Penerapan Graph Theory dan Graph Coloring dalam Memecahkan Teka-teki Sudoku

Muhammad Zaki Amanullah - 13521146[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521146@std.stei.itb.ac.id

***Abstract*—Sudoku adalah permainnan teka-teki yang meilbatkan kombinasi angka-angka yang diletakkan pada tabel yang memiliki ukuran 9x9 sel dimana sel-sel tersebut dikelompokkan kembali dalam upatabel yang berukuran 3x3. Permainan diawali dengan mengisi sebagian sel dengan angka-angka yang valid. Tujuan dari permainan ini adalah untuk mengisi sel-sel yang tidak terisi dengan angka-angka yang valid, yaitu angka-angka yang belum pernah muncul pada baris, kolom dan upatabel dimana sel tersebut berada. Makalah ini akan membahas bagaimana kita dapat memanfaatkan *graph theory* dan *graph coloring* untuk dapat melihatkan solusi-solusi yang mungkin diambil dalam permainan sudoku.**

***Keywords*—sudoku, sel, tabel, upatabel, baris, kolom, angka, valid, *graph theory*, *graph coloring*.**

I. PENDAHULUAN

Sudoku adalah permainan teka-teki yang dimainkan dalam sebuah tabel yang berukuran 9x9 sel dimana sel-sel tersebut dikelompokkan kembali menjadi upa-tabel berukuran 3x3. Tujuan dari permainan ini adalah untuk mengisi setiap sel dengan angka 1—9 yang dibuat agar angka-angka dalam suatu sel tidak muncul lagi di dalam baris, kolom, dan upagraf dimana sel tersebut berada. Karena namanya yang terdengar seperti nama yang berasal dari Jepang, banyak artikel yang merujuk bahwa sudoku adalah teka-teki yang berasal dari Jepang. Namun, pada kenyataanya permainan ini diciptakan oleh seorang ahli teka-teki pada tahun 1970-an yang terinsipirasi dari perjalanannya dari Tokyo, London, dan berakhir di New York.
Ide dari teka-teki ini sudah muncul mulai dari tahun 1783. Matematikawan Swiss, Leonhard Euler, mencetuskan sebuah permainan dengan nama 'Latin Squares', rangkaian sel yang memiliki angka-angka berbeda pada tiap sel. Pada tahun 1970-an, Dell Puzzle Magazines, sebuah penerbit majalah di New York yang membuat teka-teki mulai tahun 1931, merilis teka-teki tersebut. Hal itulah yang membuat teka-teki yang pada saat itu diberi nama 'Number Place' menjadi terkenal. Setelah terus-menerus menerbitkan teka-teki tersebut pada tahun 1980-an, teka-teki tersebut mulai dikenal dimata publik dan imitasi-pun mulai bermunculan. Di Jepang, seorang penerbit bernama Nikoli menambah beberapa konsep tambahan pada teka-teki tersebut dan Ia mengganti nama teka-teki tersebut menjadi sudoku yang berasal dari bahasa Jepang *su* yang berarti angka dan *doku* yang secara kasar memiliki arti unik. Permainan sudoku terisolasi di negara-negara timur selama 20 tahun hingga pada suatu saat seseorang yang berasal dari Matamata, Selandia Baru bernama Wayne Gould yang secara tidak sengaja melihat teka-teki tersebut saat Ia sedang membeli kebutuhan sehari-harinya. Dalam kurun waktu 6 tahun setelah itu, Ia membuat program komputer yang dapat membuat teka-teki sudoku dalam waktu singkat. Gould menerbitkan teka-teki buatannya dalam koran lokal yang lama kelamaan akan tersebar sampai ke Inggris. Setelah itu, penerbit koran mulai berlomba-lomba menerbitkan teka-teki sudoku dengan nama yang berbeda-beda.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

***Gambar 1.1**: Contoh tabel sudoku pada awal permainan.*

Sumber:
https://upload.wikimedia.org/wikipedia/commons/thumb/f/ff/Sudoku-by-L2G-20050714.svg/250px-Sudoku-by-L2G-20050714.svg.png
diakses pada 9 Desember 2022 pukul 17:15.

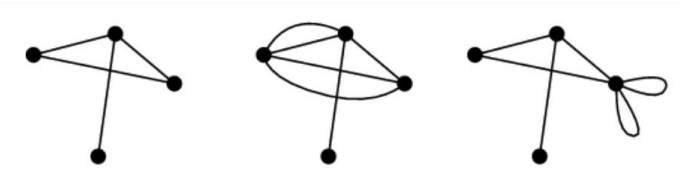
Tingkat kesulitan dari permainan ini dapat dilihat dari jumlah sel yang sudah terisi pada awal permainan. Semakin sedikit, jumlah sel yang terisi, tingkat kesulitan dari permainan ini akan semakin bertambah. Pada tingkat kesuliatan tertentu, permainan ini dapat dilakukan dengan metode eliminasi. Dengan mengeliminasi angka apa saja yang telah muncul dalam baris, kolom, dan upatabel tertentu, kita akan mendapatkan semua angka yang mungkin muncul dalam sebuah sel.

Dengan menggunakan *graph theory* dan *graph coloring* kita dapat memanfaatkan fakta bahwa setiap sel dalam sebuah kolom, baris, dan upagraf tertentu saling terkoneksi, sehingga kita dapat menemukan setiap kemungkinan angka dalam sebuah

sel. Dari pernyataan tersebut, sudah mulai tergambar bahwa kita dapat merepresentasikan sebuah tabel sudoku dalam sebuah graf tidak berarah, dimana setiap simpul adalah sebuah sel dan setiap koneksi adalah sebuah sisi. Dua buah simpul dapat kita katakan terhubung atau bertetangga apabila kedua buah simpul tersebut merupakan representasi dari dua buah sel yang berada pada baris yang sama, kolom, yang sama atau pada upatabel yang sama. Konsep dari graph coloring adalah kita tidak dapat mewarnai dua buah simpul yang bertetangga dengan warna yang sama. Konsep tersebut dapat dimanfaatkan untuk mendapatkan solusi untuk teka-tekai sudoku dengan tingkat kesulitan yang mudah. Algoritma yang akan diguankan dalam makalah ini adalah algoritma *greedy coloring* yang akan memanfaatkan *adjacency matrix* sebagai matriks untuk representasi graf tidak berarah. Namun, untuk tingkat kesulitan teka-teki sudoku yang lebih rumit, konsep graph coloring hanya akan mendapatkan kandidat-kandidat angka yang mungkin masuk dalam sebuah sel. Kandidat-kandidat angka tersebut mungkin memiliki anggota yang sama. Untuk itu, diperlukan sebuah langkah tambahan untuk mengeliminasi angka-angka yang muncul berulang dalam dua buah sel dengan baris, kolom, atau upatabel yang sama. Metode eliminasi yang akan diulas dalam makalah ini adalah metode tuple. Metode tuple pada dasarnya akan mengelompokkan semua sel yang memiliki kandidat sama dengan sel lain dalam sebuah graf, lalu mengelompokkan sebuah tuple dengan ukuran n dan dengan n kandidat unik, lalu mengeliminasi semua kemunculan kandidat unik tersebut dalam simpul yang bertetangga.

II. Teori Dasar

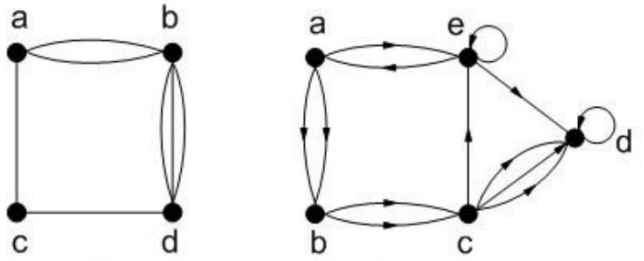
Graf didefinisikan sebagai sebuah tuple G = (V, E) yang berisikan himpunan tidak kosong simpul V = {$v_1, v_2, \ldots, v_n$} dan himpunan sisi E = {$e_1, e_2, \ldots, e_n$}, yang mengubungkan pasangan simpul. Berdasarkan kompleksitasnya, graf dapat dibedakan menjadi dua, graf sederhana dan graf tidak sederhana. Graf sederhana adalah graf yang tidak memiliki sisi ganda, dua buah sisi yang menghubungkan simpul yang sama, atau sisi gelang, sisi yang menghubungkan satu buah simpul.



***Gambar 2.1.1**: Macam graf berdasarkan kompleksitasnya. Dari kiri ke kanan: graf sederhana, graf tidak sederhana dengan sisi ganda, dan graf tidak sederhana dengan sisi gelang.*

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada 9 Desember 17:28.

Graf juga dapat dibedakan berdasarkan orientasi sisinya. Graf yang tiap sisinya memiliki arah disebut graf berarah. Graf yang sisi-sisinya tidak memiliki arah disebut graf tidak berarah.

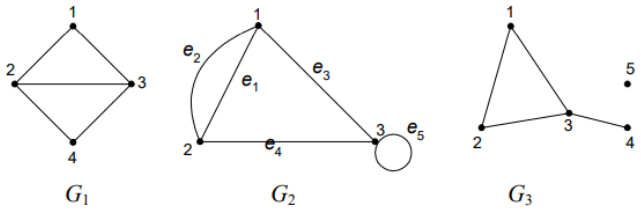


***Gambar 2.1.2**: (Dari kiri ke kanan) Graf tidak berarah dan graf tidak berarah.*

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada 9 Desember 2022, pukul 17:32.

Terminologi-terminologi dalam graf yang perlu diperhatikan adalah sebagai berikut.

1) Ketetanggaan
   Dua buah simpul dikatakan bertetangga apabila kedua simpul tersebut terhubung langsung. Sebagai contoh pada graf G_1 pada gambar 2.1.3, simpul 1 bertetangga dengan simpul 2 dan simpul 3. Simpul 3 bertetangga dengan simpul 1, simpul 2, dan simpul 4.

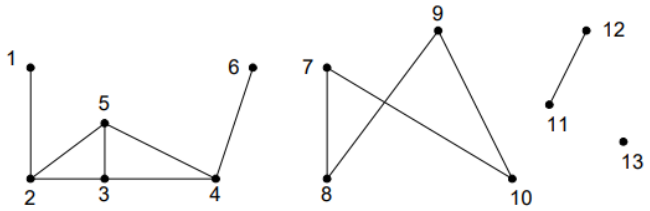


***Gambar 2.1.3**: Ilustrasi beberapa graf. Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada 10 Desember 2022 pukul 22:46 WIB.*

2) Beririsan
   Untuk sembarang sisi e = (v_j, v_k) dikatakan beririsan dengan simpul v_j dan v_k.
3) Simpul terpencil
   Simpul terpencil adalah simpul yang tidak mempunyai sisi yang beririsan dengannya.
4) Graf kosong
   Graf kosong adalah graf yang himpunan sisinya merupakan himpunan kosong.
5) Derajat
   Pada graf tidak berarah, derajat suatu simpul adalah jumlah sisi yang beririsan dengan simpul tersebut.
6) Lintasan
   Lintasan dengan panjang n dari simpul awal v_o ke simpul tujuan v_n di dalam graf G ialah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk $v_0, e_1, v_1, e_2, v_2, ..., e_n, v_n$ sedemikian sehingga $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, …, $e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf G.
7) Sirkuit
   Sirkit adalah lintasan yang berawal dan berakhir pada simpul yang sama.
8) Keterhubungan
   Dua buah simpul v_1 dan v_2 disebut terhubung jika terdapat lintasan dari v_1 ke v_2.

9) Upagraf

Misalkan G = (V, E) adalah sebuah graf. $G_1 = (V_1, E_1)$ adalah upagraf dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$. **Komponen** graf adalah jumlah maksimum upagraf terhubung dalam G. Graf pada gambar 2.1.4 adalah contoh graf yang memiliki 4 buah komponen.



***Gambar 2.1.4**: Graf yang memiliki lebih dari 4 komponen. Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf. Diakses pada 10 Desember 2022 pukul 23:05 WIB.*

10) Upagraf merentang

Upagraf $G_1 = (V_1, E_1)$ dari G = (V, E) dikatakan upagraf merentang jika $V_1 = V$ (G_1 mengandung semua simpul dari G).

11) Cut set

Cut set dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung.

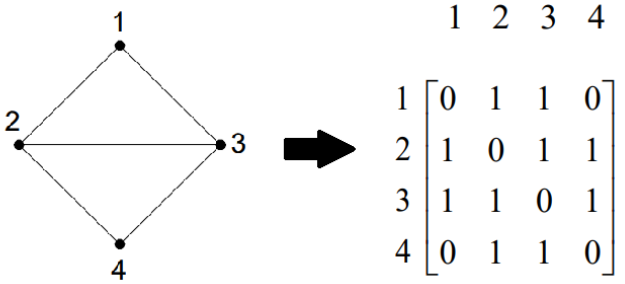
12) Graf berbobot

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot).

Graf dapat direpresentasikan dengan beberapa cara, di antaranya adalah sebagai berikut.

1) Adjacency Matrix

Adjacency matrix adalah represetasi graf dengan menggunakan matriks dimana elemen matriks pada baris *i* dan kolom *j* akan bernilai 1 apabila simpul v_i bertetangga dengan simpul v_j.

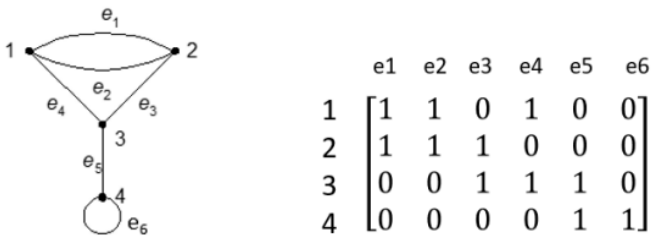


***Gambar 2.1.5**: Contoh represetasi graf dengan menggunakan matriks ketetanggaan.*

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian2.pdf. Diakses pada 9 Desember 2022 pukul 19:31.

2) Incidency Matrix

Incidency matrix adalah representasi matriks dimana setiap elemen matriks a_{ij} bernilai 1 apabila simpul *i* beririsan dengan sisi e_j dan akan bernilai 0 apabila tidak beririsan.



***Gambar 2.1.6**: Contoh represetasi graf dengan menggunakan matriks irisan.*

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian2.pdf. Diakses pada 10 Desember 2022 pukul 19:03.

3) Adjacency list

Adjacency list adalah representasi graf dimana setiap simpul akan diberikan sebuah list yang berisikan simpul-simpul lain yang bertetanggaan dengan simpul tersebut.

Graf coloring adalah cara pewarnaan simpul dalam graf sehingga dua simpul yang bertetangga mempunyai warna berbeda. Konsep *graph coloring* dapat diaplikasikan dalam berbagai bidang, salah satunya adalah dalam mewarnai peta. Peta diwarnai sedemikian rupa sehingga semua warna wilayah yang bersebelahan tidak sama.

III. Aplikasi Graph Theory dan Graph Coloring dalam Memecahkan Puzzle Sudoku

Tujuan dari permainan teka-teki sudoku adalah untuk mengisi semua sel dengan angka yang belum pernah muncul pada kolom, baris, dan subtabel yang sama dengan sel tersebut. Kita dapat memanfaatkan konsep graph coloring untuk mengisi setiap sel yang bertetangga dengan 'warna' yang berbeda. Dalam konteks ini, warna yang kita gunakan adalah angka yang dapat diisi dalam suatu sel sudoku yaitu angka dari satu sampai sembilan. Algoritma yang digunakan adalah algoritma *greedy coloring* dengan langkah-langkah secara garis besar sebagai berikut.

1) Inisiasi semua vertex
2) Inisiasi kondisi awal papan sudoku
3) Iterasi semua vertex mulai dari vertex dengan derajat yang paling tinggi — langkah ini dapat kita lewati karena pada dasarnya semua vertex pada graf sudoku memiliki derajat yang sama — serta pilih vertex yang hanya memiliki satu kemungkinan warna.
4) Cek semua vertex yang bertetangga dengan vertex tersebut, apabila vertex yang bertetangga memiliki sebuah 'warna' yang sama dengan warna pada vertex awal, maka kita eliminasi warna tersebut dari vertex tetangga.
5) Kembali ke langkah tiga sampai tidak ada perubahan lebih lanjut pada list warna yang mungkin dalam sebuah vertex.

Solusi dari permainan sudoku yang akan dibuat belum pasti merupakan solusi yang pasti, karena ada kondisi dimana warna yang mungkin menempati sebuah vertex berjumlah lebih dari satu. Hal tersebut dikarenakan pada kondisi awal permainan sel yang terisi terlalu sedikit sehingga kita tidak tahu pasti akan

solusi uniknya. Solusi yang dihasilkan pada algoritma ini hanya sebatas angka apa saja yang dapat menempati sebuah vertex.

Untuk memudahkan ilustrasi dari algoritma yang dibuat, akan dibuat sebuah program pada bahasa pemrograman python untuk merealisasikan secara sederhana. Dalam merepresentasikan papan permainan sudoku dalam sebuah graf, setiap sel pada tabel sudoku akan dijadikan sebuah simpul. Oleh karena itu, akan ada total sebanyak 81 buah simpul dengan setiap sel yang berada pada baris, kolom, atau subtabel yang sama akan bertetangga dengan sel tersebut. Untuk itu dalam merepresentasikan graf, digunakan sebuah *adjacency matrix*. Berikut kode python untuk menginisialisasi semua elemen *adjacency matrix*.

```
def getBlock(row, col):
    a = int(row / 3)
    b = int(col / 3)
    return a * 3 + b

board = [[0 for i in range(81)] for j in range(81)]
for i in range(81):
    iRow = int(i/9)
    iCol = i - (iRow * 9)
    iBlock = getBlock(iRow, iCol)
    for j in range(81):
        jRow = int(j/9)
        jCol = j - (jRow * 9)
        jBlock = getBlock(jRow, jCol)
        if (jRow == iRow or jCol == iCol or jBlock == iBlock) and i != j:
            board[i][j] = 1
```

Gambar 3.1: *Cuplikan kode untuk inisialisasi adjacency matrix.*

Selanjutnya, setiap vertex akan diberi nama sesuai dengan posisi sel yang direpresentasikan oleh vertex tersebut dalam papan permainan. Pada cuplikan kode di bawah ini digunakan *i* sebagai representasi baris dan *j* sebagai representasi kolom. Sebagai contoh vertex 11 memiliki arti bahwa vertex tersebut merepresentasikan sel pada baris ke-1 dan kolom ke-1.

```
node = [str(i) + str(j) for i in range(1, 10) for j in range(1, 10)]
t_ = {}
for i in range(len(board)):
    t_[node[i]] = i
```

Gambar 3.2: *Cuplikan kode untuk penamaan setiap vertex.*

Untuk langkah selanjutnya akan dilakukan inisialisasi semua angka yang mungkin yaitu angka 1—9 yang akan diletakkan pada sebuah dictionary pada program dengan key berisikan nama node dan value yang berisikan list dari semua angka yang mungkin. Selanjutnya kita akan mengisi sel-sel pada papan dengan angka-angka yang valid. Pada tes yang akan dilakukan, papan permainan akan diisi dengan nilai yang sesuai pada gambar 1.1. Alasan dilakukan tes terhadap teka-teki pada gambar 1.1 adalah karena teka-teki tersebut termasuk teka-teki dengan tingkat kesulitan yang cukup rendah. Diharapkan dengan tingkat kesulitan tersebut, akan didapatkan sebuah hasil yang unik hanya berdasarkan metode *greedy coloring* saja.

```
colorDict = {}
for i in range(len(board)):
    colorDict[node[i]] = ["one", "two", "three", "four", "five", "six", "seven", "eight", "nine"]
```

Gambar 3.3*: Kode untuk inisialisasi semua kemungkinan angka yang mungkin muncul pada setiap kotak.*

Setelah semua sel telah terinisialisasi, kita siap untuk melakukan eliminasi satu persatu terhadap semua vertex. Hal yang perlu dilakukan adalah mengiterasi satu per satu vertex yang ada dan dilakukan pengecekan terhadap angka yang mungkin menempati vertex tersebut. Apabila vertex tersebut memiliki tepat satu buah angka, maka vertex tersebut sudah valid dan selanjutnya akan dilakukan pengecekan terhadap vertex-vertex tetangga dari vertex awal yang memiliki tepat satu buah angka tersebut. Apabila angka yang terdapat pada vertex awal muncul kembali pada vertex-vertex tetangga, akan dilakukan penghapusan angka tersebut dalam list angka yang mungkin. Setelah semua vertex yang bertetangga telah selesai diperiksa, akan dilanjutkan iterasi ke vertex berikutnya. Apabila sel memiliki lebih dari satu buah angka yang mungkin, maka tidak akan dilakukan pengecekan pada sel-sel tetangga, karena vertex tersebut bukan merupakan vertex yang valid. Iterasi di atas dilakukan sampai dengan n kali dengan n adalah waktu minimum agar tidak ada lagi perubahan yang dapat dicapai dalam satu iterasi. Untuk tes kali ini diperlukan 9 kali iterasi untuk mendapatkan hasil yang optimal. Untuk teka-teki yang lebih rumit, menambah jumlah iterasi tidak akan cukup untuk memecahkan masalah sudoku. Karena ada sebuah threshold dimana apabila dilakukan iterasi lebih lanjut, kondisi awal dan kondisi akhir dari graf akan sama. Untuk itu akan dilakukan langkah lebih lanjut untuk masalah yang leibh rumit. Cuplikan kode untuk eliminasi warna ada pada fungsi eliminateColor() sebagai berikut.

```
def eliminateColor(colorDict):
    newDict = colorDict.copy()
    count = 0
    for n in node:
        if (len(newDict[n]) == 1):
            adjacentNode = board[t_[n]]
            for j in range(len(adjacentNode)):
                if j != count and adjacentNode[j] == 1 and (newDict[n][0] in newDict[node[j]]):
                    newDict[node[j]].remove(newDict[n][0])
        count += 1
    return newDict
```

Gambar 3.4*: Cuplikan kode untuk fungsi eliminateColor().*

Untuk mengetahui solusi dari teka-teki pada gambar 1.1 digunakan sebuah referensi lain yang dapat memeriksa validasi dari suatu teka-teki sudoku. Setelah dilakukan pengecekan, solusi dari gambar 1.1 yang valid adalah yang tertera pada gambar 3.5.

Valid | Fill and Eliminate Candidates | Next Hint | Solve

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

***Gambar 3.5**: Solusi yang valid terhadap teka-teki pada gambar 1.1*

Sumber: https://sudoku-solver.rakhman.info/ diakses pada 10 Desember 2022, pukul 19:13.

Untuk tes yang dilakukan—sesuai dengan gambar 1.1—, diperlukan 9 kali iterasi untuk mencapai hasil yang optimal, yaitu hasil yang apabila dilakukan satu kali lagi iterasi tidak akan mengalami perubahan. Jumlah iterasi yang perlu dilakukan dipengaruhi oleh tingkat kesulitan dari kondisi awal teka-teki ini. Dengan kata lain, semakin sedikit angka yang sudah terisi pada awal main, semakin banyak iterasi yang diperlukan. Untuk kondisi awal seperti gambar 1.1 hasil yang dikeluarkan oleh program yang telah dibuat adalah seperti yang tertera pada gambar 3.6.

```
Node 11  =  ['five']    Node 61  =  ['seven']
Node 12  =  ['three']   Node 62  =  ['one']
Node 13  =  ['four']    Node 63  =  ['three']
Node 14  =  ['six']     Node 64  =  ['nine']
Node 15  =  ['seven']   Node 65  =  ['two']
Node 16  =  ['eight']   Node 66  =  ['four']
Node 17  =  ['nine']    Node 67  =  ['eight']
Node 18  =  ['one']     Node 68  =  ['five']
Node 19  =  ['two']     Node 69  =  ['six']
Node 21  =  ['six']     Node 71  =  ['nine']
Node 22  =  ['seven']   Node 72  =  ['six']
Node 23  =  ['two']     Node 73  =  ['one']
Node 24  =  ['one']     Node 74  =  ['five']
Node 25  =  ['nine']    Node 75  =  ['three']
Node 26  =  ['five']    Node 76  =  ['seven']
Node 27  =  ['three']   Node 77  =  ['two']
Node 28  =  ['four']    Node 78  =  ['eight']
Node 29  =  ['eight']   Node 79  =  ['four']
Node 31  =  ['one']     Node 81  =  ['two']
Node 32  =  ['nine']    Node 82  =  ['eight']
Node 33  =  ['eight']   Node 83  =  ['seven']
Node 34  =  ['three']   Node 84  =  ['four']
Node 35  =  ['four']    Node 85  =  ['one']
Node 36  =  ['two']     Node 86  =  ['nine']
Node 37  =  ['five']    Node 87  =  ['six']
Node 38  =  ['six']     Node 88  =  ['three']
Node 39  =  ['seven']   Node 89  =  ['five']
Node 41  =  ['eight']   Node 91  =  ['three']
Node 42  =  ['five']    Node 92  =  ['four']
Node 43  =  ['nine']    Node 93  =  ['five']
Node 44  =  ['seven']   Node 94  =  ['two']
Node 45  =  ['six']     Node 95  =  ['eight']
Node 46  =  ['one']     Node 96  =  ['six']
Node 47  =  ['four']    Node 97  =  ['one']
Node 48  =  ['two']     Node 98  =  ['seven']
Node 49  =  ['three']   Node 99  =  ['nine']
Node 51  =  ['four']
Node 52  =  ['two']
Node 53  =  ['six']
Node 54  =  ['eight']
Node 55  =  ['five']
Node 56  =  ['three']
Node 57  =  ['seven']
Node 58  =  ['nine']
Node 59  =  ['one']
```

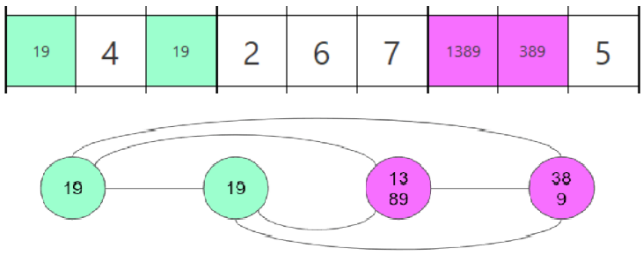
***Gambar 3.6**: Hasil dari test case sesuai dengan gambar 1.1*

Dapat dilihat bahwa setelah itearasi ke sembilan, hanya ada satu kemungkinan angka pada setiap sel dan apabila kita lakukan validasi terhadap solusi dari teka-teki pada gambar 1.1 kita akan mengetahui bahwa solusi pada gambar 3.5 adalah solusi yang valid.

Untuk teka-teki sudoku yang lebih rumit, metode ini tidak dapat memprediksi secara tepat angka apa yang harus dimasukkan dalam sebuah sel. Sebagai contoh, akan dilakukan pengujian terhadap sebuah teka-teki seperti pada gambar 3.7. Apabila dimasukkan teka-teki tersebut dalam program, hasil yang akan dikeluarkan hanya sebatas angka apa saja yang mungkin masuk dalam sebuah sel. Untuk itu, diperlukan proses tambahan untuk dapat menghasilkan angka yang tepat. Salah satu proses yang dapat dilakukan adalah dengan metode tuple. Pada konteks ini, tuple didefinisikan sebagai sebuah subset dari suatu baris, kolom, atau subtabel dengan ukuran n dan memiliki tepat n buah angka unik. Jadi di dalam sebuah kolom, baris, atau subtabel permainan sudoku, semua kandidat angka yang ada di dalam tuple T dapat dieliminasi apabila terdapat di luar T. Untuk menggunakan metode ini, diperlukan sebuah graf baru dengan himpunan simpul-simpulnya berisikan sel-sel yang belum memiliki isi yang pasti—sel sel yang memiliki lebih dari satu buah kemungkinan—lalu menghubungkan semua simpul-simpul tersebut apabila diantara simpul memiliki setidaknya satu kandidat yang sama.

					1	4	7	
		9		3	2		8	
6								
			3	8				4
	8		6					2
			2		5	3		
	3	7		4		6		9
2								
								1

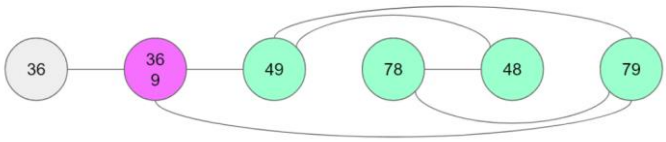
***Gambar 3.7**: Permasalahan sudoku yang lebih rumit. Sumber: https://sudoku-solver.rakhman.info/ diakses pada 10 Desember 2022, pukul 21.55.*



***Gambar 3.8**: Metode tuple dalam eliminasi kandidat angka. Sumber: https://rakhman.info/blog/solving-sudoku-with-graph-theory/ diakses pada 10 Desember 2022, pukul 22:11.*

***Gambar 3.9**: Tuple yang sudah tidak signifikan. Sumber: https://rakhman.info/blog/solving-sudoku-with-graph-theory/ diakses pada 10 Desember 2022, pukul 22:11.*

Pada gambar 3.8 terdapat tuple 19 yang terhubung oleh komponen graf yang lebih besar yakni terhadap simpul berwarna ungu. Berdasarkan definisi tuple dan teknik eliminasi yang sudah didefinisikan, kita dapat mengeliminasi kandidat 1 dan 9 pada komponen berwarna ungu sehingga tercipta sebuah graf seperti pada gambar 3.9. Metode pengeliminasian ini dapat dilakukan tidak hanya pada tuple namun juga pada quadruple seperti pada gambar 3.10. Pada gambar 3.10, terdapat quadruple 4789 yang terhubung ke simpul 369. Pada kasus ini kita dapat mengeliminasi angka 9 dari simpul 369 dan menghapus sisi yang beririsan dari simpul 369 ke komponen quadruple. Apabila kita mengiterasi terus menerus pada setiap kolom, baris, dan upatabel, pada akhirnya kita akan mendapatkan solusi unik dari sebuah teka-teki sudoku.



***Gambar 3.10**: Contoh penerapan eliminasi berdasarkan quadruple 4789. Sumber: https://rakhman.info/blog/solving-sudoku-with-graph-theory/ diakses pada 10 Desember 2022 pukul 22:38 WIB.*

Setelah dilakukan eliminasi, akan didapatkan sebuah graf kosong yang tidak memiliki sisi. Setiap simpul graf dalam baris, kolom, atau upatabel yang sama memiliki komponen yang unik dan hasil dari teka-teki sudoku diharapkan merupakan hasil yang valid. Hasil yang didapat setelah memasukkan sudoku pada gambar 3.7 dapat dilihat pada gambar 3.11.

8	2	3	9	6	1	4	7	5
7	5	9	4	3	2	1	8	6
6	4	1	8	5	7	2	9	3
1	6	2	3	8	9	7	5	4
3	8	5	6	7	4	9	1	2
9	7	4	2	1	5	3	6	8
5	3	7	1	4	8	6	2	9
2	1	6	5	9	3	8	4	7
4	9	8	7	2	6	5	3	1

***Gambar 3.11**: Hasil permainan sudoku setelah menerapkan metode eliminasi tuple terhadap kandidat angka. Sumber: https://sudoku-solver.rakhman.info/?digits=823961475+759432186+641857293+162389754+385674912+974215368+537148629+216593847+498726531 diakses pada 10 Desember 2022 pukul 22:44 WIB.*

IV. KESIMPULAN

Graph theory dan *graph coloring* merupakan salah satu materi dari mata kuliah Matematika Diskrit yang memiliki berbagai macam kegunaan dalam kehidupan sehari-hari. Contoh dari pengaplikasian graf adalah untuk mencari solusi dari teka-teki seperti sudoku. Sudoku merupakan permainan angka yang sudah popular dari tahun 1980-an hingga sekarang karena terkesan sederhana dan menarik. Tergantung dari tingkat kesulitan teka-tekinnya, graf dalam konteks ini dapat mencari kemungkinan angka-angka yang dapat masuk dari suatu sel pada tabel sudoku atau dalam tingkat kesulitan rendah, kita bahkan dapat mendapatkan suatu solusi dari teka-teki tersebut. Namun, untuk permasalahan teka-teki yang lebih rumit, yakni teka-teki yang memiliki angka inisial lebih sedikit, kita memerlukan proses tambahan untuk mengeliminasi semua komponen bertetangga yang akan memiliki elemen yang sama dengan metode tuple.

V. UCAPAN TERIMA KASIH

Puji syukur dan terimakasih penulis sampaikan terhadap Tuhan Yang Maha Esa karena berkat rahmat-Nya, penulis mampu menyelesaikan tugas berupa makalah ini dengan baik dan tepat waktu. Penulis makalah juga mengucapkan terimakasih yang sebesar-besarnya kepad Bu Fariska Zakhralativa Rukanda, S.T., M.T. selaku dosen pembinbing mata kuliah Matematika Diskrit. Penulis juga ingin mengucapkan terimakasih kepada semua pihak yang terlibat dalam pembuatan makalah "Penerapan Graph Theory dan Graph Coloring dalam Memecahkan Teka-teki Sudoku" ini atas segala bantuan dan dukungannya selama proses pembuatan tugas.

REFERENSI

[1] Munir, Rinaldi. Graf Bagian I. Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf pada 9 Desember 2022 pukul 17:15.

[2] Munir, Rinaldi. Graf Bagian II. Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf pada 9 Desember 2022 pukul 17:20.

[3] Munir, Rinaldi. Graf Bagian III. Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf pada 9 Desember 2022 pukul 17:21.

[4] Mudiyanto, Febi, "Solve Graph Coloring Problem with Greedy Algorithm and Python". Diakses melalui https://python.plainenglish.io/solve-graph-coloring-problem-with-greedy-algorithm-and-python-6661ab4154bd pada 9 Desember 2022 pukul 18:15 WIB.

[5] Iñiguez, Agustín, "Graph theory and its uses with 5 examples of real life problems". Diakses melalui https://www.xomnia.com/post/graph-theory-and-its-uses-with-examples-of-real-life-problems/ pada 9 Desember 2022 pukul 16.03 WIB.

[6] Rakhman, Kirill, "Soving Sudoku with Graph Theory". Diakses melalui https://rakhman.info/blog/solving-sudoku-with-graph-theory/ pada 9 Desember 2022 pukul 19.17 WIB.

[7] Smith, David, "So you thought Sudoku came from the Land of the Rising Sun …". Diakses melalui https://www.theguardian.com/media/2005/may/15/pressandpublishing.usnews#:~:text=The%20Sudoku%20story%20began%20in,in%20each%20row%20or%20column. Pada 9 Desember 2022 pukul 22:01.

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, 3 Desember 2020

Muhammad Zaki Amanullah 13521146