JUSINFO
(Jurnal Sains dan Informatika)
Journal homepage: https://ejournal.pelitabina.com/index.php/Jusinfo
Pp: 29-35
Published By: Pelita Bina

Volume. 1 Nomor 1
Januari 2025

Implementasi dan Analisis Algoritma Backtracking untuk Penyelesaian Sudoku

Risha Meidina[1], Miftahul Firdaus[2]

[1] Program Studi teknik informatika, STMIK Pelita nusantara, Lubuk pakam (email: rishameidina8@gmail.com)
[2] Program Studi teknik informatika, STMIK Pelita nusantara, Lubuk pakam (email: ff7820192@gmail.com)

[Diserahkan: 23 Desember 2024, Direvisi: 30 Desember 2024, Diterima: 10 Januari 2025]
Corresponding Author: Risha Meidina (email: rishameidina8@gmail.com)

Kata Kunci: Backtracking, Sudoku, Python, Visual Basic, Kompleksitas Np-Complete, Rekursi, Pruning Jalur**.**

Abstrak : Algoritma *backtracking* adalah metode pencarian solusi yang efisien untuk masalah kombinatorial, termasuk permainan Sudoku. Penelitian ini membahas implementasi algoritma *backtracking* menggunakan *Python* dengan pustaka *PyGame* dan *Visual Basic* 6.0. Hasil penelitian menunjukkan bahwa algoritma ini memberikan solusi optimal dengan akurasi 100% dan waktu penyelesaian lebih cepat dibandingkan metode *brute force*. Implementasi ini mengilustrasikan kemampuan *backtracking* dalam memangkas solusi yang tidak valid melalui pendekatan rekursif dan pruning jalur.

PENDAHULUAN

Permainan Sudoku adalah teka-teki angka yang berasal dari Jepang dan pertama kali dikenal sebagai "*Number Place*" di Amerika Serikat pada tahun 1979. Setiap pemain harus mengisi grid 9×9 dengan angka 1 hingga 9, memastikan bahwa setiap angka muncul tepat satu kali di setiap baris, kolom, dan subgrid 3×3. Setiap angka harus unik dalam satu baris, kolom, dan subgrid 3×3. Sudoku sering digunakan sebagai studi kasus untuk implementasi algoritma *kombinatorial*, karena termasuk dalam kategori *NP-complete***.**

Algoritma *backtracking* adalah metode penyelesaian yang paling cocok untuk Sudoku. *Backtracking* bekerja dengan mencoba satu per satu kemungkinan angka dalam grid kosong dan melakukan pengecekan rekursif untuk validasi angka yang dimasukkan. Jika ditemukan solusi tidak valid, algoritma akan kembali ke langkah sebelumnya dan mencoba solusi lain. Dengan mekanisme *pruning* jalur (pemangkasan jalur yang tidak valid), *backtracking* menghemat waktu komputasi secara signifikan dibandingkan *brute force*.

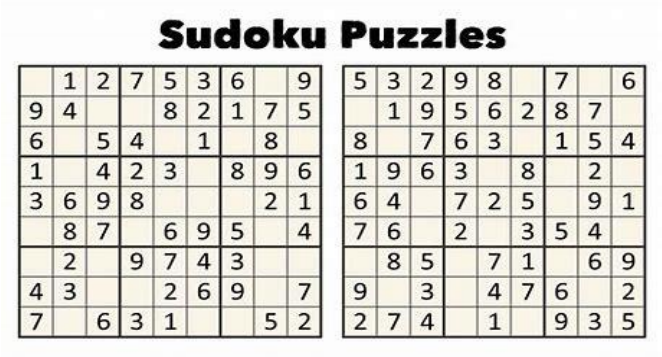
Penelitian ini bertujuan:

1. Mengimplementasikan algoritma *backtracking* dalam penyelesaian Sudoku.
2. Mengevaluasi performa algoritma dalam dua platform: *Python* dan *Visual Basic*.
3. Menjelaskan kompleksitas waktu dan ruang dalam implementasi algoritma *backtracking*.
4. Menunjukkan efektivitas *pruning* dalam menghindari eksplorasi jalur solusi tidak valid.

Peraturan Sudoku:

1. Setiap baris, kolom, dan subgrid 3×3 harus berisi angka unik dari 1 hingga 9.
2. Sel kosong diisi dengan angka valid berdasarkan aturan Sudoku.

Angka-angka ini sebenarnya tidak memiliki hubungan aritmetis satu sama lain. Anda boleh menggantinya dengan 9 huruf, lambang, atau warna yang berbeda.



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

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

Gambar 1. Puzzle Sudoku

TINJAUAN LITERATUR

Adapun penelitian terdahulu yang menjadi acuan penelitian ini adalah penelitian (Rifqo Muhammad Husni & Apridiansyah Yovi, 2017) dengan judul penelitian "Implementasi Algoritma *Backtracking* Dalam Sistem Informasi Perpustakaan Untuk Pencarian Judul Buku". Algoritma *Backtracking* mampu menemukan semua solusi pada pencarian data judul buku, kelebihan dari algorima *Backtracking* yaitu mudah merunut balik suatu langkah sehingga apabila menemui langkah terakhir maka dapat kembali pada posisi sebelumnya yang mempunyai langkah solusi sehingga dapat menemukan solusi lebih cepat.

Penelitian (Kiswanto, 2020) dengan judul penelitian "Spesifikasi Komputer Rakitan Berdasarkan Kebutuhan dan Anggaran Menggunakan Algoritma *Backtracking"*. Pada penelitian ini Algoritma *Backtracking* adalah algoritma yang berbasis pada *Depth First Search (DFS)* untuk mencari solusi persoalan secara lebih efisien. Algoritma *Backtracking* tidak memeriksa semua kemungkinan solusi yang ada, hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan, akibatnya waktu pencarian dapat dihemat.

Penelitian (Rahmawati yunia, 2020) dengan judul penelitian "Penerapan Algoritma *Backtracking* Pada *N-Queen Problem* Permainan Catur". Algoritma *Backtracking* memiliki property umum, salah satunya *Bounding Function* (Fungsi Pembatas). *Bounding Function* adalah kondisi yang bernilai benar. Kondisi dapat dikatakan benar jika ratu *(Queen)* tidak pada baris, kolom dan diagonal yang sama. Fungsi pembatas ini dinyatakan sebagai B(x1,x2,x3,...xn) akan bernilai true jika (x1,x2,x3,…..xn) mengarah pada solusi dan tidak melanggar batasan.

Penelitian (Krisdiawan et al., 2022) dengan judul penelitian "Penerapan Algoritma *Recursive Backtracking* sebagai *Maze Generator* Pada Game Labirin Aksara Sunda". Tujuan dari penelitian ini adalah membuat game labirin aksara sunda yang menarik dan dapat meningkatkan pemahaman terhadap aksara sunda dengan mengimplementasikan algoritma *recursive backtracking* sebagai *maze generator* labirin. Manfaat yang ingin dicapai dari penelitian yaitu adanya media pembelajaran alternatif yang menarik untuk siswa-siswi SDN 1 Jalaksana dalam mempelajari aksara sunda berupa permainan game labirin yang menyenangkan dengan pengimplementasian algoritma *recursive backtracking* sebagai *maze generator* labirin.

Penelitian (Imiah et al., 2008) dengan judul penelitian "Analisis Penyelesaian *Puzzle* Sudoku Dengan Menerapkan Algoritma *Backtracking* Memanfaatkan Bahasa Pemrograman *Visual Basic* 6.0". Algoritma *backtracking* adalah suatu algoritma yang merupakan perbaikan dari algoritma *brute force*, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada (6). *Backtracking* merupakan bentuk tipikal dari algoritma rekursif dan berbasis pada DFS dalam mencari solusi yang tepat. Selain itu, algoritma ini juga merupakan metode yang mencoba-coba beberapa keputusan sampai kita menemukan salah satu yang "berjalan". Kita tidak perlu memeriksa semua kemungkinan solusi yang ada, tetapi cukup yang mengarah kepada solusi saja. Dengan memangkas *(pruning)* simpul-simpul yang tidak mengarah ke solusi, sehingga waktu pencarian dapat dihemat. Algoritma ini banyak diterapkan untuk program games dan permasalahan pada bidang kecerdasan buatan.

Penelitian (Danuputri & Santosa, 2021) dengan judul penelitian "Aplikasi Pemecahan Soal Sudoku dengan Metode *Backtracking"*. Algoritma *backtracking* (runut balik) adalah sebuah urutan logis secara sistematis yang

digunakan untuk mencari solusi dari sebuah permasalahan di mana terdapat beberapa kemungkinan solusi yang ada (Rifqo & Apridiansyah, 2017). Algoritma ini bekerja secara rekursif dalam mencari solusi dari sebuah permasalahan di mana terdapat beberapa kemungkinan solusi. Algoritma ini berbasis pada algoritma *Depth-First Search (DFS)* untuk mencari solusi persoalan yang lebih efektif.

Penelitian (Angel et al., 2023) dengan judul penelitian "Penerapan Algoritma Backtracking Berbasis BFS dengan Pendekatan Heuristik dalam Permainan Hangman". Hangman adalah permainan dimana pemain akan menebak sebuah kata rahasia. Pemain hanya mengetahuijumlah huruf pada kata berdasarkan jumlah ruang kosong yang tersedia untuk(ARIEF HIJAYANTO, 2012). Untuk itu, penulis tertarik untuk membahas dan memberi sebuah inovasi pada permainanhangman. Asal mulanya permainanhangman dimulai pada abad ke-17 di Eropa dan menjadi permainan yang popular di seluruh dunia.

Penelitian (Elvina & Hakim, 2018) dengan judul penelitian "Modifikasi Algoritma Steepest-Ascent Hill Climbing Dan Backtracking Untuk Pencarian Lintasan Kritis Proyek". Dalam penelitian ini, algoritma pencarian yang digunakan untuk mencari slack time adalah Steepest-ascent Hill Climbing. Algoritma pencarian Steepest-ascent Hill Climbing pada dasarnya hampir sama dengan Algoritma pencarian Simple Hill Climbing, yang membedakannya adalah gerakan pencarian yang tidak dimulai dari posisi paling kiri namun gerakan selajutnya dicari berdasarkan nilai heuristik terbaik.

Penelitian (Munarto & Permata, 2017) dengan judul penelitian "Perancangan Sistem Penjadualan Kuliah Di Jurusan Teknik Elektro Ft.Untirta Menggunakan Teknik Pewarnaan Graph Algoritma Backtracking Welch-Powell. Dari hasil pewarnan graph dengan algoritma backtracking titik-titik dikelompokkan berdasarkan warnanya dan akan dibuat jadwal kuliah dengan ketentuan titik-titik yang warnanya sama dipetakan ke waktu kuliah yang sama dan titik-titik yang warnanya sama dipetakan ke ruang kuliah yang berbeda.

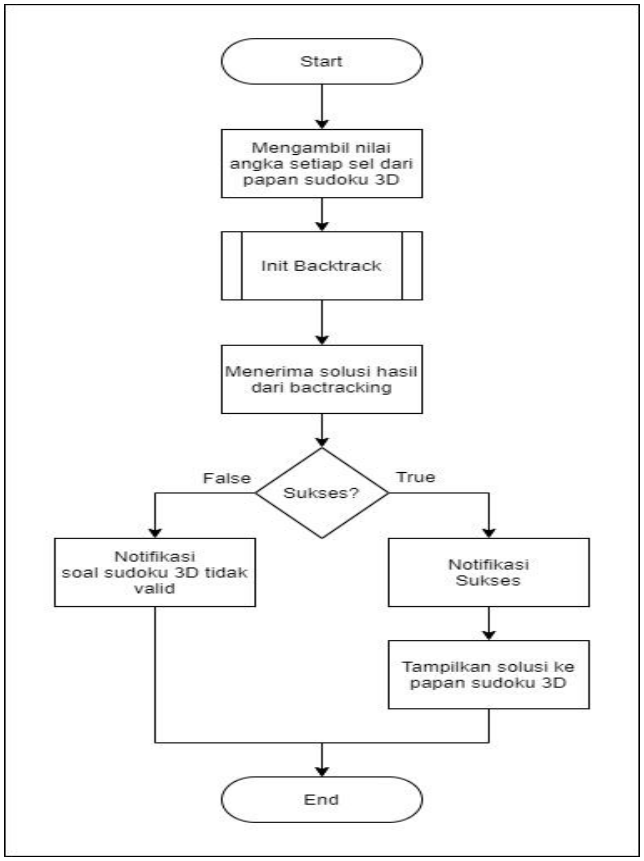
Penelitian (Nurhasanah & Saptoni, 2023) dengan judul penelitian "Penerapan Algoritma Backtracking Dengan Bounding Function Dan Depth First Search Pada Permainan Boggle. Dalam penelitian ini, kami akan menganalisis dan membandingkan penggunaan algoritma Backtracking dengan Bounding Function dan DFS dalam penyelesaian permainan Boggle. Kami akan mengevaluasi efisiensi, kompleksitas, dan keunggulan masing-masing algoritma, serta menjelajahi kemungkinan pengembangan dan peningkatan yang dapat dilakukan.Dengan pemahaman yang mendalam tentang penggunaan algoritma Backtracking dengan Bounding Function dan DFS dalam penyelesaian permainan Boggle, diharapkan penelitian ini dapat memberikan wawasan yang berharga bagi pengembangan algoritma pencarian kata-kata dalam permainan Boggle dan meningkatkan pengalaman bermain para pemain.

METODE PENELITIAN

A. Implementasi dengan *Python* dan *PyGame*

- **Metode Pengembangan**: *Rapid Application Development (RAD).*
- **Langkah Implementasi**:
    1) Menginisialisasi grid Sudoku dan menentukan sel kosong.
    2) Algoritma *backtracking* mencoba angka dari 1 hingga 9 untuk setiap sel kosong.
    3) Validasi angka dilakukan dengan memeriksa baris, kolom, dan subgrid.
    4) Jika solusi tidak valid, algoritma kembali ke langkah sebelumnya *(backtrack).*

Flowchart* Implementasi *Backtracking:



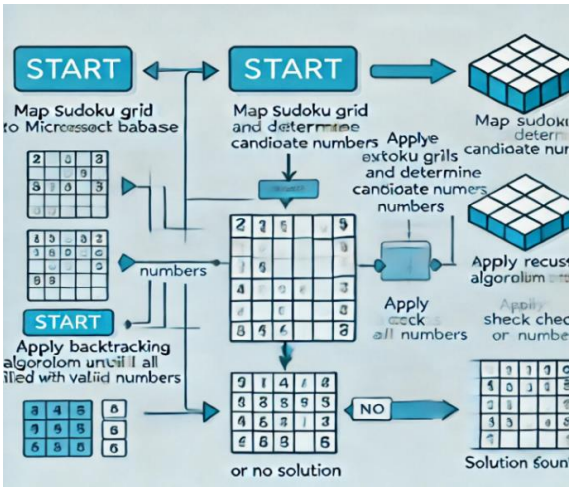
Gambar 2. Flowchart Pencarian Solusi

B. Implementasi dengan Visual Basic 6.0

- **Langkah Implementasi**:

1. Pemetaan grid Sudoku ke dalam database *Microsoft Access*.
2. Identifikasi sel kosong dan pengisian kandidat angka.
3. Penerapan algoritma *backtracking* untuk validasi angka.
4. Pengecekan rekursif dilakukan hingga seluruh sel terisi dengan angka valid.

Diagram Alur Implementasi *Visual Basic*:



Gambar 3. Diagram Alur

C. Kompleksitas Waktu dan Ruang

Kompleksitas waktu algoritma *backtracking* dalam Sudoku berukuran n×n adalah **O(n!)**, sebagai contoh, pada grid berukuran 9×9, algoritma harus mencoba hingga 9! (362.880) kemungkinan untuk satu baris penuh tanpa optimasi. Jika terdapat 20 sel kosong pada grid, maka jumlah kemungkinan yang harus diperiksa dapat mencapai 9^20 kombinasi, menunjukkan pentingnya mekanisme pruning untuk memangkas jalur yang tidak valid., karena setiap angka dalam grid memiliki kemungkinan n angka. Namun, dengan *pruning* jalur, kompleksitas dapat dikurangi.

Tabel 1

Kompleksitas *Backtracking*

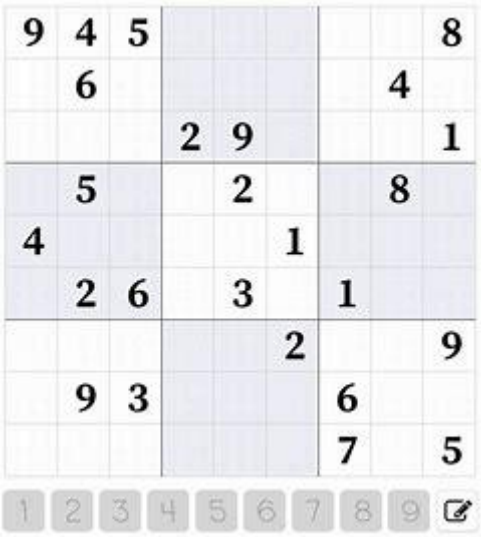
Ukuran Grid	Kompleksitas *Brute Force*	Kompleksitas *Backtracking* (Optimal)
9×9	O(9!)	O(9^n) dengan *pruning*
16×16	O(16!)	O(16^n) dengan *pruning*

HASIL PENELITIAN

A. Hasil Implementasi *Python*

- **Akurasi**: 100% pada 20 soal tingkat **hard** dan **expert**.
- **Rata-rata Waktu Penyelesaian**: 0,088 detik per soal.

Tampilan Antarmuka:



Gambar 4. Tampilan antarmuka sudoku

B. Hasil Implementasi Visual Basic

- **Akurasi:** 100%.
- **Rata-rata Waktu Penyelesaian**: 2,45 milidetik.

Perbandingan Performa:

Tabel 2

Perbandingan Performa

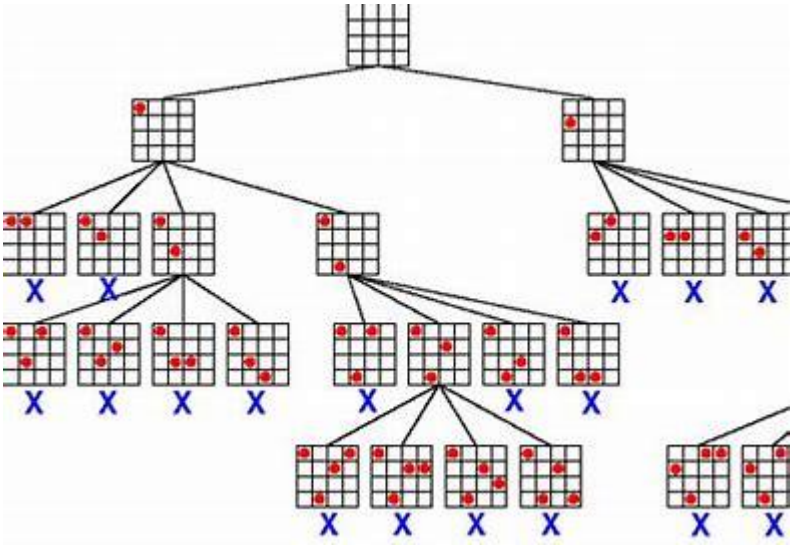
Platform	Waktu Penyelesaian (ms)	Akurasi (%)
Python	0,088	100
Visual Basic	2,45	100

PEMBAHASAN

A. Efektivitas Algoritma Backtracking

Algoritma *backtracking* efektif menyelesaikan Sudoku karena mampu memangkas jalur pencarian solusi yang tidak valid. Dengan mekanisme rekursi dan pruning jalur**,** *backtracking* hanya mengevaluasi solusi yang memiliki potensi valid.

Diagram Pohon Solusi ***Backtracking***:



Gambar 5. Diagram Pohon

B. Kelebihan dan Tantangan

- **Kelebihan**: Efisiensi pencarian solusi melalui pruning jalur.
- **Tantangan**: Kompleksitas waktu yang meningkat pada grid lebih besar.

C. Alternatif Optimasi

Untuk meningkatkan efisiensi algoritma *backtracking*, beberapa metode optimasi dapat diterapkan:

1. ***Branch and Bound***: Mengurangi jalur eksplorasi yang tidak relevan.
2. ***Dynamic Programming***: Menyimpan sub-solusi untuk menghindari perhitungan ulang.

KESIMPULAN

1. Algoritma *backtracking* efektif menyelesaikan Sudoku dengan akurasi 100%.
2. Implementasi *Python* lebih cepat dibanding *Visual Basic* dengan rata-rata 0,088 detik per soal.
3. Mekanisme *pruning* meningkatkan efisiensi pencarian solusi.
4. Tantangan pada grid besar dapat diatasi dengan optimasi seperti *branch and bound* dan *dynamic programming*.

REFERENCES

Angel, R., Agustina, W., & Yasinta, A. N. (2023). Penerapan Algoritma Backtracking Berbasis BFS dengan Pendekatan Heuristik dalam Permainan Hangman. *Jurnal Ilmiah Multidisipline*, *1*(5), 123–128. https://doi.org/10.5281/zenodo.7997018

Danuputri, C., & Santosa, N. (2021). *Aplikasi Pemecahan Soal Sudoku dengan Metode Backtracking*. *6*(3), 2622–4615. http://openjournal.unpam.ac.id/index.php/informatika506

Elvina, & Hakim, L. (2018). Modifikasi Algoritma Steepest-Ascent Hill Climbing Dan Backtracking Untuk Pencarian Lintasan Kritis Proyek Modified Steepest-Ascent Hill Climbing and Backtracking Algorithm for Project Critical Path Finding. *Cogito Smart Journal*, *4*(2), 268–282.

Ilmiah, J., Asia, I., Dewi, R., Sari, I., Kom, S., Stmik, D., & Malang, A. (2008). ANALISIS PENYELESAIAN PUZZLE SUDOKU DENGAN MENERAPKAN ALGORITMA BACKTRACKING MEMANFAATKAN BAHASA PEMROGRAMAN VISUAL BASIC 6 . 0 Sekolah Tinggi Manajemen Informatika dan Komputer ASIA Malang Sekolah Tinggi Manajemen Informatika dan Komputer ASIA Malang. *Jurnal Ilmiah Teknologi Dan Informasi*, *2*(2), 1–18.

Kiswanto, R. H. (2020). Spesifikasi Komputer Rakitan Berdasarkan Kebutuhan dan Anggaran Menggunakan Algoritma Backtracking. *Jurnal Eksplora Informatika*, *10*(1), 1–12. https://doi.org/10.30864/eksplora.v10i1.358

Krisdiawan, R. A., Fitriani, A., & Budianto, H. (2022). Penerapan Algoritma Recursive Backtracking Sebagai Maze Generator Pada Game Labirin Aksara Sunda. *Media Jurnal Informatika*, *14*(1), 31. https://doi.org/10.35194/mji.v14i1.2326

Munarto, R., & Permata, E. (2017). Perancangan sistem penjadwalan kuliah di jurusan Teknik Elektro FT.Untirta menggunakan teknik pewarnaan graph algoritma backtracking welch-powell. *Seminar Nasional Inovasi Teknologi*, 277–282.

Nurhasanah, N., & Saptoni, U. (2023). Penerapan Algoritma Backtracking Dengan Bounding Function Dan Depth First Search Pada Permainan Boggle. *Jurnal Teknik Informatika Dan Elektro*, *5*(2), 35–50. https://doi.org/10.55542/jurtie.v5i2.699

Rahmawati yunia. (2020). *Penerapan Algoritma Backtracking Pada N-Queen Problem Permainan Catur*. *3*(July), 1–23.

Rifqo Muhammad Husni, & Apridiansyah Yovi. (2017). Implementasi Algoritma Backtracking Untuk Pencarian Judul Buku. *Jurnal Pseudocode*, *Volume IV*(1), 1–7.