainan Numberlink

Francesco Michael Kusuma - 13522038
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
francescokusuma15@gmail.com

***Abstract*— Numberlink is a challenging puzzle game where players must connect number on a grid so that all pairs of numbers are connected by non-crossing paths. This game can be reduced to a Constraint Satisfaction Problem (CSP), which can then be solved using propositional satisfaction (SAT) techniques. This paper details the process of converting Numberlink constraints into Conjunctive Normal Form (CNF) so that they can be solved by a SAT solver. The steps involved include defining variables and indices, outlining five main clauses that must be satisfied, and providing specific CNF clauses for each constraint. This paper provides a robust foundation for developing an efficient SAT-based solution for solving Numberlink puzzles and demonstrates how this concept can be extended to similar games and problems.**

Keywords—Numberlink, Puzzle, SAT, Boolean, Solver

I. PENDAHULUAN

Sumber : https://www.pexels.com/search/puzzle%20pieces/

Permainan logika telah lama menjadi subjek yang menarik baik bagi penggemar teka-teki maupun peneliti dalam bidang ilmu komputer. Salah satu permainan yang terkenal dan menantang adalah Numberlink. Numberlink mengharuskan pemain untuk menghubungkan pasangan angka yang ditempatkan pada sebuah grid, di mana setiap pasangan angka harus terhubung dengan jalur kontinu yang tidak saling bersilangan dan menutupi seluruh grid. Tujuan permainan ini adalah untuk menemukan konfigurasi jalur yang benar di mana semua titik terhubung sesuai aturan yang ditentukan.

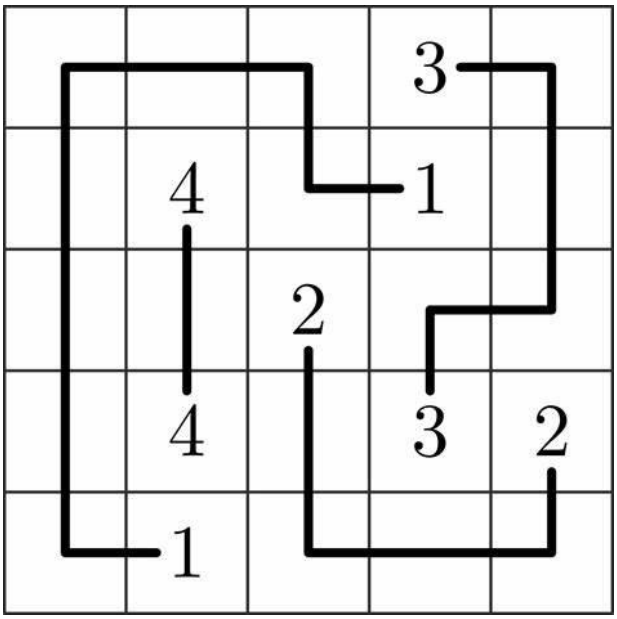
Permainan Numberlink dapat dipandang sebagai masalah pemuasan batasan atau *Constraint Satisfaction Problem (CSP)*, di mana solusi yang dicari harus memenuhi sejumlah batasan yang telah ditentukan. Pendekatan yang umum digunakan untuk menyelesaikan *CSP* adalah dengan mereduksinya menjadi masalah pemuasan proposisional atau *Boolean Satisfiability Problem (SAT).* Dalam konteks *SAT*, kita mencari nilai benar/salah kepada sekumpulan variabel logika sedemikian rupa sehingga semua klausa dalam bentuk normal konjungtif (*CNF*) terpenuhi.

Konversi batasan permainan Numberlink ke dalam *CNF* merupakan langkah yang penting dan kompleks. Bentuk normal konjungtif terdiri dari sejumlah klausa disjungtif yang semuanya harus dipenuhi secara konjungtif. Setiap klausa adalah disjungsi dari satu atau lebih literal, yang berupa variabel logika tunggal atau negasinya. Mengubah batasan logika ke dalam *CNF* memungkinkan penggunaan *solver SAT* untuk menemukan solusi yang memenuhi semua batasan tersebut.

Makalah ini bertujuan untuk memberikan panduan langkah demi langkah dalam mereduksi permainan Numberlink menjadi masalah *SAT*. Penulis akan menjelaskan bagaimana mendefinisikan variabel dan indeks yang diperlukan, serta memberikan klausa-klausa dalam *CNF* untuk setiap batasan yang harus dipenuhi. Dengan demikian, makalah ini menyediakan fondasi yang kuat bagi pengembangan solusi berbasis *SAT* untuk permainan Numberlink dan memperluas konsep ini ke masalah-masalah serupa.

II. LANDASAN TEORI

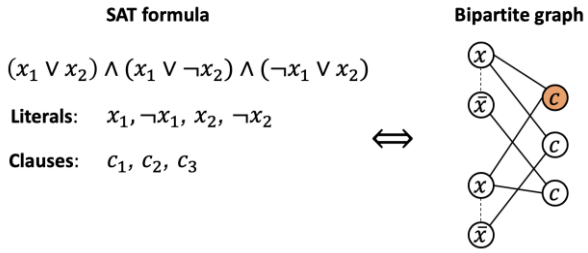
A. *Numberlink*



Sumber: https://www.isnphard.com/i/numberlink/

Permainan Numberlink adalah sebuah permainan teka-teki logika yang sering dimainkan secara soliter. Permainan ini biasa dimainkan di atas sebuah *grid* berukuran tertentu, yang di dalamnya terdapat sejumlah pasangan angka yang harus dihubungkan dengan garis. Garis-garis tersebut harus terhubung secara kontinu dan tidak boleh bersilangan dengan garis lainnya. Setiap angka hanya dapat dihubungkan dengan satu angka lainnya. Tidak boleh ada cabang dalam garis yang menghubungkan angka-angka tersebut.

B. *SAT(Boolean satisfiability problem)*



Sumber: G2SAT (stanford.edu)

SAT atau biasa disebut sebagai *Boolean satisfiability problem* adalah masalah dalam ilmu komputer dan teori komputasi yang berhubungan dengan menentukan apakah suatu formula proposisi Boolean dapat *satisfy* atau tidak. Dengan kata lain, apakah suatu proporsi Boolean jika diganti nilainya akan selalu bernilai *True* atau tidak. Jika iya, proposisi Boolean tersebut dikatakan sebagai *satisfiable*. Formula proposisi ini terdiri dari variabel-variabel Boolean (biasanya dilambangkan dengan x_1, x_2, hingga x_n.

SAT adalah salah satu dari sedikit masalah dalam kelas *NP* yang dikenal memiliki status kompleksitas yang paling sulit untuk dipecahkan. Ini berarti bahwa, meskipun tidak ada algoritma yang diketahui secara efisien dapat menyelesaikan semua instansi masalah *SAT*, jika kita diberikan solusi, kita dapat memverifikasi kebenarannya dalam waktu yang efisien.

Bentuk normal konjungtif atau *Conjunctive Normal Form (CNF)* adalah bentuk standar untuk merepresentasikan formula proposisional dalam *SAT*. *CNF* terdiri dari klausa-klausa yang dihubungkan dengan operator *AND* (konjungsi), di mana setiap klausa adalah disjungsi dari satu atau lebih literal. Literal adalah variabel logika atau negasinya. Mengubah batasan logika ke dalam *CNF* memungkinkan kita untuk memanfaatkan *solver SAT* untuk mencari solusi yang memenuhi semua batasan tersebut.

III. KONVERSI NUMBERLINK MENUJU SAT

Untuk menyelesaikan permainan Numberlink menggunakan *SAT*, buat variabel setiap sel pada grid untuk bisa digunakan pada *SAT*. Misal suatu grid memiliki ukuran M x N dengan terdapat C angka. Lalu, arah dari jalur angka yang dilalui hanya terdapat enam yang disimbolkan dengan t, yaitu

t	Arah
1	atas-bawah
2	kiri-kanan
3	atas-kiri
4	atas-kanan
5	bawah-kanan
6	bawah-kiri

Suatu sel disimbolkan dengan $x_{i,j,u,t}$ dan memiliki arti, pada sel (i, j), jika diberi angka u akan menghasilkan benar atau salah dan memiliki arah t. Nilai dari x ∈ {0 melambangkan salah, 1 melambangkan benar}, i ∈ {1, …, M}, j ∈ {1, …, N}, u ∈ {1, …, C}, dan t ∈ {1, 2, 3, 4, 5, 6}.

Terdapat lima kalusa yang harus dipenuhi oleh suatu permainan Numberlink

A. ***Setiap sel hanya boleh diisi dengan satu angka***

Untuk setiap sel (i, j), dibuat ${}_C C_2 = C(C-1)/2$ variabel untuk menentukan kombinasi dari dua angka yang tidak boleh ada pada sel yang sama. Jika solusi dari $x_{i,j,u,t}$ bernilai benar, sel (i, j) angka u dan memiliki arah t. Oleh karena itu, klausa yang terbentuk adalah

$$(\neg x_{i,j,1,t} \vee \neg x_{i,j,2,t}) \wedge (\neg x_{i,j,1,t} \vee \neg x_{i,j,3,t}) \wedge \cdots \wedge (\neg x_{i,j,1,t} \vee \neg x_{i,j,C,t}) \wedge (\neg x_{i,j,2,t} \vee \neg x_{i,j,3,t}) \wedge (\neg x_{i,j,2,t} \vee \neg x_{i,j,4,t}) \wedge \cdots \wedge (\neg x_{i,j,1,t} \vee \neg x_{i,j,C,t}) \wedge \cdots \wedge (\neg x_{i,j,C-1,t} \vee \neg x_{i,j,C,t})$$

B. ***Angka di titik awal dan titik akhir telah ditentukan***

Angka titik awal dan titik akhir telah ditentukan sehingga setiap titik awal dan akhir yang berada pada sel (i, j) dan memiliki angka u dan awah t. Ini dapat menjadi suatu klausa baru

$$x_{i,j,u,t} \wedge \neg x_{i,j,v,t} \text{ dengan } v \neq u$$

C. ***Setiap titik awal dan titik akhir hanya memiliki satu tetangga yang memiliki angka sama***

Misalkan titik awal atau titik akhir (i, j) memiliki angka u. Karena setidaknya satu tetangga dari titik awal atau titik akhir harus memiliki angka u, kita bisa menulis klausa

$$x_{i,j,u,t} \vee x_{i+1,j,u,t} \vee x_{i-1,j,u,t} \vee x_{i,j+1,u,t} \vee x_{i,j-1,u,t}$$

D. ***Arah dari setiap sel yang bukan titik awal dan titik akhir sesuai dengan salah satu dari enam jenis arah.***

Setiap sel (i, j) yang memiliki angka u hanya boleh memiliki 1 arah saja. Tidak mungkin terdapat suatu sel yang bisa memiliki arah lebih dari satu. Klausa yang dihasilkan oleh syarat ini adalah

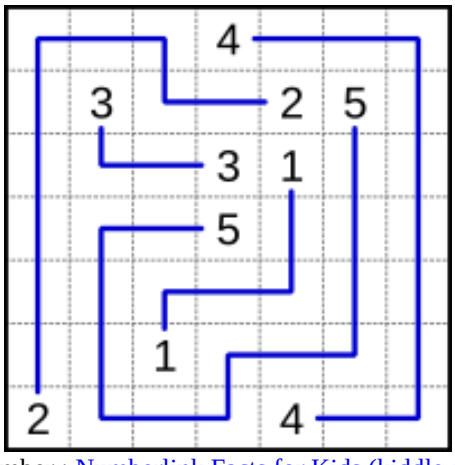
$$(x_{i,j,u,1} \vee x_{i,j,u,2} \vee x_{i,j,u,3} \vee x_{i,j,u,4} \vee x_{i,j,u,5} \vee x_{i,j,u,6}) \wedge$$
$$(\neg x_{i,j,u,1} \vee \neg x_{i,j,u,2} \vee \neg x_{i,j,u,3} \vee \neg x_{i,j,u,4} \vee \neg x_{i,j,u,5} \vee \neg x_{i,j,u,6})$$

E. ***Tetangga dari sebuah sel yang sesuai dengan jenis arahnya harus memiliki angka yang sama***

Setiap tetangga sel yang ditunjuk oleh arah sel (i, j) harus memiliki angka yang sama. Ini menyebabkan menghasilkan klausa

$$(x_{i,j,u,1} \wedge x_{i+1,j,u,1} \wedge x_{i-1,j,u,1}) \wedge$$
$$(x_{i,j,u,2} \wedge x_{i,j+1,u,2} \wedge x_{i,j-1,u,2}) \wedge$$
$$(x_{i,j,u,3} \wedge x_{i-1,j,u,3} \wedge x_{i,j-1,u,3}) \wedge$$
$$(x_{i,j,u,4} \wedge x_{i-1,j,u,4} \wedge x_{i,j+1,u,4}) \wedge$$
$$(x_{i,j,u,5} \wedge x_{i+1,j,u,5} \wedge x_{i,j+1,u,5}) \wedge$$
$$(x_{i,j,u,6} \wedge x_{i+1,j,u,6} \wedge x_{i,j+1,u,6}) \wedge$$

IV. IMPLEMENTASI SAT PADA NUMBERLINK



Sumber : Numberlink Facts for Kids (kiddle.co)

Pada Bagian III, telah ditunjukkan bahwa harus ada lima syarat yang dipenuhi oleh suatu permainan Numberlink untuk bisa menjadi solusi formal. Akan ditinjau kelima syarat tersebut apakah memenuhi syarat untuk menjadi solusi pada contoh permainan di atas :

A. ***Setiap sel hanya boleh diisi dengan satu angka***

Dalam contoh permainan tersebut, setiap sel pada grid hanya diisi oleh satu angka saja. Tidak ada sel yang memiliki lebih dari satu angka atau lebih dari satu arah. Oleh karena itu, syarat ini terpenuhi.

B. ***Angka di titik awal dan titik akhir telah ditentukan***

Dalam contoh permainan, titik awal dan titik akhir sudah ditentukan sebelumnya dengan angka-angka yang sesuai. Hal ini memastikan bahwa jalur yang menghubungkan titik awal dan titik akhir dengan angka yang sama adalah tetap dan konsisten. Oleh karena itu, syarat ini terpenuhi.

C. ***Setiap titik awal dan titik akhir hanya memiliki satu tetangga yang memiliki angka sama***

Dalam contoh permainan, setiap titik awal dan titik akhir hanya memiliki satu tetangga dengan angka yang sama, memastikan bahwa jalur yang menghubungkan mereka adalah unik dan tidak bercabang. Oleh karena itu, syarat ini terpenuhi.

D. ***Arah dari setiap sel yang bukan titik awal dan titik akhir sesuai dengan salah satu dari enam jenis arah.***

Dalam contoh permainan, setiap sel yang bukan titik awal atau titik akhir memiliki arah yang jelas sesuai dengan salah satu dari enam jenis arah yang ditentukan. Tidak ada sel yang memiliki lebih dari satu arah atau arah yang tidak sesuai. Oleh karena itu, syarat ini terpenuhi.

E. ***Tetangga dari sebuah sel yang sesuai dengan jenis arahnya harus memiliki angka yang sama***

Dalam contoh permainan, setiap tetangga dari sebuah sel yang sesuai dengan arah dari sel tersebut memiliki angka yang sama. Ini memastikan bahwa jalur yang dibuat adalah konsisten dan benar sesuai dengan arah yang ditentukan. Oleh karena itu, syarat ini terpenuhi.

Setelah meninjau semua syarat yang harus dipenuhi oleh solusi formal dari permainan Numberlink, dapat disimpulkan bahwa contoh permainan Numberlink memenuhi semua lima syarat yang telah ditentukan. Dengan demikian, dapat disimpulkan bahwa kandidat solusi yang diberikan untuk permainan Numberlink adalah solusi yang benar dan memenuhi semua syarat formal yang telah ditentukan.

V. IMPLEMENTASI DALAM C++

Kode ini bertujuan untuk menyelesaikan permainan Numberlink menggunakan pendekatan *SAT (Boolean Satisfiability Problem)*. *SAT solver* digunakan untuk menentukan apakah ada solusi yang memenuhi semua kondisi yang diberikan untuk permainan ini.

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <cassert>
#include "solver.h" // Include file for
SAT solver

using namespace std;

const int MAX_N = 100;
const int MAX_C = 10;

struct Cell {
    int u; // Number
    int t; // Direction
};

int M, N, C;
Cell grid[MAX_N][MAX_N];

// Function to convert grid cell
coordinates to variable index
int var(int i, int j, int u, int t) {
    return ((i * N + j) * C + u) * 6 + t
- 1;
}

// Function to add clause to SAT solver
void addClause(vector<vector<int>>&
clauses, int literal) {
    clauses.push_back({literal});
}

// Function to solve the Numberlink
puzzle using SAT solver
bool solveNumberlinkSAT() {
    Solver solver;
    vector<vector<int>> clauses;

    // Clause A: Each cell is filled with
only one number
    for (int i = 0; i < M; ++i) {
        for (int j = 0; j < N; ++j) {
            for (int u = 1; u <= C; ++u)
{
                for (int t1 = 1; t1 <= 6;
++t1) {
                    for (int t2 = t1 + 1;
t2 <= 6; ++t2) {
addClause(clauses, -var(i, j, u, t1));

addClause(clauses, -var(i, j, u, t2));
                    }
                }
            }
        }
    }

    // Clause B: Initial and final points
are fixed
    // Initial point
    addClause(clauses, var(0, 0,
grid[0][0].u, grid[0][0].t));
    // Final point
    addClause(clauses, var(M - 1, N - 1,
grid[M - 1][N - 1].u, grid[M - 1][N -
1].t));

    // Clause C: Initial and final points
have exactly one neighbor with the same
number
    addClause(clauses, -var(0, 0,
grid[0][0].u, grid[0][0].t));
    addClause(clauses, -var(M - 1, N - 1,
grid[M - 1][N - 1].u, grid[M - 1][N -
1].t));

    // Clause D: Each cell has only one
direction
    for (int i = 0; i < M; ++i) {
        for (int j = 0; j < N; ++j) {
            for (int u = 1; u <= C; ++u)
{
                for (int t = 1; t <= 6;
++t) {
                    for (int tt = 1; tt
<= 6; ++tt) {
                        if (t != tt)
addClause(clauses, -var(i, j, u, t));
                    }
                }
            }
        }
    }

    // Clause E: Neighbors with the same
direction have the same number
    for (int i = 0; i < M - 1; ++i) {
        for (int j = 0; j < N - 1; ++j) {
            for (int u = 1; u <= C; ++u)
{
                for (int t = 1; t <= 6;
++t) {
                    addClause(clauses, -
var(i, j, u, t));
```

```
                    addClause(clauses, -
var(i + 1, j + 1, u, t));
                }
            }
        }
    }

    // Adding all clauses to the SAT
solver
    for (const auto& clause : clauses) {
        solver.addClause(clause);
    }

    // Solving the SAT problem
    vector<int> solution;
    bool isSatisfiable =
solver.solve(solution);

    // Output the solution
    if (isSatisfiable) {
        cout << "Solution found:" <<
endl;
        // Your implementation here to
output the solution
    } else {
        cout << "No solution found." <<
endl;
    }

    return isSatisfiable;
}
```

Penjelasan Kode:

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <cassert>
#include "solver.h" // Include file for
SAT solver

using namespace std;

const int MAX_N = 100;
const int MAX_C = 10;
```

Bagian ini memuat header yang diperlukan, termasuk pustaka standar C++ dan header untuk *SAT solver*. Konstanta `MAX_N` dan `MAX_C` mendefinisikan ukuran maksimum grid dan jumlah maksimum angka yang berbeda.

```
struct Cell {
    int u; // Number
    int t; // Direction
};

int M, N, C;
Cell grid[MAX_N][MAX_N];
```

Struktur `Cell` menyimpan angka (`u`) dan arah (`t`) dari setiap sel dalam grid. Variabel `M`, `N`, dan `C` masing-masing menyimpan ukuran grid (baris dan kolom) serta jumlah angka yang berbeda.

```
// Function to convert grid cell
coordinates to variable index
int var(int i, int j, int u, int t) {
    return ((i * N + j) * C + u) * 6 + t
- 1;
}
```

Fungsi `var` mengubah koordinat grid `(i, j)`, angka `u`, dan arah `t` menjadi indeks variabel unik untuk digunakan dalam SAT solver.

```
// Function to add clause to SAT solver
void addClause(vector<vector<int>>&
clauses, int literal) {
    clauses.push_back({literal});
}
```

Fungsi `addClause` menambahkan klausa baru ke dalam daftar klausa. Klausa disimpan sebagai vektor dari literalan.

```
// Function to solve the Numberlink
puzzle using SAT solver
bool solveNumberlinkSAT() {
    Solver solver;
    vector<vector<int>> clauses;

    // Clause A: Each cell is filled with
only one number
    for (int i = 0; i < M; ++i) {
        for (int j = 0; j < N; ++j) {
            for (int u = 1; u <= C; ++u)
{
                for (int t1 = 1; t1 <= 6;
++t1) {
                    for (int t2 = t1 + 1;
t2 <= 6; ++t2) {

addClause(clauses, -var(i, j, u, t1));

addClause(clauses, -var(i, j, u, t2));
                    }
                }
            }
        }
    }
```

```
    // Clause B: Initial and final points
are fixed
    // Initial point
    addClause(clauses, var(0, 0,
grid[0][0].u, grid[0][0].t));
    // Final point
    addClause(clauses, var(M - 1, N - 1,
grid[M - 1][N - 1].u, grid[M - 1][N -
1].t));

    // Clause C: Initial and final points
have exactly one neighbor with the same
number
    addClause(clauses, -var(0, 0,
grid[0][0].u, grid[0][0].t));
    addClause(clauses, -var(M - 1, N - 1,
grid[M - 1][N - 1].u, grid[M - 1][N -
1].t));

    // Clause D: Each cell has only one
direction
    for (int i = 0; i < M; ++i) {
        for (int j = 0; j < N; ++j) {
            for (int u = 1; u <= C; ++u)
{
                for (int t = 1; t <= 6;
++t) {
                    for (int tt = 1; tt
<= 6; ++tt) {
                        if (t != tt)
addClause(clauses, -var(i, j, u, t));
                    }
                }
            }
        }
    }

    // Clause E: Neighbors with the same
direction have the same number
    for (int i = 0; i < M - 1; ++i) {
        for (int j = 0; j < N - 1; ++j) {
            for (int u = 1; u <= C; ++u)
{
                for (int t = 1; t <= 6;
++t) {
                    addClause(clauses, -
var(i, j, u, t));
                    addClause(clauses, -
var(i + 1, j + 1, u, t));
                }
            }
        }
    }

    // Adding all clauses to the SAT
solver
    for (const auto& clause : clauses) {
        solver.addClause(clause);
    }

    // Solving the SAT problem
    vector<int> solution;
    bool isSatisfiable =
solver.solve(solution);

    return isSatisfiable;
}
```

Pada fungsi `solveNumberlinkSAT()`, terdapat tiga hal yang dilakukan, yaitu:

a. Menginisialisasi *solver SAT* dan vektor klausa.
b. Menambahkan klausa untuk setiap aturan permainan Numberlink (A sampai E) ke dalam solver.
c. Menyelesaikan masalah SAT dan mengeluarkan hasilnya.

VI. KESIMPULAN

Permainan Numberlink menawarkan tantangan logika yang menarik. Pemain harus menghubungkan pasangan titik-titik berwarna pada sebuah grid melalui jalur yang tidak saling bersilangan. Menyelesaikan permainan ini secara manual bisa sangat menantang, terutama untuk grid berukuran besar. Oleh karena itu, mereduksi permainan Numberlink ke dalam masalah pemuasan proposisional (*SAT*) adalah pendekatan yang efisien dan menarik. Namun, perlu diingat bahwa *SAT* termasuk ke dalam kelompok *NP-Complete* yang berarti waktu untuk memverifikasi solusi saja yang bisa berjalan dalam waktu polinomial. Namun, waktu untuk menerka solusi tidak bisa dijalankan dalam waktu polinomial.

Dengan menggunakan metode yang diuraikan dalam makalah ini, permainan Numberlink dapat dikonversi ke dalam bentuk SAT dan memanfaatkan *solver SAT* untuk menemukan solusi. Pendekatan ini tidak hanya terbatas pada Numberlink, tetapi juga dapat diterapkan pada berbagai jenis permainan lainnya.

VII. UCAPAN TERIMA KASIH

Puji Tuhan kepada Tuhan Yang Maha Esa karena atas Rahmat dan berkat-Nya, makalah berjudul “Implementasi SAT pada Permainan Numberlink” dapat diselesaikan dengan baik. Terima kasih juga diberikan kepada dosen pengampu mata kuliah IF2211 Strategi Algoritma Dr. Nur Ulfa Maulidevi atas bimbingan selama perkuliahan.

PRANALA VIDEO

Demo penjelasan makalah:
https://youtu.be/XQL3_n44QTg

REFERENCES

[1] Kotsuma, Kouichi, & Takenaga, Yasuhiko**.** (March 2010). "NP-Completeness and Enumeration of Number Link Puzzle". *IEICE Technical Report. Theoretical Foundations of Computing*, 109 (465): 1–7.

[2] Yuen, Henry. (Fall 2019). "CS 121: Introduction to Theoretical Computer Science - Project: Solving Puzzles with SAT Solvers". Retrieved from https://www.henryyuen.net/fall2019/projects/puzzles.pdf.

[3] Zucker, Matt. (September 2016). "Eating SAT-Flavored Crow". Retrieved from https://mzucker.github.io/2016/09/02/eating-sat-flavored-crow.html.

[4] Vaney, Tor. "Flow Solver: An Exploration into Solving the Flow Free Puzzle Using SAT". Retrieved from https://torvaney.github.io/projects/flow-solver.html.

[5] Miller, Jack. (March 2020). "Deep Learning vs. Puzzle Games". *Towards Data Science*. Retrieved from https://towardsdatascience.com/deep-learning-vs-puzzle-games-e996feb76162.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Francesco Michael Kusuma (13522038)