Penerapan Operasi Teori Himpunan dan Teori Graf pada Algoritma Constraint Propagation untuk Penyelesaian Sudoku

Mochammad Adhitya Nur Rohman - 13525038
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: adityanr199@gmail.com , 13525038@std.stei.itb.ac.id

***Abstract*—Penyelesaian Sudoku 9x9 merupakan representasi klasik dari Constraint Satisfaction Problem (CSP) dalam Ilmu Komputer. Penelitian ini bertujuan untuk mengimplementasikan dan membandingkan kinerja dua paradigma komputasi: algoritma Constraint Propagation (CP) berbasis Teori Himpunan dan Teori Graf, dengan algoritma Backtracking (Depth-First Search). Evaluasi komparatif dilakukan melalui bahasa javascript dengan menguji matriks dari tingkat kesulitan Easy hingga Evil. Parameter performa dievaluasi berdasarkan efisiensi waktu eksekusi dan jumlah langkah komputasi. Hasil pengujian membuktikan adanya trade-off yang signifikan antara efisiensi waktu dan kompleksitas program. Pada Sudoku tingkat rendah (Easy dan Medium), Backtracking unggul akibat ketiadaan overhead struktur data. Namun, ketiadaan pemangkasan logika membuat DFS mengalami ledakan kombinatorial pada Sudoku tingkat tinggi (Hard), meroket hingga lebih dari 60.000 iterasi dan membengkak menjadi 58 ms. Sebaliknya, mesin CP memamerkan stabilitas asimtotik yang superior, menyelesaikan papan secara konsisten dalam 43 hingga 63 langkah, serta mengungguli DFS pada level Hard dengan waktu 18,33 ms. Meskipun CP memenuhi standar penalaran transparan (Explainable AI), sistem ini menemui batasan berupa kondisi jalan buntu pada konfigurasi Evil. Penelitian ini menyimpulkan bahwa arsitektur komputasi yang paling optimal untuk teka-teki logika adalah sistem hibrida: memosisikan Constraint Propagation sebagai mesin pencarian utama, dan menggunakan Backtracking secara murni sebagai mekanisme jaring fallback mechanism.**

Kata kunci—sudoku; teori himpunan; teori graf; constraint propagation; backtracking

I. Pendahuluan

Sudoku merupakan permainan teka-teki logika kombinatorial pada matriks 9x9. Tujuan dari permainan ini adalah mengisi angka 1 hingga 9 pada masing masing kotak sehingga tidak ada pengulangan angka pada baris, kolom, maupun box 3x3 yang sama. Secara teoritis, penyelesaian ruang masalah Sudoku diklasifikasikan ke dalam kelas NP-Complete. Kompleksitas komputasional inilah yang menjadikan Sudoku menarik untuk dieksplorasi.

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

Gambar 1.1: Contoh tabel sudoku

Sumber: https://sudokutable.com/howtoplay/?lang=id

Eksplorasi metode penyelesaian Sudoku pernah dibahas oleh beberapa makalah sebelumnya. Amanullah (2022) menggunakan *graph theory* dan *graph coloring* untuk menyelesaikan Sudoku. Pendekatan tersebut mampu menyelesaikan Sudoku pada level dasar, namun akan gagal untuk Sudoku dengan tingkat yang lebih sulit. Tambunan (2024) berfokus pada hubungan Sudoku dengan permutasi, tetapi pada makalah tersebut terdapat pendekatan lain untuk menyelesaikan Sudoku yaitu, teknik *brute force*. Teknik *brute force* pasti dapat memberikan solusi untuk semua tingkat kesulitan sudoku, namun teknik ini memiliki kompleksitas waktu yang sangat besar yaitu O(9^n) dengan n adalah jumlah sel kosong.

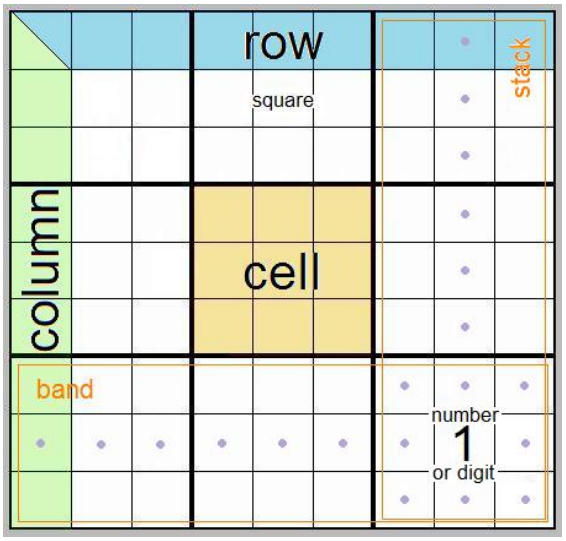
Untuk mengatasi permasalahan tersebut pada makalah ini akan dikembangkan program penyelesaian Sudoku yang berbasis *constraint propagation.* Program ini dibuat dengan memanfaatkan dua pilar matematika diskrit, yaitu teori himpunan untuk manipulasi kandidat awal (teknik *hidden single, naked single, hidden pair, naked pair* dan *locked candidates*) dan teori graf untuk manipulasi kandidat tingkat

lanjut (teknik *y-wing, x-wing, skyscraper, crane*). Program ini nantinya akan diuji dengan berbagai tingkat kesulitan Sudoku untuk mengukur tingkat keberhasilan dan kompleksitas waktu program.

II. Landasan Teori

A. Tinjauan matematis dan keunikan solusi

Ruang permainan Sudoku direpresentasikan oleh sebuah matriks persegi 9x9 yang dibagi lebih kecil lagi menjadi sembilan box 3x3. Pada kondisi awal matriks ini akan diisi oleh beberapa angka petunjuk. Sebuah rancangan teka-teki Sudoku yang valid secara matematis harus memiliki tepat satu solusi tunggal. Apabila sebuah konfigurasi awal memungkinkan terjadinya lebih dari satu solusi yang valid, Sudoku tersebut diklasifikasikan cacat. Syarat keunikan inilah yang memberikan justifikasi matematis bahwa teka-teki dapat diselesaikan secara murni menggunakan deduksi logika tanpa memerlukan tebakan.



Gambar 2.1: Struktur papan sudoku

Sumber: https://urbanrim-org-uk.translate.goog/sudoku.htm?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=imgs

B. Tingkat kesulitan sudoku

Tingkat kesulitan komputasional Sudoku sering kali disalahpahami hanya bergantung pada jumlah sel yang terisi. Padahal, tingkat kesulitan Sudoku dinilai bukan sekadar dari minimnya angka awal, melainkan dari kedalaman teknik yang diperlukan untuk menyelesaikan sudoku.

SE RATING	DIFFICULTY LEVEL	TECHNIQUES REQUIRED
1.0 – 2.0	Trivial	Naked singles only
2.0 – 3.0	Easy	Hidden singles
3.0 – 4.0	Medium	Naked pairs, pointing pairs
4.0 – 5.0	Hard	X-Wing, Swordfish
5.0 – 7.0	Very Hard	XY-Wing, simple chains
7.0 – 9.0	Expert	Complex chains, ALS
9.0 – 11.0	Extreme	Forcing chains, nested chains
11.0+	Ultra-Extreme	Beyond standard technique catalogs

Gambar 2.2: Tingkat kesulitan sudoku dilihat dari *SE Rating*

Sumber: https://sudokupulse.com/articles/worlds-hardest-sudoku-puzzles/#can-humans-solve-the-worlds-hardest-sudoku-puzzles-without-guessing

Sebagai contoh, matriks AI Escargot (2006) yang diciptakan oleh matematikawan Finlandia, Arto Inkala, secara akademis diakui sebagai salah satu teka-teki Sudoku tersulit di dunia. Meskipun Sudoku tersebut memuat 21 angka petunjuk awal. Kesulitannya melebihi teka-teki standar karena teknik dasar secara instan menemui jalan buntu dan mendapatkan *rating* 11.0 pada skala Sudoku Explainer (SE). Sudoku dengan tingkat ekstrem seperti ini memaksa pemain untuk menavigasi kompleksitas relasi graf jarak jauh secara simultan dan merangkai teknik inferensi tingkat lanjut secara tumpang tindih. Hal inilah yang membedakan penyelesaian logika deduktif dengan algoritma *brute force*.

C. Pemodelan teori himpunan

Matriks Sudoku dimodelkan sebagai sekumpulan himpunan. Misalkan terdapat himpunan semesta U untuk seluruh kemungkinan nilai sel, yang didefinisikan sebagai U = {1, 2, 3, 4, 5, 6, 7, 8, 9}. Untuk setiap sel kosong pada koordinat baris r dan kolom c, dibentuk sebuah Himpunan Kandidat Kr,c di mana pada tahap inisialisasi awal berlaku Kr,c = U.

Constraint propagation tingkat dasar beroperasi melalui Operasi Selisih Himpunan (A - B). Jika sebuah sel tetangga pada ruang unit yang sama (baris, kolom, atau box) telah bernilai pasti x, sistem secara konstan mengeksekusi penghapusan elemen: Kr,c = Kr,c - {x}. Untuk reduksi tingkat lanjut seperti pencarian Naked Pair, program juga memanfaatkan operasi Irisan (A ∩ B) dan Gabungan (A ∪ B) guna mengurangi banyak anggota himpunan kandidat pada sel. Syarat bagi sebuah sel untuk mencapai kondisi selesai (solved) didasarkan pada Kardinalitas Himpunan (|A|). Apabila rentetan operasi himpunan menghasilkan |Kr,c| = 1, maka satu-satunya elemen yang tersisa adalah nilai jawaban yang benar untuk kotak tersebut.

D. Pemodelan teori graf

Ketika operasi himpunan tidak lagi menghasilkan perubahan (semua sisa sel memiliki |K| >= 2), pendekatan lanjutan dilakukan dengan Teori Graf. Teori graf disini akan digunakan untuk merepresentasikan Kandidat Angka sebagai

himpunan simpul (V) dan Relasi Logis antar kandidat sebagai himpunan sisi (E).Pendekatan ini membentuk lintasan graf yang dikenal dengan *Alternating Inference Chains* (AIC). sisi E pada AIC dibedakan menjadi dua jenis relasi:

1. Tautan Kuat (*Strong Link*): Direpresentasikan sebagai sisi berbobot khusus yang terjadi ketika hanya terdapat tepat dua sel memuat kandidat angka tertentu di dalam sebuah unit. Berdasarkan logika disjungtif, jika simpul pertama bernilai Salah, simpul kedua mutlak bernilai Benar.
2. Tautan Lemah (*Weak Link*): Terjadi antar kandidat di dalam unit yang memuat lebih dari dua kandidat serupa. Relasi ini menyatakan bahwa jika simpul pertama Benar, simpul yang saling bertetangga mutlak Salah.

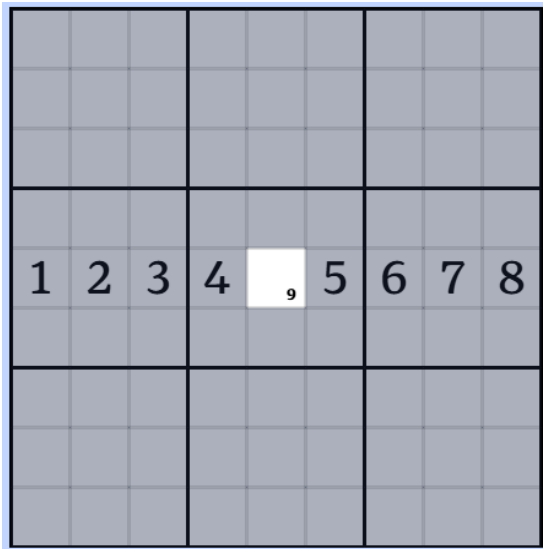
Dengan melacak lintasan graf yang berselang-seling antara *Strong Link* dan *Weak Link*, program dapat mengimitasi teknik tingkat lanjut pada Sudoku (seperti *Skyscraper, X-Wing* dan *Y-wing*). Formasi sirkuit dan lintasan graf ini memungkinkan algoritma deduktif untuk mengeliminasi kandidat pada simpul persimpangan, sehingga mencegah program *stuck* dari operasi himpunan sebelumnya.

III. Implementasi Program

Sistem simulasi penyelesaian Sudoku deduktif ini dikembangkan menggunakan bahasa pemrograman JavaScript. Setiap sel kosong pada ruang matriks 9x9 diinstansiasi sebagai objek struktur data Set(). Set() digunakan untuk menjamin bahwa tidak akan ada elemen duplikat pada himpunan kandidat Kr,c. Set() dipilih karena telah memiliki beberapa *method* yang membantu untuk melakukan manipulasi elemen seperti .has(x) untuk mengecek apakah himpunan memiliki suatu elemen x, .delete(x) untuk menghapus elemen x dari himupunan, .size untuk operasi kardinalitas dan sebagainya. Sehingga kita bisa memangkas kompleksitas waktu pencarian dan penghapusan kandidat menjadi O(1).

Setelah Sudoku berhasi diinstansiansi program akan melanjutkan dengan langkah-langkah berikut

A. *Naked Single*

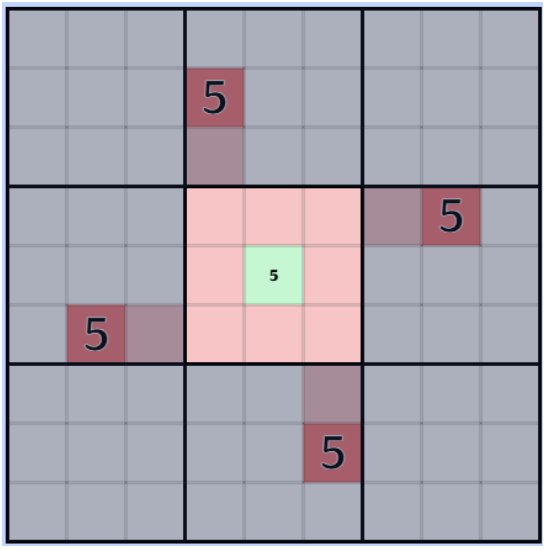


Gambar 3.1: Contoh *Naked Single* pada *row* 5 dan *col* 5

Sumber: https://sudoku.coach/en/learn/naked-single

Naked Single adalah kondisi ketika sel hanya memiliki satu buah kandidat yang tersisa.

B. *Hidden Single*

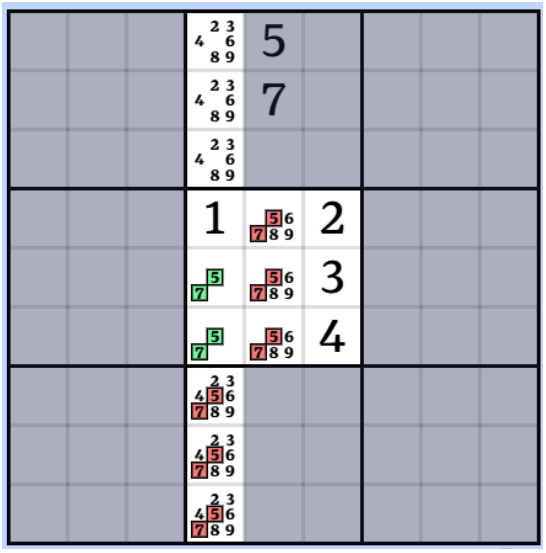


Gambar 3.2: Contoh *Hidden Single* pada *row* 5 dan *col* 5

Sumber: https://sudoku.coach/en/learn/hidden-single

Hidden Single adalah kondisi dimana hanya ada satu sel pada satu baris, kolom atau box yang dapat ditempati oleh suatu angka.

C. Naked Pair

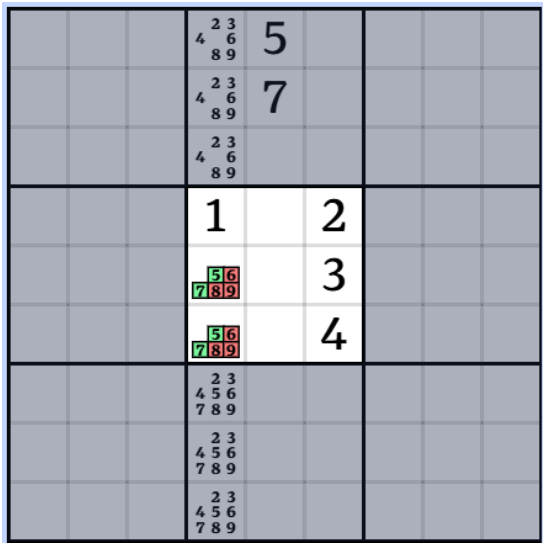


Gambar 3.3: Contoh *Naked pair* pada *row* 5 dan *col* 4 dan *row* 6 dan *col* 4

Sumber: https://sudoku.coach/en/learn/naked-pair

Naked Pair adalah kondisi ketika dua sel memiliki tepat dua kandidat yang sama dalam satu baris, kolom, ataupun box. Dengan ini kita bisa melakukan eliminasi anggota *Naked Pair* pada baris, kolom, atau box yang sama (lihat kotak merah pada gambar 3.3).

D. Hidden Pair

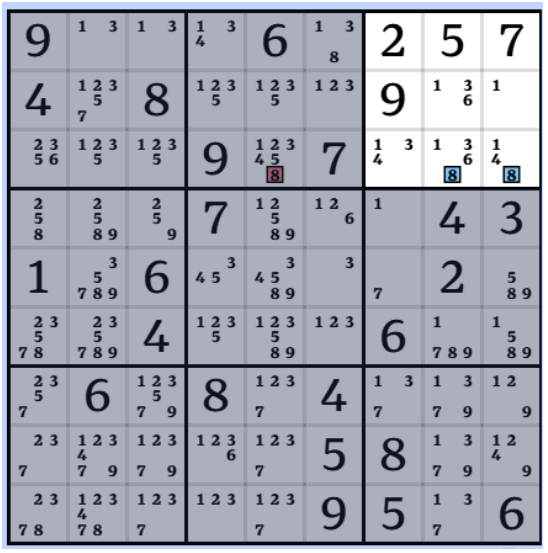


Gambar 3.4: Contoh *Hidden Pair* pada *row* 5 dan *col* 4 dan *row* 6 dan *col* 4

Sumber: https://sudoku.coach/en/learn/hidden-pair

Hidden Pair adalah kondisi ketika ada tepat dua angka yang telah tereliminasi dari seluruh kandidat pada baris, kolom ataupun box kecuali pada tepat dua sel. Dengan ini kita bisa menghapus kandidat lain pada kedua sel tersebut (lihat kotak merah pada gambar 3.4).

E. Locked Candidate

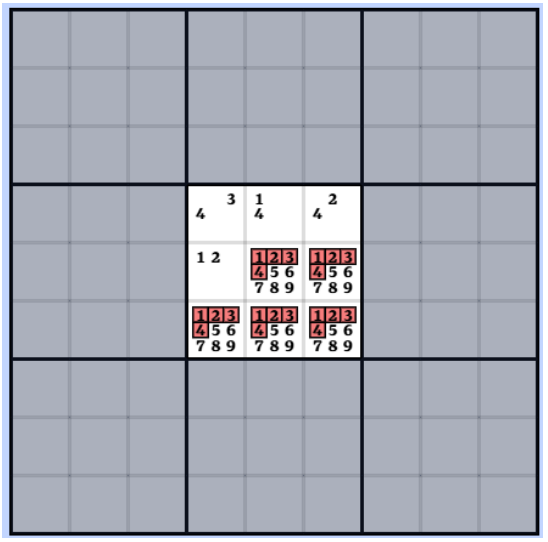


Gambar 3.5: Contoh *Locked Candidate* pada *row* 3 dan *col* 8 dan row 3 dan col 9

Sumber: https://sudoku.coach/en/learn/locked-candidate

Locked Candidate adalah kondisi ketika suatu angka terkunci pada suatu baris, kolom, ataupun box. Pada gambar 3.5 angka 8 terkunci pada box kanan atas, sehingga kita bisa melakukan eliminasi kandidat 8 pada baris 3 kolom 5.

F. Disjoint Group

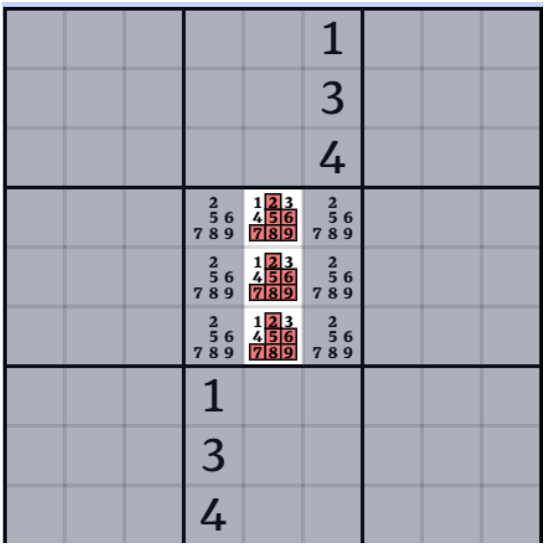


Gambar 3.6: Contoh *Disjoint Group* pada *row* 4 dan *col* 4, *row* 4 dan *col* 5, *row* 4 dan *col* 6 dan *row* 5 dan *col* 4

Sumber: https://sudoku.coach/en/learn/disjoint-groups

Disjoint Group adalah kondisi ketika n sel memiliki tepat n buah kandidat yang sama dalam satu baris, kolom, ataupun box untuk union ke semua n sel tersebut. Dengan ini kita bisa melakukan eliminasi anggota *Disjoint Group* pada baris, kolom, atau box yang sama (lihat kotak merah pada gambar 3.6).

G. Hidden Group

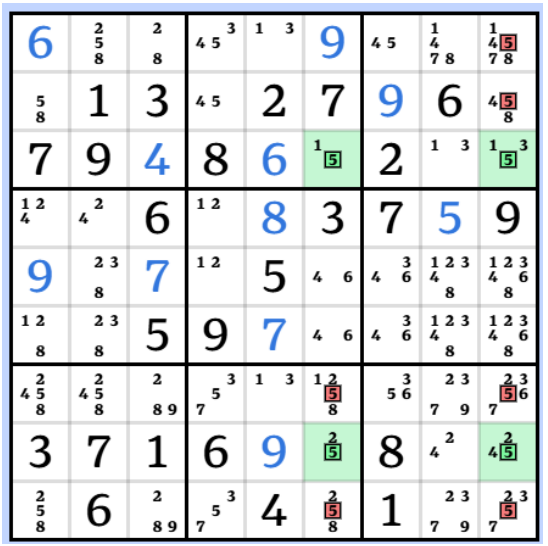


Gambar 3.7: Contoh *Hidden Group (3)* pada *row* 4 dan *col* 5, *row* 5 dan *col* 5, dan *row* 6 dan *col* 5

Sumber: https://sudoku.coach/en/learn/hidden-groups

Hidden Group adalah kondisi ketika ada lebih dari dua angka yang telah tereliminasi dari seluruh kandidat pada baris, kolom ataupun box kecuali pada lebih dari dua sel. Dengan ini kita bisa menghapus kandidat lain pada semua sel tersebut (lihat kotak merah pada gambar 3.7).

H. X-Wing

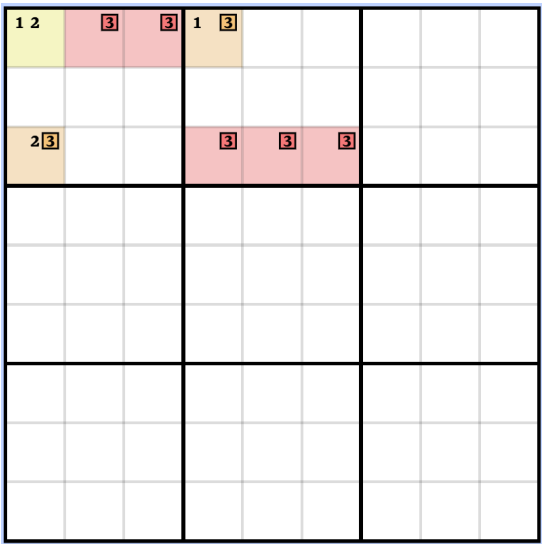


Gambar 3.8: Contoh *X-Wing* pada *row* 3 dan *col* 6, *row* 3 dan *col* 9, *row* 8 dan *col* 6 dan *row* 8 dan *col* 9

Sumber: https://sudoku.coach/en/learn/x-wing

X-Wing adalah kondisi ketika ada dua buah baris atau kolom yang memiliki tepat dua posisi yang paralel untuk salah satu kandidat angka. Pada gambar 3.8 baris 3 dan baris 8 keduanya memiliki dua kandidat angka 5 yang saling pararel pada kolom 6 dan kolom 9, sehingga kita bisa melakukan eliminasi angka 5 pada semua kolom 6 dan kolom 9, kecuali pada baris 3 dan baris 8.

I. Y-Wing



Gambar 3.9: Contoh *Y-Wing* pada *row* 1 dan *col* 1, *row* 1 dan *col* 4, dan *row* 3 dan *col* 1

Sumber: https://sudoku.coach/en/learn/y-wing

Y-Wing adalah kondisi ketika terdapat 3 buah sel yang masing masing memiliki dua buah kandidat, jumlah elemen dari union ketiganya adalah 3 dan berbentuk bengkok. Dengan ini kita bisa melakukan eliminasi anggota *Y-Wing* yang berada diujung (pada gambar 3.9 adalah 3) pada baris, kolom dan box yang beririsan antara baris 1 kolom 4 dan baris 3 kolom 1.

J. SkyScrapper

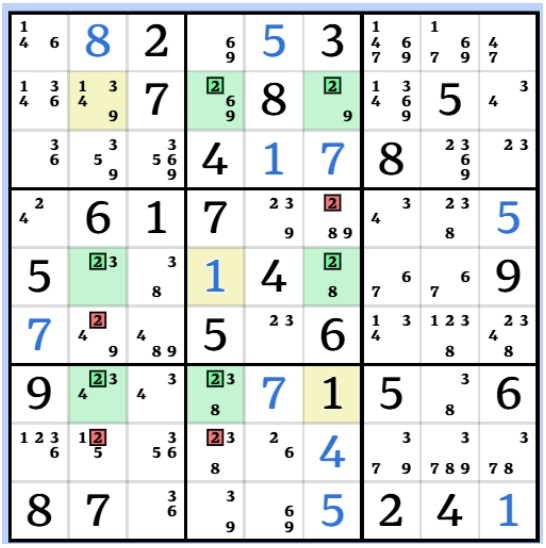
Gambar 3.10: Contoh *SkyScrapper* pada *row* 2 dan *col* 3, *row* 3 dan *col 7, row* 8 dan *col* 3, dan *row* 6 dan *col* 7

Sumber: https://sudoku.coach/en/learn/skyscraper

SkyScrapper adalah kondisi yang hampir sama dengan *X-Wing,* hanya saja salah satu dari selnya tidak sejajar. Dengan ini kita bisa melakukan eliminasi pada baris, kolom, dan box

irisan dari atap *SkyScrapper* (pada gambar 3.10 berwarna biru).

K. SwordFish

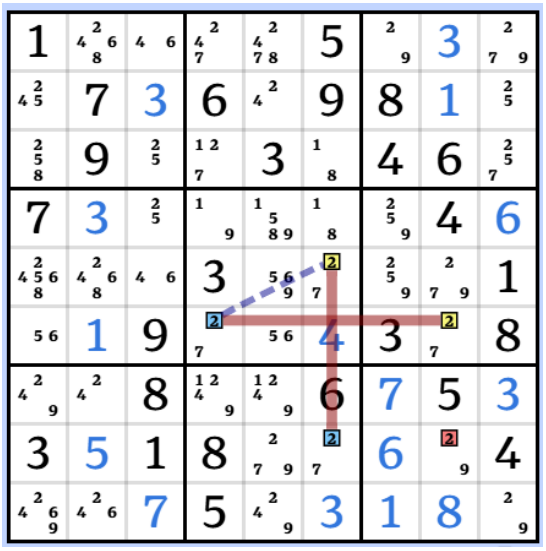


Gambar 3.11: Contoh *SwordFish* pada *row* 2 dan *col* 2, *row* 2 dan *col* 4, *row* 2 dan *col* 6, *row* 4 dan *col* 2, *row* 4 dan *col* 4, *row* 4 dan *col* 6, *row* 6 dan *col* 2, *row* 6 dan *col* 4, *row* 6 dan *col* 6

Sumber: https://sudoku.coach/en/learn/swordfish

Swordfish adalah kondisi yang hampir sama juga dengan kondisi *X-Wing,* hanya saja *SwordFish* memiliki ukuran 3x3 dan dapat mengeliminasi kandidat pada 3 baris dan 3 kolom juga.

L. Two-String-Kite

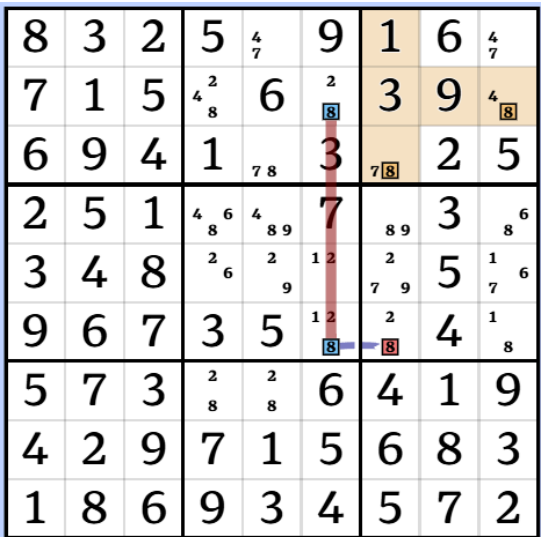


Gambar 3.12: Contoh *Two-String-Kite* pada *row* 5 dan *col* 6, *row* 6 dan *col* 4, *row* 6 dan *col* 8, dan *row* 8 dan *col* 6

Sumber: https://sudoku.coach/en/learn/two-string-kite

Two-String-Kite adalah kondisi ketika ada 1 baris dan 1 kolom yang memiliki tepat 2 posisi untuk salah satu kandidat angka, dimana salah satu dari sel pada baris dan kolom tersebut harus berada pada satu box yang sama (pada gambar 3.12 baris 5 kolom 6 dan baris 6 kolom 4), sedangkan yang lainnya berada pada dua box berbeda (pada gambar 3.12 baris 6 kolom 8 dan baris 8 kolom 6). Dengan ini kita bisa melakukan eliminasi pada irisan dari kolom sel luar dari baris yang sama dengan baris sel luar dari kolom yang sama (pada gambar 3.12 baris 8 kolom 8).

M. Empty Rectangle



Gambar 3.13: Contoh *Empty Rectangle*

Sumber: https://sudoku.coach/en/learn/empty-rectangle

Empty Rectangle akan terjadi ketika beberapa syarat berikut terpenuhi:

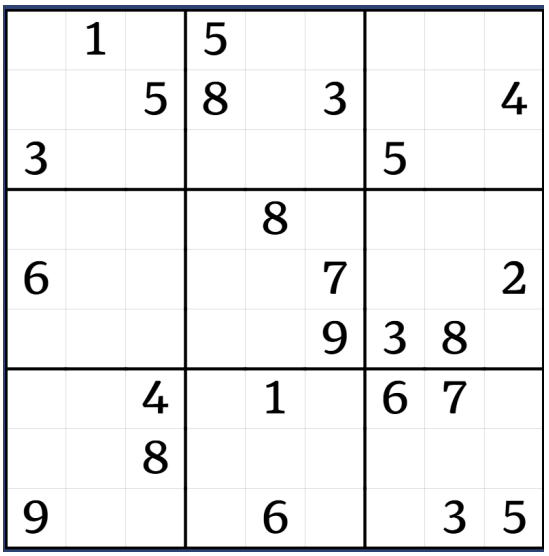
1). Terdapat satu box yg memiliki kandidat angka sebanyak 5 sel dengan ketentuan baris dan kolom yang memiliki kandidat tersebut harus tegak lurus (lihat kotak yang berwarna oranye pada gambar 3.13).

2). Terdapat satu ikatan kuat pada kandidat diluar box utama (pada gambar 3.13 ada pada baris 2 kolom 6 dengan baris 6 kolom 6) yang berarti jika salah satu dari sel tersebut salah maka sel lainnya dapat dipastikan kebenarannya.

3). 3 sel yang diluar harus berhasil membentuk persegi panjang dengan baris atau kolom box yang dijelaskan pada nomor 1.

Jika ketiga syarat diatas sudah terpenuhi kita dapat mengeliminasi kandidat yang tidak memiliki strong link dari ketiga sel diluar box utama.

Dengan menggunakan ke 14 teknik diatas program berhasil menyelesaikan Sudoku dengan tingkat kesulitan evil atau dengan SE Rating 6.00-8.00.



Gambar 3.14: Contoh *Evil* level sudoku

Sumber: https://sudokupulse.com/evil/

Program ini dapat menyelesaikan Sudoku pada gamabar 3.14 dalam waktu 16.30ms, sedangkan teknik *backtracking* membutuhkan waktu sebesar 61.50ms. Selain dengan Sudoku pada gambar 3.14 kedua metode tersebut juga diuji dengan Sudoku lain dengan level yang berbeda-beda.

Tabel 1. Tabel pengujian kecepatan teknik *Constraint Propagation* (CP) dan *Backtracking (B)*

kode	level	*Time step(1)*	*Time step(2)*	*Time step(3)*	*mean*
CP	easy	4.30 ms 43 step	5.60 ms 44 step	3.10 ms 44 step	4.33 ms 43.6 step
B	easy	1.80ms 109 step	2.70ms 117 step	0.70 ms 48 step	4.73 ms 91.3 step
CP	medium	3.50 ms 46 step	3.00 ms 46 step	3.80 ms 41 step	3.43 ms 44.3 step
B	medium	2.10 ms 128 step	4.90 ms 751 step	1.20 ms 149 step	2.73 ms 342.6 step
CP	hard	8.20 ms 57 step	8.90 ms 59 step	37.9 ms 61 step	18.33 ms 59 step
B	hard	94.0 ms 127792 step	34.8 ms 17936 step	45.2 ms 34474 step	58 ms 60067.3 step
CP	evil	Gagal	36.6 ms 61 step	23.2 ms 66 step	29.9 ms 63.5 step
B	evil	1.60 ms 5593 step	12.00 ms 6055 step	18.90 ms 10626 step	108.33 ms 7424.6 step

Berdasarkan Tabel 1, yang membedakan kinerja kedua algoritma terletak pada jumlah langkah yang dibutuhkan untuk mencapai goal state. Algoritma Constraint Propagation (CP) menunjukkan stabilitas yang luar biasa; terlepas dari tingkat kesulitan sudoku (dari Easy hingga Evil), program ini hanya membutuhkan antara 43 hingga 63 langkah konklusif. Hal ini terjadi karena CP memangkas ruang pencarian menggunakan evaluasi Teori Himpunan dan Graf sehingga tidak melakukan pencabangan.

Sebaliknya, algoritma *backtracking* (B) sangat rentan terhadap Ledakan Kombinatorial (Combinatorial Explosion). Pada level Easy, *backtracking* hanya membutuhkan rata-rata 91,3 langkah tebakan. Namun pada level Hard, jumlah rekursi meledak secara eksponensial mencapai rata-rata 60.067 langkah (dengan kasus terburuk mencapai 127.792 langkah pada seed pertama). Ini membuktikan kelemahan fatal dari Depth-First Search (DFS) primitif saat menghadapi ruang pencarian dengan tingkat konstrain yang minim.

Skenario terbaik (Level Easy & Medium): Algoritma *backtracking* (rata-rata 2,73 ms - 4,73 ms) memiliki performa waktu yang sedikit mengungguli atau bersaing ketat dengan algoritma CP (rata-rata 3,43 ms - 4,33 ms). Hal ini terjadi karena DFS hanya menggunakan instruksi primitif tingkat rendah (low-level array checking), sementara CP membutuhkan memori tambahan untuk instansiasi objek Set().

Skenario terburuk (Level Hard): Waktu eksekusi rata-rata *backtracking* meroket menjadi 58 ms, sementara algoritma CP menyelesaikan matriks secara jauh lebih efisien dalam 18,33 ms. Ini mengonfirmasi bahwa kompleksitas polinomial dari algoritma deduktif akan selalu mengalahkan kompleksitas eksponensial dari DFS pada skenario ekstrem.

Karena program hanya menggunakan 14 teknik Sudoku saja, pasti akan ada saatnya teknik yang ada akan gagal semua (seperti pada percobaan level *evil* pertama). Kegagalan ini membuktikan bahwa ada konfigurasi Sudoku di tingkat Evil yang sengaja dirancang secara matematis kebal terhadap metode eliminasi graf standar, sehingga memaksa algoritma memasuki kondisi jalan buntu. Sebagai langkah penyempurnaan sistem di masa mendatang, penggunaan *backtracking* tetap krusial untuk diintegrasikan sebagai langkah validasi akhir hanya saat program CP telah menyerah.

Meskipun Tabel 1 menggunakan istilah yang sama pada jumlah langkah (*step*) dari kedua algoritma, definisi satu

langkah komputasi pada *backtracking* berbeda dengan satu langkah pada *Constraint Propagation*.

Pada algoritma *backtracking*, satu langkah didefinisikan sebagai satu kali percobaan penempatan angka (*node visit* pada *state space tree*). Sistem hanya menempatkan angka, melakukan iterasi pengecekan baris/kolom/box, dan bergerak maju (*push*) atau mundur mencabut angka (*pop*). Karena kesederhanaan komputasinya, satu langkah DFS hanya memakan hitungan sepersekian milidetik. Namun, sifatnya yang beroperasi sebagai blind search murni membuat sistem harus "menebak" puluhan ribu kali pada matriks dengan tingkat kesulitan tinggi, yang memicu lonjakan jumlah langkah secara eksponensial.

Sebaliknya, pada algoritma CP, satu langkah merepresentasikan satu siklus operasi deduksi logis. Sistem harus memindai seluruh 27 unit matriks, menghitung kardinalitas struktur data Set, memetakan *Adjacency List* untuk mencari hubungan, dan mengeksekusi operasi himpunan (seperti irisan atau selisih) untuk menghapus kandidat yang tidak valid secara serentak.

Penjelasan ini menjawab mengapa waktu eksekusi rata-rata per langkah pada CP jauh lebih lambat dibandingkan *backtracking*, namun CP mampu menyelesaikan Sudoku hanya dengan puluhan langkah. Satu langkah CP mampu meniadakan ribuan potensi langkah tebakan DFS di masa depan karena ruang pencariannya telah dipangkas secara matematis. Oleh karena itu, CP merepresentasikan "efisiensi kecerdasan penalaran", sedangkan *backtracking* merepresentasikan "kecepatan iterasi coba-coba". ”

IV. Kesimpulan

Berdasarkan hasil pengujian komparatif pada penyelesaian Sudoku 9x9, algoritma *Constraint Propagation* (CP) terbukti jauh lebih superior dalam memangkas ruang pencarian dibandingkan algoritma *backtracking* (DFS). Satu langkah komputasi pada algoritma CP merupakan siklus deduksi makro yang memungkinkan penyelesaian secara stabil hanya dalam 43 hingga 63 langkah. Sebaliknya, pendekatan DFS sangat rentan terhadap ledakan kombinatorial, dimana ketiadaan pemangkasan dengan logika memaksa sistem melakukan lebih dari 60.000 iterasi tebakan pada Sudoku tingkat tinggi (level *Hard*). Perbedaan mekanisme pencarian ini menghasilkan trade-off yang terukur antara waktu eksekusi dan kompleksitas program. Pada skenario Sudoku tingkat mudah (*Easy* dan *Medium*), *backtracking* unggul berkat ketiadaan overhead memori pada operasi primitifnya. Namun, keunggulan tersebut hilang pada Sudoku tingkat tinggi, terbukti dari algoritma CP yang menyelesaikan papan rata-rata dalam 18,33 ms dibandingkan DFS yang membengkak hingga 58 ms. Hal ini memvalidasi bahwa efisiensi polinomial dari algoritma deduktif mutlak mengungguli kompleksitas eksponensial O(9^n) dari *Backtracking* pada skenario terburuk. Meskipun secara teori jauh lebih efisien, algoritma CP memiliki batasan ketika berhadapan dengan konfigurasi Sudoku *Evil* yang sengaja dirancang untuk memicu kebuntuan. Oleh karena itu, penyelesaian komputasional yang paling optimal adalah sebuah sistem *hybrid*. Pendekatan ini memposisikan CP sebagai algoritma utama yang mengurangi ruang pencarian secara drastis, sementara algoritma *backtracking* diimplementasikan murni sebagai *fallback mechanism* apabila algoritma CP telah membentur batasnya.

V. Lampiran

Kode yang digunakan ada pada link dibawah.

https://docs.google.com/document/d/1twJe7BUskyPp1uOezYddEApd7fGPClReFl6vTK03C1o/edit?usp=sharing

VI. Ucapan Terima Kasih

Terima kasih kepada Tuhan Yang Maha Esa karena berkat rahmat dan kasih karunia-Nya penulis dapat menyelesaikan makalah kali ini dengan baik dan tanpa kendala yang terlalu berat. Tak lupa juga penulis ingin mengucapkan terima kasih kepada keluarga yang telah memberikan dukungan sehingga penulis dapat menyelesaikan makalah ini. Terima kasih juga penulis sampaikan kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah IF1220 untuk kelas 02 karena telah memberikan pengetahuan yang dapat digunakan penulis untuk menulis makalah ini. Penulis berharap bahwa makalah ini nanti kedepannya dapat digunakan sebagai referensi baik bagi para pelajar yang penasaran dengan ilmunya atau bahkan sebagai referensi penulis kedepannya.

Referensi

[1] Munir, Rinaldi. 2025. Himpunan (bagian 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/02-Himpunan(2026)-1.pdf (Diakses 18 Juni 2026)

[2] Munir, Rinaldi. 2025. Himpunan (bagian 2). https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/03-Himpunan(2026)-2.pdf (Diakses 18 Juni 2026)

[3] Munir, Rinaldi. 2025. Himpunan (bagian 2). https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/04-Himpunan(2026)-3.pdf (Diakses 18 Juni 2026)

[4] Sudoku Coach. 2020. Learn sudoku. https://sudoku.coach/en/learn (Diakses 18 Juni 2026)

[5] Hoss James. 2026. *The World's Hardest Sudoku Puzzles: Can You Solve Them?* https://sudokupulse.com/articles/worlds-hardest-sudoku-puzzles/ (Diakses 18 Juni 2026)

[6] Tambunan, Clarissa Nethania. 2024. *Penerapan Permutasi Sebagai Bagian dari Kombinatorika pada Game Sudoku* https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/Makalah/Makalah-IF1220-Matdis-2024%20(141).pdf (Diakses 18 Juni 2026)

[7] Amanullah, Muhammad Zaki. 2022. *Penerapan Graph Theory dan Graph Coloring dalam Memecahkan Teka-teki Sudoku* . https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2022-2023/Makalah2022/Makalah-Matdis-2022%20(127).pdf (diakses 19 Juni 2026)

[8] Williams, Brian. 2005. *Sudoku dasar-dasar dancontoh yang telah dikerjakan.* https://urbanrim-org-uk.translate.goog/sudoku.htm?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=imgs (diakses 19 Juni 2026)

[9] Sudokutable. 2026. *Cara bermain sudoku. Aturan dan metode penyelesaian* https://sudokutable.com/howtoplay/?lang=id (diakses 19 Juni 2026)

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, 19 Juni 2025

Mochammad Adhitya Nur Rohman 13525038