Penerapan Algoritma Backtracking Pada Metode Backward Chaining untuk Diagnosis Penyakit Demam Berdarah

Implementation of Backtracking Algorithm in Backward Chaining Method for Diagnosis of Dengue Fever

Rahmat Haryadi Kiswanto[1], Nourman S. Irjanto[2], M. Risman[3], Carolina C.Y. Imbiri[4]
[1,2,3,4]Teknik Informatika, Universitas Sepuluh Nopember Papua
E-mail: [1]kissonetwo74@gmail.com, [2]omanbm@gmail.com, [3]rismanm498@gmail.com, [4]carolinaimbiri@gmail.com

Abstrak

Penelitian ini meningkatkan teknik *backward chaining* untuk diagnosis Demam Berdarah Dengue (DBD) dengan mengintegrasikan algoritma *backtracking*, dengan tujuan untuk meningkatkan efisiensi penelusuran gejala dan aturan dalam sistem pakar. Hasil eksperimen menunjukkan bahwa penggunaan *backtracking* secara signifikan mengurangi waktu pemrosesan dibandingkan dengan metode tanpa *backtracking*. Pada eksperimen yang melibatkan 3 aturan, waktu pemrosesan berkurang dari 168,50 ms menjadi 19,53 ms, sementara untuk 6 aturan, waktu pemrosesan turun dari 161,63 ms menjadi 20,88 ms. Demikian pula, dalam pengujian dengan 10 aturan, waktu pemrosesan menurun dari 161,68 ms menjadi 17,38 ms. Secara keseluruhan, integrasi algoritma *backtracking* meningkatkan efisiensi penelusuran rata-rata sebesar 88%, terbukti menjadi pendekatan yang sangat efektif, terutama dalam sistem pakar yang memiliki banyak aturan dan gejala.

Kata kunci: Backward Chaining, Backtracking, Diagnosis DBD, Efisiensi

Abstract

This study enhances the backward chaining technique for Dengue Fever (DBD) diagnosis by incorporating a backtracking algorithm, with the aim of enhancing symptom and rule tracing efficiency in expert systems. The experimental outcomes reveal that employing backtracking substantially decreases processing time compared to non-backtracking methods. In experiments involving 3 rules, the processing time was reduced from 168.50 ms to 19.53 ms, while for 6 rules, it decreased from 161.63 ms to 20.88 ms. Similarly, in tests with 10 rules, the processing time dropped from 161.68 ms to 17.38 ms. Overall, the integration of the backtracking algorithm improved tracing efficiency by an average of 88%, proving to be a highly effective approach, particularly in expert systems with numerous rules and symptoms.

Keyword: Backward chaining, Backtracking, DBD Diagnosis, Efficiency

1. PENDAHULUAN

Demam Berdarah Dengue (DBD) merupakan penyakit akibat virus dengue yang ditularkan melalui gigitan nyamuk Aedes aigypti dan Aedes albopictus. Nyamuk Aedes menyukai genangan atau tempat penampungan air, seperti selokan, vas atau pot tanaman, tempat minum hewan peliharaan, kolam renang atau tempat sampah sebagai tempat perindukan. Penyakit ini merupakan masalah kesehatan masyarakat yang serius di banyak negara tropis dan sub tropis, termasuk Indonesia. Di Indonesia jumlah kasus demam berdarah cenderung meningkat setiap musim penghujan seiring dengan bermunculannya tempat perindukan. Pada tahun 2022 terdapat 143.266 kasus DBD dengan jumlah kematian 1.237 kasus. Kasus dan kematian akibat DBD ini mengalami peningkatan dibandingkan dengan tahun 2021 yaitu sebesar 73.518 kasus dan 705 kematian[1]. Secara umum gejala klinis DBD meliputi demam tinggi, nyeri otot dan sendi, serta

dapat berkembang menjadi penyakit yang mengancam jiwa jika tidak ditangani dengan cepat dan tepat.

Diagnosis dini DBD menjadi kunci dalam penanganan efektif penyakit ini. Namun diagnosis penyakit ini sering gejalanya dapat bervariasi dan menyerupai dengan penyakit lain. Oleh karena itu metode diagnosis yang tepat dan efisien menjadi krusial dalam upaya penanganan DBD. Dalam bidang teknologi informasi dan komunikasi beberapa pengembangan sistem pakar dalam upaya untuk mendiagnosis dini gejala penyakit ini menggunakan metode *backward chaining* [2], [3]. Backward chaining merupakan teknik inferensi dalam sistem pakar berbasis aturan. Metode ini dikatakan lebih efektif dibandingkan metode *forward chaining* dalam diagnosis penyakit [4], [5]. Metode *backward chaining* ini juga telah digunakan pada beberapa penelitian diagnosis penyakit selain penyakit DBD [2], [6], [7], [8], [9], [10].

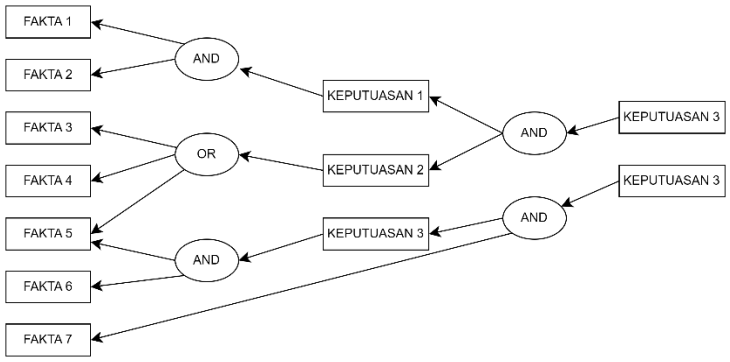
Metode *backward chaining* pada sistem pakar digunakan untuk penalaran mundur berdasarkan goal untuk menentukan sebab akibat dari suatu kejadian dan gejala. Metode ini digunakan untuk mengidentifikasi kemungkinan gejala yang muncul pada pasien, sehingga dapat membantu mendiagnosis dini DBD. Cara penelusuran fakta pada *backward chaining* menggunakan prinsip *Depth First Search* (DFS). Namun demikian penelusuran fakta-fakta gejala dapat menjadi kurang efisien dan kurang optimal jika jumlah fakta-fakta yang harus ditelusuri sangat banyak dan mirip dengan fakta-fakta penyakit lain, yang mana prosesnya bisa memakan waktu dan sumber daya komputasi yang besar. Kondisi ini dapat berdampak pada hasil diagnosis penyakit menjadi tidak optimal. Hasil yang tidak optimal bisa berdampak pada penanganan pengobatan penyakit DBD menjadi tidak tepat. Penelitian [4], metode backward chaining menggunakan prinsip DFS sehingga memiliki keterbatasan dalam kecepatan pemrosesan dan efisiensi saat digunakan digunakan dengan basis pengetahuan yang luas. Algoritma inferensi mundur adalah rekursif, rekursi digunakan untuk mengonfirmasi sub-tujuan inferensi, yang merupakan kondisi aturan yang saat ini bukan fakta. Sayangnya, jumlah rekursif yang tidak perlu mungkin sangat tinggi untuk basis aturan yang besar. Setiap panggilan rekursif yang tidak perlu membutuhkan waktu dan memperlambat proses konfirmasi tujuan utama inferensi [11].

Berdasarkan inefisiensi dan ketidakoptimalan penelusuran fakta dari metode *backward chaining*, maka penelitian ini akan mengoptimalkan proses penelusuran fakta pada metode *backward chaining* dengan menerapkan algoritma *backtracking*. Pada beberapa penelitian terdahulu algoritma *backtracking* terbukti dapat meningkatkan kinerja beberapa metode pada sistem pakar. Penelitian [12], algoritma *backtracking* dapat meningkatkan kinerja dari inferensi metode *Bayesian*. Pada penelitian [13], pemangkasan pada algoritma *backtracking* menjadi cara yang bernilai untuk menyimpan rekursif. Selanjutnya penelitian [14], *backtracking* diterapkan pada optimalisasi kombinasi perangkat keras komputer rakitan sehingga proses penelusuran kombinasi menjadi lebih efisien dari sisi komputasi dan efektif dari sisi performa komputer berdasarkan batasan anggaran yang diberikan.

2. METODE PENELITIAN

2.1. Metode *Backward Chaining*

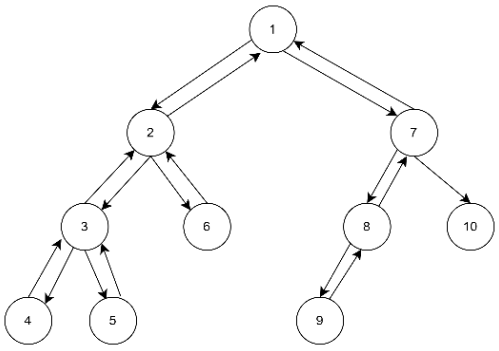
Diketahui metod backward chaining bekerja mirip dengan cara dokter mendiagnosis penyakit, yaitu mulai dari tujuan (*goal*) kemudian bergerak mundur mencocokkan fakta-fakta yang sesuai dengan tujuan, lihat gambar 1.



Gambar 1 Proses Kerja *Backward Chaining*

2.2. Algoritma *Backtracking*

Algoritma *backtracking* adalah algoritma yang dapat digunakan untuk menemukan solusi dari suatu masalah kombinatorial dengan membuat urutan dan pada titik tertentu jika pilihan tidak mengarah ke solusi maka akan mundur (membatalkan pilihan lanjutan) ke keadaan valid terakhir dan mencoba pilihan berbeda, lihat gambar 2. Pada algoritma ini terdapat fungsi rekursif yang memberikan efisiensi pada proses penelusuran kembali ke ruang pencarian jika penelusuran sebelumnya tidak mengarah ke solusi.

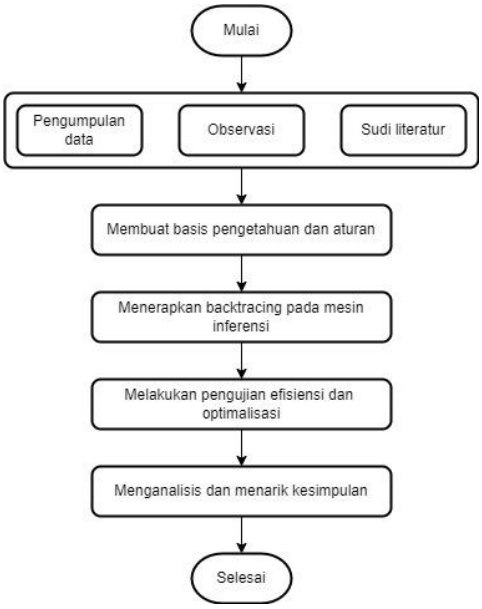


Gambar 2 Proses Penelusuran algoritma *backtracking*

Secara teoritis algoritma *backtracking* dapat digunakan untuk permasalahan efisiensi dan optimalisasi kombinasi. Pada permasalahan diagnosis awal penyakit DBD menggunakan *backward chaining* adalah pada permasalahan efisiensi dan efektifitas kombinasi fakta-fakta yang valid terhadap *goal* (hipotesis awal penyakit).

2.3. Tahapan Penelitian

Supaya penelitian ini dapat berjalan dengan terstruktur dan sesuai dengan target yang diinginkan, maka disusun tahapan penelitian yang dapat dilihat pada gambar 3.



Gambar 3. Tahapan Penelitian.

2.5 Pendekatan pemecahan Masalah.

Pendekatan pemecahan masalah optimalisasi dengan *backtracking*:

1. Pemangkasan awal: dengan menentukan fungsi pembatas sehingga aturan dan fakta yang tidak relevan dengan tujuan dapat diidentifikasi dan dipangkas.
2. Pencarian terpadu: menggunakan heuristic untuk memprioritaskan aturan atau fakta yang lebih mengarah ke solusi. Pengurutan aturan yang memiliki fakta-fakta yang hampir sama diprioritaskan terurut untuk penyakit demam berdarah, baru kemudian diikuti dengan aturan-aturan dari penyakit lain.
3. Penggunaan cache: bertujuan untuk menyimpan hasil inferensi sebelumnya untuk menghindari perhitungan ulang.

2.6 Pengujian.

Pengujian efisiensi dilakukan dengan menggunakan analisis kompleksitas yang berfokus pada evaluasi kinerja algoritma dalam hal ini kebutuhan waktu dan ruang. Langkah-langkah untuk menguji kompleksitas algoritma:

1. Buat kasus uji dengan berbagai ukuran masukan untuk mengevaluasi kinerja algoritma.
2. Gunakan fungsi pengaturan waktu untuk mengukur waktu eksekusi algoritma untuk ukuran masukan berbeda.
3. Plot waktu eksekusi terhadap ukuran masukan untuk memvisualisasikan tingkat pertumbuhan dan membandingkannya dengan ekspektasi teoritis.
4. Pantau penggunaan memori selama eksekusi algoritma untuk berbagai ukuran input.

Demam berdarah disebabkan oleh virus dengue yang dibawa oleh nyamuk Aedes aegypti dan Aedes albopictus. Ada empat serotipe virus dengue (DENV-1, DENV-2, DENV-3, dan DENV-4), yang dapat menyebabkan beberapa bentuk penyakit demam berdarah. Berikut adalah jenis-jenisnya:

1. Demam Dengue (*Dengue Fever*)
   Bentuk yang lebih ringan dari dari infeksi degue.
2. Demam Berdarah Dengue (*Dengue Hemorrhagic Fever* - DHF)

DHF lebih parah dan bisa menyebabkan syok atau perdarahan hebat jika tidak ditangani dengan cepat.

3. *Dengue Shock Syndrome* (DSS).
   DSS adalah kondisi yang mengancam jiwa dan memerlukan perawatan medis segera.

Tabel 1. Gejala – Gejala Penyakit Dan Jenis Demam Berdarah

Id	Gejala (*Symtomp*)	Dengue Fever	DHF	DSS
S01	Demam tinggi	√	√	√
S02	Sakit kepala parah	√	√	√
S03	Sakit perut parah		√	√
S04	Shock			√
S05	Tekanan darah rendah			√
S06	Pendarahan dari hidung, gusi, atau kulit menyebabkan memar berwarna ungu		√	√
S07	Pendarahan parah			√
S08	Nyeri pada bagian belakang mata	√	√	√
S09	Muntah terus menerus		√	√
S10	Nyeri otot	√	√	√
S11	Nyeri sendi	√	√	√
S12	Kebocoran di luar pembuluh darah			√
S13	Sulit bernafas setelah demam awal mereda		√	√
S14	Kerusakan pada pembuluh darah dan getah bening		√	√
S15	Mual dan muntah	√		
S16	Bintik-bintik merah (ruam) 3 -4 hari setelah demam	√	√	√

Data gejala penyakit dan jenis demam berdarah pada tabel 1 diperoleh dari wawancara dengan pakar, yaitu dokter dari PUSKESMAS Elly Uyo di Jayapura Selatan, Papua.

3. HASIL DAN PEMBAHASAN

3.1. Membangun Basis Pengetahuan

Data gejala penyakit yang sudah diperoleh dimasukan kedalam basis pengetahuan dalam bentuk aturan-aturan yang terdiri dari hipotesis dan premis-premis dapat dilihat pada tabel 2.

Tabel 2. Basis Pengetahuan

Id rule	Hipotesis	Premis - premis
R01	Dengue Fever	S01 is True AND S02 is True AND S08 is True AND S10 is True AND S11 is True AND S15 is True AND S16 is True
R02	DHF	S01 is True AND S02 is True AND S03 is True AND S06 is True AND S08 is True AND S09 is True AND S10 is True AND S11 is True AND S13 is True AND S14 is True AND S16 is True
R03	DSS	S01 is True AND S02 is True AND S03 is True AND S04 is True AND S05 is True AND S06 is True AND S07 is True AND S08 is True AND S09 is True AND S10 is True AND S11 is True AND S12 is True AND S13 is True AND S14 is True AND S16 is True

3.2. Pemecahan Masalah

Pemecahan masalah optimasi untuk penelusuran fakta dilakukan dengan cara untuk setiap gejala yang tersimpan direpresentasikan dalam model pohon keputusan, sehingga pohon keputusan akan terdiri atas simpul/*node* dan *edge*. Simpul akan mewakili representasi dari gejala, sedangkan *edge* akan mewakili representasi dari koneksi antara setiap gejala. Setelah struktur dari pohon keputusan terbentuk kemudian menentukan fungsi pembatas, dimana dalam hal ini fungsi pembatas adalah jumlah gejala yang cocok dengan aturan pada basis pengetahuan. Jika jumlah gejala yang diperlukan tidak terpenuhi, maka kita bisa menghentikan pencarian pada aturan. Hal ini dapat membantu mengurangi eksplorasi lebih dalam yang tidak relevan.

Diberikan sebagai berikut:

1. S adalah himpunan semua gejala yang dialami pasien.
2. R_1, R_2, R_3, …, R_k adalah aturan diagnosis demam berdarah dengan gejala-gejala terkait.
3. C(R) adalah fungsi yang menghitung jumlah gejala dalam aturan R yang sesuai dengan gejala yang ada pada pasien.

Fungsi pembatas B dapat didefinisikan sebagai berikut:

$$B(R,S) = \begin{cases} True & jika\ C(R) = |R| \\ False & jika\ C(R) < \min\_gejala \end{cases} \quad (1)$$

dimana:

- |R| : jumlah total gejala yang diperlukan untuk aturan R
- min_gejala : jumlah minimal gejala yang harus cocok agar pencarian dilanjutkan
- C(R) : Fungsi yang menghitung berapa banyak gejala dalam aturan R yang terdapat dalam himpunan gejala pasien

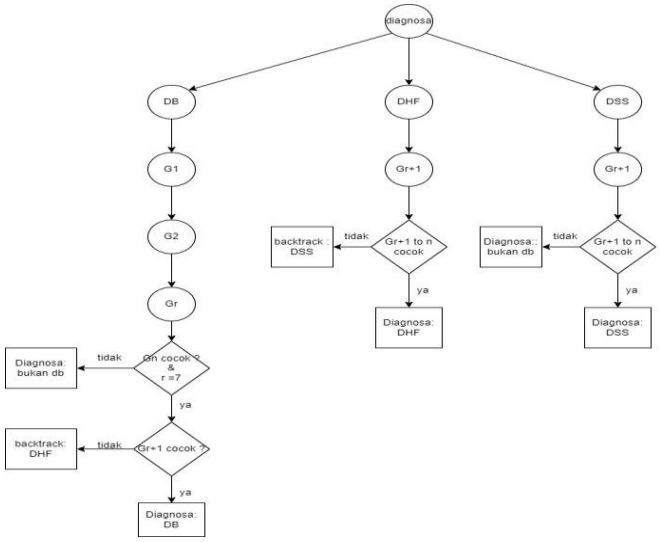
Jika fungsi pembatas mengembalikan *False*, pencarian pada aturan tersebut dihentikan, karena tidak mungkin aturan akan terpenuhi.

3.3. Implementasi fungsi pembatas

Pada kasus demam berdarah ini, berdasarkan tabel 2 bahwa treshold seseorang terindikasi demam berdarah = 7, sehingga kita bisa mendefinisikan fungsi pembatas sebagai:

$$B(R,S) = \begin{cases} True & jika\ C(R)\ \geq\ 7\ (Treshold\ atau\ pembatas) \\ False & jika\ C(R) < 7 \end{cases} \quad (2)$$

Misalkan Gejala yang dialami pasien = ['demam tinggi', 'sakit kepala parah', 'nyeri sendi', 'nyeri otot', 'nyeri pada bagian belakang mata', 'mual dan muntah', 'ruam kulit', 'sulit bernafas']
Untuk jumlah gejala di atas adalah 8 dan memenuhi fungsi pembatas C(R01) = 7, sehingga untuk R01 bernilai *True* dan pencarian di lanjutkan ke gejala ke 8 tetapi untuk R01 gejala ke 8 tidak ada sehingga bernilai *False* , maka mundur ke R02 tetapi disini penelusuran gejala tidak lagi dilakukan dari awal karena gejala yang sebelumnya telah disimpan di variabel *memoization* tetapi langsung pengecekan ke gejala ke 8 dan bernilai *True*. Kemudian dilanjutkan pengecekan ke gejala ke 9 untuk R02 tetapi disini bernilai *False*, sehingga mundur ke R03 pada pengecekan gejala R03 ini juga tidak dilakukan dari awal lagi tetapi langsung pengecekan pada gejala yang tidak tersimpan pada variable *memorization*, saat pengecekan di R03 ternyata gejala yang diderita pasien tidak sesuai lagi dengan gejala di R03 dan penelusuran dihentikan. Dari hasil penelusuran maka kesimpulan pasien menderita penyakit demam berdarah karena gejala yang dialami pasien bernilai *True* untuk R01. Jika direpresentasikan dalam bentuk pohon keputusan setiap kemungkinan gejala penyakit dapat dilihat pada gambar 4.



Gambar 4. Pohon keputusan penelusuran gejala penyakit demam berdarah

Algoritma untuk fungsi pembatas pada backward chaining:

```
function bounding_function(rule, gejala, pembatas):
    match_count = 0;
    for each condition in rule:
        if condition exists in gejala:
            match_count += 1
    if match_count >= pembatas:
        return true
    else:
        return false

function backward_chaining_with_bounding(penyakit, gejala, memo, pembatas):
    if penyakit exists in memo:
      return memo[penyakit]
    if penyakit not in rules:
      return False, 0
    best_match_count = 0
    for each rule in rules[penyakit]:
     // Cek kecocokan dengan bounding function
        match_count, valid = bounding_function(rule, gejala, pembatas)
    if valid:
        if match_count > best_match_count:
            best_match_count = match_count
        // Cek apakah semua kondisi dalam aturan terpenuhi
        all_conditions_met = True
        for each condition in rule:
            if condition not in gejala:
            all_conditions_met = False
            break
        if all_conditions_met:
            memo[penyakit] = (True, best_match_count)
                return True, best_match_count
    memo[penyakit] = (False, best_match_count)
    return False, best_match_count
```

3.4. Pengujian

Untuk pengujian optimalisasi terhadap metode *backward chaining* dilakukan dengan menggunakan analisis kompleksitas yang berfokus pada evaluasi kinerja algoritma dalam hal ini kebutuhan waktu dan ruang.

A. Kompleksitas *Backward Chaining* tanpa fungsi pembatas.

Setiap aturan diperiksa dengan lengkap tanpa pemangkasan berdasarkan batasan tertentu. Dengan demikian algoritma akan menelusuri semua aturan dan mengecek apakah semua kondisi dalam setiap aturan dipenuhi oleh gejala pasien, bahkan jika sebagian besar gejala tidak cocok. Hal ini dapat menyebabkan waktu eksekusi lebih lama dan kompleksitas tinggi.

Kompleksitas:

Jumlah aturan : n penyakit dipertimbangkan, dan masing-masing penyakit memiliki m aturan

Jumlah gejala per aturan: setiap aturan memiliki k gejala.

Untuk setiap aturan, algoritma harus memeriksa apakah setiap gejala pada aturan cocok dengan gejala pasien, yang memerlukan perbandingan sebesar *O(k)*. Sehingga kompleksitas waktu dapat ditulis *O(n x m x k)*.

B. Kompleksitas *Backward Chaining* dengan fungsi pembatas (*backtracking*).

Dengan fungsi pembatas *backtracking*, kita menggunakan pemangkasan (*pruning*) untuk mengurangi pencarian aturan yang tidak perlu. Setelah memeriksa beberapa gejala pada sebuah aturan, jika ditemukan bahwa aturan tersebut tidak memenuhi ambang batas minimal gejala yang cocok, maka proses pencarian dihentikan lebih awal (*pruning*), dan aturan berikutnya diperiksa. Ini akan mengurangi waktu eksekusi dalam skenario di mana banyak aturan tidak cocok dengan gejala pasien.

Kompleksitas sama seperti aturan sebelumnya, hanya karena ada *pruning* dengan *bounding function*, kompleksitas waktu dipangkas berdasarkan seberapa cepat menghentikan pemeriksaan aturan. Jika banyak aturan tidak memenuhi ambang batas minimal (*threshold*) gejala yang cocok, maka algoritma dapat berhenti lebih awal.

Kompleksitas waktu terbaik *O(n x m)*, karena bisa banyak aturan yang mungkin tidak memerlukan pengecekan semua *k* gejala. Kompleksitas waktu terburuk (*worst-case complexity*) adalah sama dengan komplesitas waktu tanpa pruning: *O(n x m x k)*.

Berdasarkan analisis menggunakan kompleksitas algoritma maka perbandingan kompleksitas tanpa dan dengan diterapkannya algoritma *backtracking* pada metode *backward chaining* dapat dilihat pada tabel 3.

Tabel 3. Perbandingan kompleksitas

Pendekatan	**Kompleksitas waktu (*worst-case*)**	**Kompleksitas ruang**	**Keterangan**
Backward chaining tanpa pembatas (backtracking)	*O(n x m x k)*	*O(n x m x k)*	Algoritma akan memeriksa semua aturan dan gejala tanpa menghentikan pencarian lebih awal
Backward chaining dengan pembatas (backtracking)	*O(n x m)* hingga *O(n x m x k) (best – worst case)*	*O(n x m x k)*	Pruning menghentikan pencarian lebih awal jika aturan tidak memenuhi ambang batas gejala.

C. Pengujian dengan beberapa ukuran masukan.

Setelah diperoleh kompleksitas algoritma. Kemudian dilakukan pengukuran waktu eksekusi algoritma dengang beberapa ukuran masukan. Sehingga pada pengujian ini ditambahkan beberapa rule penyakit diluar penyakit demam berdarah sebanyak 7 penyakit tambahan yaitu malaria tropika, malaria tertiana, malaria quartana, malaria ovale, influenza, diabetes militus dan

infeksi bakteri. Pengujian dilakukan terhadap tiga kondisi *rule* dengan setiap kondisi dilakukan 5 kali pengukuran, yaitu pertama dengan 3 *rule*, kedua dengan 6 *rule*, dan ketiga dengan 10 *rule* untuk masing-masing tanpa menggunakan *backtracking* dan dengan menggunakan *backtracking*. Pengujian dengan 3 rule yaitu *rule* untuk demam berdarah, DHF dan DSS, Pengujian dengan 6 *rule*, yaitu *rule* demam berdarah, DHF, dan DSS, diabetes militus dan influenza, dan Pengujian dengan 10 rule, yaitu *rule* demam berdarah, DHF, dan DSS, diabetes militus, malaria tropika, malaria dan influenza. Tabel 4 merupakan hasil pengujian, dimana semua pengujian menggunakan kasus untuk gejala DHF.

Tabel 4. Hasil Pengujian

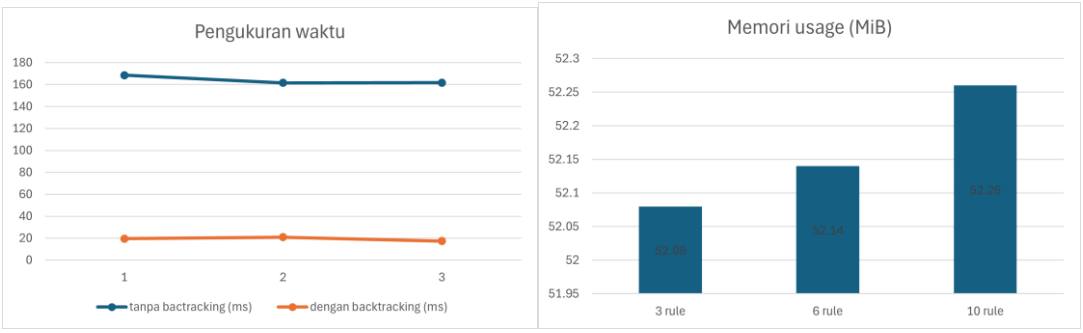
Jumlah rule	**Pengujian**	**Tanpa backtracking**		**Dengan backtracking**	
		Pengguanaan memori (MiB)	**Waktu eksekusi (ms)**	**Pengguanaan memori (MiB)**	**Waktu eksekusi (ms)**
3	1	52.3	161.62014	52.3	18.753052
	2	51.9	160.209417	51.9	16.667604
	3	52.1	170.728683	52.1	19.318342
	4	52.1	177.624702	52.1	21.990299
	5	52	172.333241	52	20.942926
6	1	52	156.18062	52	23.008823
	2	52.2	162.840128	52.2	16.575575
	3	52.1	161.82065	52.1	17.746925
	4	52.3	165.252209	52.3	30.044794
	5	52.1	162.066221	52.1	17.050982
10	1	52.4	158.42104	52.4	17.202139
	2	52.1	160.159349	52.1	19.34576
	3	52.3	167.839289	52.3	16.427517
	4	52.2	160.805941	52.2	16.25824
	5	52.3	161.149502	52.3	17.662048

Dari hasil pengujian pada tabel 4, kemudian kita buat rata-rata hasil pengujian untuk setiap jumlah rule, seperti yang ditampilkan pada tabel 5.

Tabel 5. Nilai Rata-Rata Untuk Setiap Rule

Jumlah rule	**Jumlah Pengujian**	**Tanpa backtracking**		**Dengan backtracking**		**Persentase kenaikan (%)**
		Pengguanaan memori (MiB)	**Waktu eksekusi (ms)**	**Pengguanaan memori (MiB)**	**Waktu eksekusi (ms)**	
3	5	52.08	168.5032366	52.08	19.5344446	88.4070
6	5	52.14	161.6319656	52.14	20.8854198	87.0784
10	5	52.26	161.6750242	52.26	17.3791408	89.2506

Gambar 5 merupakan visualisasi perbandingan pengujian waktu eksekusi dan penggunaan memori



Gambar 5. Visualisasi Waktu Eksekusi dan Penggunaan Memori

Dari hasil pengujian dapat dilihat bahwa metode *backward chaining* dengan menerapkan algoritma *backtracking* lebih efisien dari sisi penelusuran fakta dibandingkan tanpa menerapkan algoritma *backtracking*, dimana terjadi kenaikan yang cukup signifikan rata-rata waktu eksekusi sebesar 88% setelah menggunakan algoritma *backtracking*, hal ini dikarenakan adanya fungsi pembatas yang akan memangkas penelusuran yang tidak perlu dan *memoization* yang digunakan untuk menyimpan hasil penelusuran fakta jika terdapat gejala yang sama diantara beberapa *rule* yang terdapat pada basis pengetahuan. Berdasarkan pada penelitian [4], [11] yang menyampaikan keterbatasan dan kekurangan dari metode backward chaining yaitu pada proses penelusuran fakta dengan cara rekursif sehingga akan memakan waktu yang cukup banyak, maka pengintegrasian algoritma backtracking telah memberikan pengoptimalan dari sisi efisiensi dan fleksibilitas penelusuran fakta pada teknik inferensi dari metode backward chaining.

4. KESIMPULAN DAN SARAN

Hasil Penelitian menunjukkan bahwa penerapan algoritma *backtracking* berhasil mengoptimalkan proses penelusuran fakta pada metode *backward chaining* dalam diagnosis dini penyakit demam berdarah. Dengan menggunakan fungsi pembatas dan teknik *memoization*, algoritma *backtracking* mampu memangkas penelusuran aturan yang tidak relevan dan menghindari pengulangan evaluasi gejala yang sudah dianalisis. Hasil pengujian memperlihatkan peningkatan efisiensi, dari segi waktu eksekusi. Implementasi algoritma backtracking meningkatkan kinerja *backward chaining* hingga rata-rata 88% lebih cepat dibandingkan tanpa menggunakan *backtracking*. Peningkatan ini sangat signifikan dalam sistem pakar berbasis aturan yang menangani berbagai gejala penyakit secara kompleks. Oleh karena itu, integrasi algoritma *backtracking* dalam *backward chaining* dapat menjadi solusi yang efektif dalam diagnosis penyakit yang kompleks seperti demam berdarah.

Hasil penelitian fokus pada teknik penerapan algoritma *backtracking* pada metode *backward chaining* untuk optimalisasi penelusuran gejala / fakta penyakit, bukan pada akurasi hasil diagnosis, sehingga kedepannya penelitian ini dapat dikembangkan dari sisi akurasi diagnosis yang bisa mengkombinasikannya lagi dengan algoritma lain seperti algoritma yang berhubungan dengan *uncertainty factor* karena biasanya pasien bisa saja memberikan jawaban yang agak samar atas gejala yang dideritanya.

UCAPAN TERIMA KASIH

Terima kasih kepada Direktorat Riset, Teknologi dan Pengabdian kepada Masyarakat (DRTPM) yang telah memberikan hibah penelitian, sehingga penelitian ini dapat dilaksanakan dengan baik mengikuti prosedur yang telah ditetapkan. Semoga dengan adanya hibah dari DRTPM ini akan terus memberikan motivasi yang besar bagi peneliti untuk terus berkarya.

DAFTAR PUSTAKA

[1] "Profil Kesehatan Indonesia 2022," *Kementerian Kesehatan Republik Indonesia*, pp. 218–238, 2022.

[2] R. Yogie Aldiansyah, "Sistem Pakar Diagnosa Penyakit Demam Berdarah Dengue Menggunakan Metode Backward Chaining Berbasis Android (Studi Kasus Klinik Pratama Sartika 77)," *Ubiquitous: Computers and its Applications Journal*, vol. 3, no. 1, pp. 27–34, 2020.

[3] R. Wariyanto Abdullah and F. Prasetyo Nugroho, "Sistem Pakar Deteksi Penyakit Tipes, DBD, Campak dan Diare Dengan Metode Backward Chaining," 2019.

[4] A. Al-Ajlan, "The Comparison between Forward and Backward Chaining," *Int J Mach Learn Comput*, vol. 5, no. 2, pp. 106–113, Apr. 2015, doi: 10.7763/ijmlc.2015.v5.492.

[5] I. Akil, "Analisa Efektifitas Metode Forward Chaining Dan Backward Chaining Pada Sistem Pakar," *Jurnal Pilar Nusa Mandiri*, vol. 13, no. 1, pp. 35–42, 2017, [Online]. Available: https://ejournal.nusamandiri.ac.id/index.php/pilar/article/view/12

[6] R. H. Kiswanto, S. Bakti, and R. M. H. Thamrin, "Rancang Bangun Sistem Pakar Diagnosa Penyakit Kucing Menggunakan Metode Backward Chaining," *Jurnal Eksplora Informatika*, vol. 11, no. 1, pp. 67–76, Jan. 2022, doi: 10.30864/eksplora.v11i1.610.

[7] S. A. Rahmah, A. Voutama, and S. Sobur, "Sistem Pakar Diagnosis Obesitas Pada Orang Dewasa Menggunakan Metode Backward Chaining," *INTECOMS: Journal of Information Technology and Computer Science*, vol. 4, no. 2, pp. 169–177, 2021.

[8] H. Sholikhin, D. Wahiddin, and K. A. Baihaqi, "Penerapan Algoritma Backward Chaining Untuk Mendiagnosa Penyakit Dan Hama Tanaman Padi," *Scientific Student Journal for Information, Technology and Science*, vol. III, no. 1, pp. 22–27, 2022.

[9] J. Manajemen, A. Chindianto, D. Oktiviani, H. Sya'ban Triaji, and H. Isnanto, "Analisa Sistem Pakar Diagnosa Penyakit Covid-19 Berbasis Online Menggunakan Metode Backward Chaining," vol. 1, no. 1, 2022, [Online]. Available: https://journal.mediapublikasi.id/index.php/bisik

[10] A. R. Jari and M. F. Asnawi, "Rancang Bangun Sistem Pakar Diagnosa Penyakit Pada Buah Salak Berbasis Web Menggunakan Algoritma Backward Chaining," vol. 2, no. 1, pp. 15–19, 2023, doi: 10.55123.

[11] R. Simiński, "Extraction of Rules Dependencies for Optimization of Backward Inference Algorithm," 2014, pp. 191–200. doi: 10.1007/978-3-319-06932-6_19.

[12] F. Bacchus, S. Dalmao, and T. Pitassi, "Value elimination: Bayesian inference via backtracking search," *Proceedings of the Nineteenth …*, pp. 20–28, 2002, [Online]. Available: http://dl.acm.org/citation.cfm?id=2100587

[13] J. Sleegers and D. van den Berg, "Backtracking (the) Algorithms on the Hamiltonian Cycle Problem," 2021, [Online]. Available: http://arxiv.org/abs/2107.00314

[14] R. H. Kiswanto, "Spesifikasi Komputer Rakitan Berdasarkan Kebutuhan dan Anggaran Menggunakan Algoritma Backtracking," *Jurnal Eksplora Informatika*, vol. 10, no. 1, pp. 1–12, 2020, doi: 10.30864/eksplora.v10i1.358.