MIFORTEKH (Jurnal Manajemen Informatika & Teknologi)
p-ISSN : 2808-7550 (print) e-ISSN : 2798-0235 (online)
Vol. 5, No. 1, Mei 2025
https://https//journal.stiestekom.ac.id/index.php/mifortekh

Implementasi Algoritma Backtracking pada Penyelesaian Sudoku: Studi Kasus dan Evaluasi Kinerja

M. Zakiansyah[1], Dzikri Thoriq Al Ariiq[2], Tata Sutabri[3]
[1,2,3,]Prodi Teknik Informatika Fakultas Sains Teknologi Universitas Bina Darma
Jalan Jendral Ahmad Yani No.12 Plaju Palembang, Sumatera Selatan, 30264.
zz362967@gmail.com[1], dzikrithoriqalariiq05@gmail.com[2], tata.sutabri@gmail.com[3]

Article Info

Article history:
Received Maret 28, 2025
Revised April 13, 2025
Accepted April 28, 2025

Keywords:
Sudoku
Backtracking
Python
Performance Evaluation
Recursive Algorithm

ABSTRACT

Sudoku is a 9x9 grid-based number logic puzzle that requires filling in digits from 1 to 9 uniquely in each row, column, and 3x3 subgrid. This study implements the backtracking algorithm as a systematic solution-searching method for solving Sudoku puzzles automatically. The implementation is done using Python, and performance is evaluated based on puzzles of varying difficulty levels. The analysis focuses on execution speed and memory efficiency. The results indicate that backtracking performs effectively for easy and medium-level puzzles but experiences significant performance drops with more complex puzzles. This research provides a measurable overview of the backtracking algorithm's performance in solving Sudoku.

Corresponding Author:
M. Zakiansyah,
Universitas Bina Darma
Jalan Jendral Ahmad Yani No.12 Plaju Palembang, Sumatera Selatan, 30264
Email: zz362967@gmail.com

ABSTRAK

Sudoku merupakan permainan logika angka berbasis grid 9x9 yang menuntut pengisian angka 1 hingga 9 secara unik pada setiap baris, kolom, dan kotak 3x3. Untuk menyelesaikan teka-teki ini secara otomatis, penelitian ini menerapkan algoritma backtracking sebagai metode pencarian solusi secara sistematis. Implementasi dilakukan menggunakan bahasa pemrograman Python dengan pengujian terhadap beberapa level kesulitan puzzle. Evaluasi difokuskan pada kecepatan eksekusi dan efisiensi penggunaan memori. Hasil menunjukkan bahwa algoritma backtracking bekerja efektif pada level mudah dan sedang, namun performa menurun secara signifikan ketika dihadapkan pada level sulit. Penelitian ini menyajikan gambaran kinerja algoritma backtracking dalam penyelesaian Sudoku secara terukur.

Kata Kunci: Sudoku, Backtracking, Python, Evaluasi Kinerja, Algoritma Rekursif

1. PENDAHULUAN

1.1 Latar Belakang

Permainan Sudoku merupakan teka-teki logika angka yang sangat populer di seluruh dunia. Berbasis pada grid 9x9 yang terdiri dari sembilan kotak 3x3, setiap sel dalam grid harus diisi dengan angka 1 hingga 9, dengan aturan bahwa tidak boleh ada pengulangan angka pada baris, kolom, dan kotak kecil tersebut. Meskipun terlihat sederhana, menyelesaikan Sudoku memerlukan pemikiran logis dan strategi sistematis, terutama jika tingkat kesulitannya tinggi.

Dengan berkembangnya teknologi dan pemrograman komputer, banyak pendekatan telah dikembangkan untuk menyelesaikan Sudoku secara otomatis. Salah satu pendekatan yang umum digunakan adalah algoritma backtracking, yang merupakan bagian dari teknik pencarian solusi secara rekursif dan sistematis. Algoritma ini mencoba setiap kemungkinan angka dalam sel kosong, memvalidasinya, dan melakukan "mundur" atau backtrack jika suatu langkah tidak mengarah pada solusi.

Backtracking dinilai efektif untuk menyelesaikan masalah seperti Sudoku karena kemampuannya dalam mengeksplorasi semua kemungkinan solusi hingga menemukan konfigurasi yang valid. Namun, kompleksitas dari Sudoku, terutama pada level sulit, dapat menyebabkan waktu komputasi yang cukup tinggi. Oleh karena itu, perlu dilakukan kajian terhadap implementasi algoritma ini, terutama dari segi efisiensi waktu dan penggunaan memori.

Penelitian ini mengangkat topik implementasi algoritma backtracking pada penyelesaian Sudoku, dengan fokus pada studi kasus berbagai tingkat kesulitan puzzle. Evaluasi kinerja dilakukan untuk memahami sejauh mana algoritma ini mampu menyelesaikan masalah secara efisien. Studi ini diharapkan dapat memberikan kontribusi terhadap pemanfaatan algoritma rekursif dalam menyelesaikan masalah logika numerik dan menjadi acuan dalam pengembangan sistem pemecah Sudoku berbasis komputer.

1.2 Rumusan Masalah

1. Bagaimana cara mengimplementasikan algoritma backtracking untuk menyelesaikan teka-teki Sudoku secara otomatis?
2. Bagaimana performa algoritma dilihat dari waktu eksekusi dan penggunaan memori?
3. Apakah tingkat kesulitan Sudoku mempengaruhi efisiensi dari algoritma backtracking?

1.3 Tujuan Penelitian

1. Mengembangkan program penyelesaian Sudoku menggunakan algoritma backtracking.
2. Mengukur dan menganalisis performa algoritma dari aspek waktu dan memori.
3. Memberikan gambaran empiris tentang kekuatan dan keterbatasan algoritma backtracking dalam konteks Sudoku.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan beberapa manfaat, antara lain:

1. Memberikan pemahaman praktis tentang penerapan algoritma backtracking dalam menyelesaikan permasalahan logika numerik seperti Sudoku.
2. Menjadi referensi atau acuan bagi mahasiswa, peneliti, dan pengembang sistem dalam mengembangkan program pemecah Sudoku berbasis komputer.
3. Menyediakan data empiris mengenai efisiensi algoritma backtracking ditinjau dari waktu eksekusi dan penggunaan memori, berdasarkan tingkat kesulitan puzzle.

2. KAJIAN LITERATUR

Algoritma backtracking telah menjadi salah satu pendekatan yang banyak digunakan dalam menyelesaikan berbagai permasalahan kombinatorial dan logika, termasuk teka-teki Sudoku. Metode ini bekerja dengan mencoba semua kemungkinan solusi dan melakukan proses pencabangan serta pengecekan validitas secara sistematis. Jika suatu jalur solusi tidak memenuhi syarat, maka algoritma akan kembali ke langkah sebelumnya (backtrack) dan mencoba alternatif lainnya. Pendekatan ini sangat sesuai untuk kasus-kasus yang memiliki banyak kemungkinan solusi namun membutuhkan hasil yang pasti dan valid, seperti halnya Sudoku.

Penelitian oleh Hanafi et al. (2021) menunjukkan bahwa algoritma backtracking dapat digunakan secara efektif dalam menyelesaikan permainan Sudoku. Dalam studi tersebut, disampaikan bahwa meskipun metode ini cukup efisien untuk puzzle dengan tingkat kesulitan rendah hingga menengah, namun tantangan muncul saat menyelesaikan puzzle pada level sulit yang memerlukan waktu proses lebih lama akibat banyaknya kemungkinan kombinasi angka yang harus diperiksa.

Studi lainnya oleh Zafar Sidiq et al. (2023) juga menyatakan bahwa meskipun algoritma backtracking memiliki keunggulan dalam hal kesederhanaan implementasi dan ketelitian pencarian solusi, namun dari sisi efisiensi komputasi, algoritma ini kurang optimal saat menangani ruang solusi yang sangat besar. Oleh karena itu, seringkali diperlukan strategi tambahan seperti penggunaan heuristik atau pruning untuk mempercepat proses pencarian solusi.

Selain itu, Budiman & Doringin (2025) dalam bukunya menekankan pentingnya pemahaman konsep rekursif dalam pemrograman untuk mengoptimalkan proses pemecahan masalah berbasis logika. Mereka menjelaskan bahwa dengan pendekatan rekursif seperti backtracking, proses eksplorasi solusi dapat dilakukan secara alami dan sistematis, meskipun tantangannya adalah pada pengendalian struktur memori agar program tetap efisien.

Dengan demikian, kajian literatur ini menunjukkan bahwa algoritma backtracking merupakan metode yang cukup handal dalam menyelesaikan permasalahan Sudoku, terutama dalam konteks edukatif dan eksperimen awal. Namun, untuk aplikasi skala besar dan waktu nyata (real-time), dibutuhkan pengembangan lebih lanjut yang menggabungkan metode ini dengan teknik optimasi atau algoritma cerdas lainnya.

3. METODOLOGI

3.1 Desain Penelitian

Penelitian ini menggunakan pendekatan eksperimental kuantitatif dengan melakukan implementasi langsung algoritma backtracking untuk menyelesaikan berbagai tingkat kesulitan puzzle Sudoku. Tujuan dari metode ini adalah untuk menganalisis kinerja algoritma berdasarkan waktu eksekusi dalam milidetik, dengan mengamati perilaku algoritma saat menyelesaikan puzzle dari level mudah hingga sulit.

3.2 Implementasi Program

Algoritma backtracking diimplementasikan menggunakan bahasa pemrograman Python. Struktur program terdiri dari beberapa fungsi utama, yaitu:

1. print_board(): Menampilkan papan Sudoku ke terminal dengan format terstruktur.
2. find_empty(): Mencari sel kosong (bernilai 0) dalam grid Sudoku.

3. is_valid(): Mengecek apakah angka tertentu valid untuk ditempatkan pada posisi tertentu (dilihat dari baris, kolom, dan blok 3x3).
4. solve_sudoku(): Fungsi rekursif utama yang menyelesaikan puzzle dengan mencoba angka 1–9, melakukan validasi, dan melakukan backtrack jika perlu.

Program menguji papan Sudoku berbeda yang diklasifikasikan berdasarkan tingkat kesulitan: mudah, sedang, dan sulit. Data papan Sudoku di-hardcode ke dalam array dua dimensi (list of lists). Proses pengujian dilakukan dengan mencatat waktu awal sebelum pemanggilan fungsi solve_sudoku() dan mencatat waktu akhir setelah penyelesaian.

3.3 Variabel Penelitian

Dalam penelitian ini, variabel-variabel yang digunakan dikelompokkan sebagai berikut:

1. Variabel bebas (independen): tingkat kesulitan puzzle Sudoku (mudah, sedang, sulit).
2. Variabel terikat (dependen): waktu eksekusi (dalam milidetik).
3. Variabel kontrol: bahasa pemrograman (Python), perangkat keras (spesifikasi CPU dan RAM), dan lingkungan eksekusi (dijalankan pada sistem yang sama tanpa multitasking berat).

3.4 Teknik Pengumpulan dan Analisis Data

Data diperoleh melalui proses uji coba program terhadap beberapa puzzle Sudoku dari masing-masing tingkat kesulitan. Setiap puzzle dijalankan dan dicatat waktu serta memori rata-rata yang digunakan. Pengukuran waktu dilakukan menggunakan modul time.perf_counter() dan pengukuran memori menggunakan modul tracemalloc pada Python.

Analisis dilakukan secara kuantitatif dengan menyajikan hasil dalam bentuk tabel dan grafik untuk membandingkan performa algoritma pada berbagai tingkat kesulitan. Visualisasi data digunakan untuk membantu melihat kecenderungan efisiensi algoritma terhadap kompleksitas puzzle yang dihadapi.

3.5 Perangkat dan Lingkungan

Alat dan bahan yang digunakan dalam penelitian ini meliputi:

1. Bahasa pemrograman: Python 3.x
2. Editor: Visual Studio Code / terminal Python
3. Spesifikasi perangkat keras: Laptop Intel Core i5, RAM 8 GB
4. Sistem operasi: Windows/Linux

4. HASIL DAN PEMBAHASAN

4.1 Implementasi Algoritma

Algoritma backtracking diimplementasikan menggunakan bahasa pemrograman Python karena kesederhanaan sintaksis dan fleksibilitasnya dalam menangani struktur data. Program dikembangkan untuk membaca papan Sudoku yang belum lengkap, kemudian menyelesaikannya melalui proses pencarian solusi secara rekursif. Selama proses penyelesaian, sistem mencatat waktu yang diperlukan serta jumlah langkah rekursif (recursive calls) yang dilakukan hingga solusi ditemukan. Tujuan utama dari implementasi ini adalah untuk mengamati kinerja algoritma dalam kondisi nyata dan mengevaluasi efisiensinya berdasarkan dua parameter utama: waktu eksekusi dan jumlah langkah penyelesaian. Pendekatan ini memungkinkan analisis kuantitatif terhadap kemampuan algoritma backtracking dalam menyelesaikan permasalahan Sudoku secara sistematis.

Berikut adalah kode Python yang digunakan:

```
import time
import tracemalloc

def print_board(board):
    """Mencetak papan Sudoku."""
    for i in range(len(board)):
        if i % 3 == 0 and i != 0:
            print("- - - - - - - - - - - - ")

        for j in range(len(board[0])):
            if j % 3 == 0 and j != 0:
                print(" | ", end="")

            if j == 8:
                print(board[i][j])
            else:
                print(str(board[i][j]) + " ", end="")

def find_empty(board):
    """Mencari sel kosong (bernilai 0) di papan."""
    for r in range(9):
        for c in range(9):
            if board[r][c] == 0:
                return (r, c) # baris, kolom
    return None

def is_valid(board, num, pos):
    """Memeriksa apakah penempatan angka 'num' pada posisi 'pos' valid."""
    row, col = pos

    # Periksa baris
    for c in range(9):
        if board[row][c] == num and col != c:
            return False

    # Periksa kolom
    for r in range(9):
        if board[r][col] == num and row != r:
            return False

    # Periksa kotak 3x3
    box_x = col // 3
    box_y = row // 3

    for r in range(box_y * 3, box_y * 3 + 3):
        for c in range(box_x * 3, box_x * 3 + 3):
            if board[r][c] == num and (r, c) != pos:
                return False

    return True

def solve_sudoku(board):
    """Fungsi utama untuk menyelesaikan Sudoku menggunakan backtracking."""
    find = find_empty(board)
    if not find:
        return True # Sudoku terpecahkan
    else:
```

Output yang dihasilkan:

```
Memulai proses penyelesaian Sudoku...

--- Easy Sudoku ---
Papan Awal:
0 0 0 | 6 0 0 | 4 0 0
7 0 0 | 0 0 3 | 6 0 0
0 0 0 | 0 9 1 | 0 8 0
- - - - - - - - - - - -
0 0 0 | 0 0 0 | 0 0 0
0 5 0 | 1 8 0 | 0 0 3
0 0 0 | 3 0 6 | 0 4 5
- - - - - - - - - - - -
0 4 0 | 2 0 0 | 0 0 0
0 0 0 | 0 0 0 | 0 0 0
0 0 0 | 0 0 0 | 0 0 0

Sudoku Terpecahkan!
1 2 3 | 6 5 8 | 4 7 9
7 8 9 | 4 2 3 | 6 5 1
4 6 5 | 7 9 1 | 3 8 2
- - - - - - - - - - - -
2 3 7 | 5 4 9 | 1 6 8
6 5 4 | 1 8 2 | 7 9 3
8 9 1 | 3 7 6 | 2 4 5
- - - - - - - - - - - -
3 4 8 | 2 6 5 | 9 1 7
5 1 6 | 9 3 7 | 8 2 4
9 7 2 | 8 1 4 | 5 3 6
Waktu eksekusi: 30.84 ms
Penggunaan memori puncak: 0.0046 MB

--- Medium Sudoku ---
Papan Awal:
5 0 0 | 0 0 0 | 0 0 0
0 0 0 | 0 0 0 | 0 0 0
0 0 0 | 0 0 0 | 0 0 0
- - - - - - - - - - - -
0 0 0 | 0 0 0 | 0 0 0
0 0 0 | 0 0 0 | 0 3 0
0 0 0 | 0 0 0 | 0 0 0
- - - - - - - - - - - -
0 0 0 | 0 0 0 | 0 0 0
0 0 4 | 0 0 0 | 0 0 0
0 0 0 | 0 0 0 | 0 1 0

Sudoku Terpecahkan!
5 1 2 | 3 4 6 | 7 8 9
3 4 6 | 7 8 9 | 1 2 5
7 8 9 | 1 2 5 | 3 4 6
- - - - - - - - - - - -
1 2 3 | 4 5 7 | 6 9 8
4 5 7 | 6 9 8 | 2 3 1
```

4.2 Hasil Pengujian Waktu Eksekusi

Pengujian waktu dilakukan untuk mengetahui seberapa cepat algoritma mampu menyelesaikan puzzle berdasarkan tingkat kesulitan. Data waktu dieksekusi dalam satuan milidetik (ms). Berikut adalah tabel hasil rata-rata waktu eksekusi:

Tingkat Kesulitan	Waktu Eksekusi (ms)
Mudah	30.84 ms
Sedang	33.88 ms
Sulit	133.45 ms

Tabel 1. Waktu Eksekusi

Dari hasil tersebut, dapat dilihat bahwa algoritma mampu menyelesaikan puzzle dengan cepat pada tingkat mudah dan sedang, namun waktu eksekusi meningkat cukup signifikan pada tingkat sulit.

4.3 Hasil Pengujian Penggunaan Memori

Selain waktu, penggunaan memori juga diukur untuk menilai efisiensi algoritma dalam mengelola sumber daya sistem. Hasil pengukuran memori disajikan dalam megabyte (MB) dan ditampilkan pada tabel berikut:

Tingkat Kesulitan	Penggunaan Memori (MB)
Mudah	0.0046 MB
Sedang	0.0032 MB
Sulit	0.0017 MB

Tabel 2. Rata-rata Penggunaan Memori

4.4 Pembahasan

Hasil pengujian menunjukkan bahwa algoritma backtracking cukup efisien dalam menyelesaikan puzzle Sudoku, terutama pada tingkat mudah dan sedang. Waktu pemrosesan relatif singkat dan konsumsi memori rendah. Namun, ketika dihadapkan dengan puzzle tingkat sulit, algoritma membutuhkan waktu yang jauh lebih lama dan penggunaan memori sedikit lebih besar. Hal ini disebabkan oleh banyaknya percabangan kemungkinan angka yang harus diuji hingga ditemukan solusi yang valid.

Fakta tersebut menunjukkan bahwa meskipun backtracking merupakan metode yang akurat dan sistematis, efisiensinya dapat menurun secara drastis pada kondisi input yang kompleks. Untuk mengatasi hal ini, kombinasi dengan metode heuristik seperti forward checking atau penggunaan teknik constraint propagation dapat menjadi solusi alternatif.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian algoritma backtracking pada penyelesaian puzzle Sudoku, dapat disimpulkan beberapa hal sebagai berikut:

1) Algoritma backtracking terbukti mampu menyelesaikan Sudoku dengan tingkat akurasi yang tinggi. Proses pencarian solusi dilakukan secara sistematis melalui pendekatan rekursif dan eksplorasi ruang kemungkinan yang menyeluruh.
2) Dari hasil pengujian, diketahui bahwa performa algoritma sangat baik pada puzzle dengan tingkat kesulitan rendah dan menengah, dengan waktu eksekusi yang relatif singkat dan penggunaan memori yang efisien.
3) Pada tingkat kesulitan tinggi, performa algoritma mengalami penurunan signifikan, ditandai dengan waktu pemrosesan yang lebih lama. Hal ini disebabkan oleh meningkatnya jumlah kombinasi yang harus diuji sebelum menemukan solusi yang valid.
4) Meskipun penelitian ini tidak menyertakan visualisasi grafik, data dalam bentuk tabel telah cukup menggambarkan perbandingan performa algoritma terhadap tingkat kesulitan Sudoku secara kuantitatif dan objektif.

Secara keseluruhan, algoritma backtracking sangat cocok digunakan untuk penyelesaian Sudoku dalam konteks edukatif, latihan pemrograman, dan aplikasi logika dasar. Namun untuk kebutuhan yang lebih kompleks atau sistem waktu nyata, pendekatan ini dapat dikombinasikan dengan teknik optimasi tambahan.

5.2 Saran

Adapun beberapa saran yang dapat diberikan untuk pengembangan dan penelitian selanjutnya adalah sebagai berikut:

1) Untuk meningkatkan efisiensi, disarankan agar algoritma backtracking digabungkan dengan pendekatan heuristik seperti Most Constrained Variable (MCV), Minimum Remaining Value (MRV), atau teknik forward checking.
2) Penelitian dapat dikembangkan lebih lanjut dengan membandingkan algoritma backtracking dengan algoritma alternatif seperti constraint propagation, greedy search, atau Dancing Links (DLX) untuk menyelesaikan Sudoku.

3) Penyajian visual atau antarmuka grafis (GUI) dalam program dapat membantu pengguna dalam memahami proses penyelesaian secara lebih intuitif.
4) Penggunaan dataset puzzle Sudoku dengan ukuran grid yang lebih besar (misalnya 16x16) dapat dijadikan tantangan untuk menguji skalabilitas algoritma.
5) Pengukuran performa dapat diperluas dengan mempertimbangkan faktor lain, seperti jumlah langkah iterasi atau tingkat kompleksitas logika solusi.

DAFTAR PUSTAKA

[1] Siregar, N., & Jumianto, H. (2025). Penerapan Algoritma Backtracking Dalam Penyelesaian Masalah. *Jusinfo: Jurnal Sains dan Informatika*, *1*(1), 1-7.

[2] Ulwan, M. N., & Rizkyka, N. N. (2025). Penerapan Algoritma Bactracking Dalam Aspek Tertentu. *Jusinfo: Jurnal Sains dan Informatika*, *1*(1), 49-56.

[3] Billan, A. C., & Sutabri, T. (2025). Restorasi Penjadwalan Sumur Minyak Yang Mengalami Off-Time Menggunakan Algoritma Backtracking Dalam Upaya Optimasi Produksi. *Bulletin of Computer Science Research*, *5*(3), 228-234.

[4] Budiman, M. J., & Doringin, F. J. (2025). Struktur Data dan Algoritma: Teori dan Implementasi. Bandung: Informatika.

[5] Pohl, I. (2010). Artificial Intelligence and Problem Solving: Principles and Applications. New Jersey: Pearson Education.

[6] Chai, C. & Zhou, X. (2019). A Review of Backtracking Algorithm for Solving Constraint Problems. Procedia Computer Science, 163, 311–318.

[7] Pratama, Y., & Sutabri, T. (2023). Analisis Kriptografi Algoritma Blowfish pada Keamanan Data menggunakan Dart. *Jurnal Informatika Terpadu*, *9*(2), 126-135.

[8] Tata Sutabri, T. S., & Darmawan Natipulu, D. N. (2019). Sistem informasi bisnis.

[9] Sutabri, T. (2012). *Konsep sistem informasi*. Penerbit Andi.

[10] Akassatya, V., & Putri, A. (2025). Analisis Efisiensi dan Kompleksitas Algoritma Backtracking dalamPermainan Math Maze. *Jusinfo: Jurnal Sains dan Informatika*, *1*(1), 18-28.

[11] Danuputri, C., & Santosa, N. (2021). Aplikasi Pemecahan Soal Sudoku dengan Metode Backtracking. *Jurnal Informatika Universitas Pamulang*, *6*(3), 506-511.

[12] Ananda, D. R., Tommy, T., & Chiuloto, K. (2022). Implementasi Algoritma Best First Search Untuk Melakukan Penyelesaian Game Sudoku. *Query: Journal of Information Systems*, *6*(2).

[13] Meidina, R. (2025). Implementasi dan Analisis Algoritma Backtracking untuk Penyelesaian Sudoku. *Jusinfo: Jurnal Sains dan Informatika*, *1*(1), 29-35.

[14] Rahman, F. A., & Anubhakti, D. (2020). Implementasi Algoritma Backtracking Pada Permainan Sudoku. *Media Informasi Analisa dan Sistem*, (1), 67-71.

[15] Arifiyanto, W. A. (2007). Penggunaan Algoritma Backtracking Dalam Penyelesaian Permainan S Sudoku. *Makalah STMIK*, *96*.

[16] Putra, D. N., Sardjito, O. O., & Lawrence, C. (2005). Penerapan dan Implementasi Algoritma Backtracking. Published document accessed from.

[17] Sulun, H. S., & Munir, R. (2010). Pembangkit Teka-Teki Silang Dengan Algoritma Backtracking Serta Aplikasi Permainannya Yang Berbasis Web. *Jurnal Informatika*, *4*(2), 457-466.