# Analisis Kinerja Algoritma Backtracking untuk Pemecahan Masalah Cryptarithm

Fauzan Mohamad Abdul Ghani - 13524113
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: fauzan210mohamad@gmail.com , 13524113@std.stei.itb.ac.id

***Abstract*—Cryptarithm merupakan teka-teki matematis yang mengganti digit pada suatu operasi aritmatika dengan huruf, sehingga penyelesaiannya berupa pencarian pemetaan huruf-ke-digit di mana 1 digit hanya dapat ditampung oleh 1 huruf, begitupun sebaliknya. Pendekatan brute force yang mencoba seluruh permutasi pemetaan memerlukan biaya komputasi besar, mencapai 10! Kemunkinan untuk puzzle dengan 10 huruf berbeda. Makalah ini menganalisis kinerja algoritma backtracking dengan pemangkasan berbasis penjumlahan per kolom. Penulis menerapkan dan membandingkan tiga algoritma berbeda, yaitu brute force, backtracking dasar, dan backtracking dengan pemangkasan berbasis penjumlahan per kolom. Pengujian dilakukan pada sejumlah puzzle dengan tingkat kesulitan beragam, menggunakan metrik jumlah simpul yang dibangkitkan dan waktu eksekusi.**

***Keywords—cryptarithm; backtracking; pemangkasan; pemetaan huruf-ke-digit;***

## I. PENDAHULUAN

Cryptarithm merupakan teka-teki matematis di mana digit-digit pada sebuah operasi aritmatika digantikan oleh suatu huruf, sedemikian sehingga setiap huruf yang berbeda mewakili satu digit yang berbeda. Tujuan penyelesaiannya adalah menemukan pemetaan huruf-ke-digit yang membuat operasi aritmatika tersebut bernilai benar. Contoh penggunaanya adalah dalam penjumlahan SEND + MORE = MONEY, yang memiliki solusi unik S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2 [1]. Persoalan ini tergolong ke dalam persoalan kombinatorika sekaligus *constraint satisfaction problem*, karena solusinya berupa salah satu dari sekian banyak permutasi penetapan digit yang harus memenuhi sejumlah kendala sekaligus.

Pendekatan brute force memungkinkan untuk menyelesaikan cryptarithm dengan mencoba seluruh kemungkinan pemetaan huruf-ke-digit dan baru memeriksa kebenaran persamaan setelah semua huruf memperoleh nilai. Karena terdapat sepuluh digit angka (0-9), sebuah persoalan dengan sepuluh huruf berbeda memiliki hingga 10! atau sekitar 3,6 juta kemungkinan pemetaan yang harus dievaluasi satu per satu. Karena seiring bertambahnya jumlah huruf jumlah kemungkinan akan terus tumbuh, maka pendekatan brute force menjadi tidak efektif untuk menyelesaikan persoalan berukuran besar.

Algoritma backtracking memungkinkan untuk membangun solusi sebuah persoalan secara bertahap dan memangkas cabang-cabang yang sudah dipastikan tidak mengarah ke solusi melalui sebuah fungsi pembatas (*bounding function).* Pada cryptarithm, pemangkasan dapat dilakukan dengan memeriksa konsistensi penjumlahan dan nilai *carry* per kolom segera setelah huruf-huruf yang relevan ditetapkan, sehingga Sebagian besar ruang pencarian dapat dibuang lebih awal tanpa perlu dievaluasi.

Meskipun secara konseptual backtracking lebih unggul, seberapa besar keunggulan tersebut dalam praktik bergantung pada kualitas pemangkasan yang diterapkan dan karakteristik persoalan. Oleh karena itu, makalah ini menganalisis kinerja algoritma backtracking dalam menyelesaikan persoalan cryptarithm dengan cara mengimplementasikan dan membandingkannya terhadap pendekatan brute force sebagai baseline. Perbandingan dilakukan menggunakan metrik jumlah simpul yang dibangkitkan dan waktu eksekusi pada sejumlah persoalan dengan tingkat kesulitan beragam.

Terdapat beberapa permasalahan yang dibahas dalam makalah ini. Pertama, bagaimana cara memodelkan persoalan cryparithm ke dalam kerangka algoritma backtracking, yang mencakup penentuan vektor solusi, fungsi pembangkit, dan fungsi pembatas. Kedua, bagaimana perbandingan kinerja algoritma backtracking terhadap algoritma brute force ketika keduanya digunakan untuk menyelesaikan persoalan cryptarithm. Ketiga, seberapa besar pengaruh strategi pemangkasan terhadap jumlah simpul yang dieksplorasi dan waktu eksekusi, serta bagaimana pengaruh tersebut berubah seiring bertambahnya jumlah huruf berbeda dalam persoalan.

Makalah ini bertujuan untuk memodelkan persoalan cryptarithm ke dalam kerangka formal algoritma backtracking, kemudian mengimplementasikan algoritma brute force dan backtracking untuk menyelesaikannya. Selanjutnya, makalah ini bertujuan menganalisis dan membandingkan kinerja kedua algoritma secara kuantitatif berdasarkan hasil eksperimen, sehingga dapat diketahui sejauh mana strategi pemangkasan pada backtracking mampu meningkatkan efisiensi penyelesaian persoalan cryptarithm.

## II. Dasar Teori

### A. Algoritma Brute Force dan Exhaustive Search

Algoritma brute force adalah pendekatan penyelesaian persoalan secara langsung yang umumnya diturunkan langsung dari pernyataan atau definisi persoalan. Pendekatan ini sederhana dan mudah untuk diimplementasikan, serta menjamin ditemukannya solusi apabila solusi dari persoalan terkait memang ada. Kelemahannya, brute force memerlukan waktu komputasi yang sangat lama terutama untuk persoalan berukuran besar, sehingga sering kali hanya digunakan sebagai basis pembanding bagi algoritma lain yang lebih efisien [1].

Exhaustive search adalah salah satu bentuk algoritma brute force yang khusus diterapkan pada persoalan kombinatorika, yaitu persoalan yang kemungkinan solusinya berupa permutasi, kombinasi, atau himpunan bagian dari sekumpulan objek diskrit. Langkah-langkahnya adalah mengenumerasi seluruh kemungkinan solusi secara sistematis, mengevaluasi setiap kemungkinan yang ada satu per satu, lalu memilih solusi yang benar atau solusi yang terbaik. Karena seluruh ruang solusi diperiksa, kompleksitas waktu umumnya tumbuh secara eksponensial atau factorial terhadap ukuran masukan [3].

### B. Algoritma Backtracking

Algoritma backtracking merupakan perbaikan dari exhaustive search. Jika pada exhaustive search seluruh kemungkinan solusi dibangkitkan dan dievaluasi, maka pada backtracking hanya pilihan yang mengarah ke solusi yang dieksplorasi, sedangkan pilihan yang tidak mengarah ke solusi langsung diabaikan. Dengan kata lain, backtracking memangkas simpul-simpul yang tidak mengarah ke solusi sehingga ruang pencarian yang ditelusuri menjadi jauh lebih kecil [2].

Sebuah persoalan yang diselesaikan dengan backtracking memiliki tiga komponen utama. Pertama, solusi dinyatakan sebagai vektor X = (x1, x2, …., xn) dengan setiap xi merupakan anggota suatu himpunan nilai. Kedua, terdapat fungsi pembatas (bounding function) B(x1,…,xk) yang bernilai benar apabila vektor parsial tersebut masih berpeluang mengarah ke solusi, yakni belum melanggar kendala persoalan. Apabila fungsi pembatas bernilai salah, simpul yang bersangkutan dimatikan sehingga seluruh anaknya ikut terpangkas.

Seluruh kemungkinan solusi diorganisasikan dalam sebuah pohon ruang status, dengan setiap simpul menyatakan status persoalan dan setiap lintasan dari akar ke daun menyatakan satu kemungkinan solusi. Pembangkitan simpul mengikuti aturan DFS (depth-first-search) dengan simpul yang sedang diperluas disebut simpul-E, dan apabila lintasan yang sedang dibentuk tidak lagi mengarah ke solusi, pencarian akan mundur (backtrack) ke simpul pada arah di atasnya. Pada kasus terburuk, ketika hampir tidak ada pemangkasan yang terjadi, kompleksitas waktu backtracking mencapai O(p(n).2^n) atau O(q(n).n!) bergantung pada bentuk ruang solusinya [2].

### C. Persoalan Cryptarithm

Cryptarithm adalah teka-teki aritmatika yang setiap digitnya digantikan oleh huruf. Penyelesaian persoalan ini harus memenuhi sejumlah kendala: setiap huruf yang berbeda mewakili satu digit yang berbeda (0-9), huruf yang sama selalu mewakili digit yang sama, huruf yang menempati posisi paling depan suatu bilangan karena tidak boleh bernilai nol, dan operasi aritmatika yang terburuk harus bernilai benar. Karena solusinya merupakan salah satu dari sekian banyak permutasi penetapan digit yang harus memenuhi seluruh kendala secara bersamaan, cryptarithm tergolong persoalan kombinatorika sekaligus *constraint satisfaction problem.*

## III. Analisis dan Perancangan

### A. Pemodelan Persoalan dalam Kerangka Backtracking

Tinjau sebuah persoalan cryparithm berbentuk penjumlahan dua suku, misalnya kata pertama dijumlahkan dengan kata kedua menghasilkan kata ketiga. Misalkan persoalan tersebut memuat sebanyak L huruf yang berbeda. Solusi dinyatakan sebagai vektor X = (x1, x2, …, xL) dengan xi adalah digit yang ditetapkan bagi huruf ke-i dan xi $\in$ {0,1,…,9}.

Vektor solusi tersebut harus memenuhi tiga jenis kendala. Pertama, kendala keunikan, setiap huruf yang berbeda harus dipetakan ke digit yang berbeda, sehingga seluruh nilai xi tidak ada yang sama. Kedua, kendala digit paling depan, huruf yang menempati posisi paling kiri pada salah satu bilangan tidak boleh bernilai nol. Ketiga, kendala aritmatika, setelah seluruh huruf disubstitusi dengan digitnya, operasi penjumlahan yang terbentuk harus bernilai benar.

Kendala aritmatika ini dapat dinyatakan secara berkolom. Apabila penjumlahan diproses dari kolom paling kanan (satuan) menuju kolom paling kiri, maka pada setiap kolom ke-j berlaku hubungan:

(digit suku pertama)j + (digit suku kedua)j + cj = (digit hasil)j + 10-cj+1

dengan cj menyatakan nilai *carry* yang masuk ke kolom ke-j dan cj+1 adalah *carry* yang keluar dari kolom tersebut, keduanya bernilai 0 atau 1. Carry pada kolom paling kanan bernilai nol.

Mengikuti kerangka formal backtracking, fungsi pembangkit T() berfungsi menetapkan sebuah digit yang belum terpakai kepada huruf berikutnya, sedangkan fungsi pembatas B(x1, …, xk) memeriksa apakah penetapan parsial sejauh ini masih berpeluang mengarah ke solusi, yakni belum melanggar ketiga jenis kendala di atas. Perbedaan utama antara ketiga strategi yang dibandingkan terletak pada seberapa dini dan seberapa ketat fungsi pembatas ini diterapkan.

### B. Strategi 1: Brute Force

Pada strategi brute force, seluruh kemungkinan penetapan digit dibangkitkan tanpa mempertimbangkan kendala apa pun selama proses, dan kebenaran solusi baru diperiksa setelah semua huruf memperoleh nilai. Karena setiap huruf dapat

menerima salah satu dari sepuluh digit secara bebas, terdapat 10^L kemungkinan yang harus dievaluasi. Setiap kemungkinan kemudian diuji terhadap kendala keunikan, kendala digit paling depan, dan kendala aritmatika sekaligus. Strategi ini menjadi baseline yang merepresentasikan pencarian tanpa pemangkasan sama sekali, sejalan dengan karakteristik exhaustive search.

### C. Strategi 2: Backtracking Dasar

Strategi backtracking dasar membangun vektor solusi secara bertahap, satu huruf pada satu waktu. Pemangkasan dilakukan hanya berdasarkan dua kendala struktural, yaitu keunikan dan digit paling depan. Kendala keunikan diterapkan langsung pada fungsi pembangkit dengan hanya membangkitkan digit yang belum dipakai oleh huruf-huruf sebelumnya, sehingga seluruh cabang yang mengandung digit berulang tidak pernah dibangkitkan. Digit paling depan diperiksa dengan menolak penetapan nilai nol. Akan tetapi, kebenaran kendala aritmatika baru diperiksa pada simpul daun, yakni ketika seluruh huruf telah terisi. Akibatnya, jumlah lintasan lengkap yang terbentuk menyusut dari 10^L menjadi sebanyak permutasi P(10, L) = 10! / (10 – L)!.

### D. Strategi 3: Backtracking dengan Pemangkasan Kolom

Strategi ini memperketat fungsi pembatas dengan turut memeriksa kendala aritmatika secepat mungkin. Huruf-huruf ditetapkan dengan urutan yang mengikuti susunan kolom penjumlahan dari kanan ke kiri, sehingga begitu seluruh huruf penyusunan sebuah kolom telah memperoleh nilai, fungsi pembatas langsung memeriksa apakah hubungan aritmatika pada kolom tersebut dapat dipenuhi oleh suau nilai carry yang benar. Apabila kolom tersebut tidak konsisten, simpul langsung dimatikan dan seluruh penetapan huruf di kolom-kolom berikutnya, yang seharusnya membentuk subpohon besar, ikut terpangkas tanpa perlu dibangkitkan.

Dengan demikian, fungsi pembatas pada strategi ini menggabungkan ketiga kendala, yaitu keunikan, digit paling depan, dan konsistensi aritmatika per kolom. Karena pelanggaran aritmatika sering kali sudah dapat dideteksi setelah hanya beberapa huruf ditetapkan Sebagian besar ruang pencarian terbuang jauh sebelum mencapai daun.

### E. Ilustrasi pada contoh Persoalan

Sebagai ilustrasi, untuk persoalan SEND + MORE = MONEY yang memiliki delapan huruf berbeda (S, E, N, D, M, O, R, Y). Penjumlahan berkolomnya menghasilkan rangkaian hubungan berikut, dari kolom satuan hingga kolom paling kiri:

D + E = Y + 10.c1
N + R + c1 = E + 10.c2
E + O + c2 = N + 10.c3
S + M + c3 = O + 10.c4
c4 = M

Pada strategi brute force, delapan huruf tersebut akan disubstitusi satu per satu dengan seluruh kombinasi yang mungkin sebelum persamaan diperiksa. Pada backtracking dasar, hanya kombinasi dengan digit unik yang dibangkitkan, namun verifikasi kelima persamaan kolom tetap ditunda hingga seluruh huruf terisi. Sedangkan untuk backtracking dengan pemangkasan kolom, persamaan kolom satuan sudah dapat diperiksa segera setelah D, E, dan Y ditetapkan, jika pasangan nilai tersebut tidak konsisten, seluruh penetapan untuk lima huruf sisanya tidak pernah dibangkitkan.

## IV. Implementasi dan Pengujian

### A. Lingkungan dan Metodologi Pengujian

Ketiga strategi diimplementasikan dalam bahasa Python dan dijalankan secara berurutan pada lingkungan perangkat keras yang sama, sehingga perbandingan waktu eksekusi di antara ketiganya bersifat adil. Seluruh pengujian dijalankan dalam satu utas (single-thread) tanpa pemanfaatan paralelisme, agar yang terukur murni efektivitas algoritma dan bukan kemampuan perangkat keras.

Kumpulan persoalan uji disusun dengan Tingkat kesulitan yang semakin meningkat, dihitung dari jumlah huruf berbeda L yang terlibat dalam persoalan. Persoalan uji yang dipilih merupakan persoalan cryptarithm penjumlahan dua suku, mulai dari persoalan kecil dengan tiga huruf hingga persoalan berukuran Sembilan huruf. Pemilihan rentang L ini bertujuan memperlihatkan bagaimana kinerja masing-masing strategi berubah seiring membesarnya ruang pencarian. Untuk setiap persoalan, program dirancang mencari seluruh solusi yang ada, bukan berhenti pada Solusi pertama, agar jumlah simpul yang tercatat bersifat deterministik dan tidak bergantung pada urutan penemuan solusi.

Dua metrik dicatat pada setiap pengujian. Metrik pertama adalah jumlah simpul yang dibangkitkan, yaitu banyaknya penetapan digit-ke-huruf yang dicoba selama penelusuran pohon ruang status. Metrik ini dihitung secara identik pada ketiga strategi, yakni setiap kali sebuah digit ditetapkan kepada sebuah huruf, sehingga ketiganya dapat dibandingkan dalam satuan yang sama. Keunggulan metrik ini adalah sifatnya yang deterministic dan tidak bergantung pada perangkat keras, sehingga hasilnya dapat direproduksi secara persis di mesin mana pun. Metrik kedua adalah waktu eksekusi dalam milidetik. Berbeda dengan jumlah simpul, waktu eksekusi bergantung pada perangkat keras dan kondisi sistem, sehingga di sini hanya digunakan untuk memperlihatkan kecenderungan relative antar strate

### B. Implementasi Ketiga Strategi

Ketiga strategi diimplementasikan di atas kerangka penelusuran depth-first yang sama. Vektor solusi dibangun huruf demi huruf mengikuti urutan tertentu, dan setiap huruf ditetapkan dengan sebuah digit yang belum terpakai sehingga kendala keunikan terpenuhi secara otomatis. Perbedaan mendasar antar strategi hanya terletak pada kapan dan seberapa ketat kendala diperiksa, yakni pada bentuk fungsi pembatasnya.

Strategi pertama, brute force, tidak melakukan pemangkasan apa pun selama penelusuran. Seluruh kombinasi penetapan digit dibangkitkan hingga simpul daun, dan kebenaran solusi, meliputi kendala digit paling depan serta kendala aritmatika, baru diperiksa setelah seluruh huruf memiliki nilai. Pemeriksaan persamaan dilakukan secara efisien melalui koefisien yang telah dihitung sebelumnya untuk setiap huruf.

```python
def solve_brute(meta):
    letters, leading, m, cols, coeff, order = meta
    L = len(order)
    assign = {}
    used = [False] * 10
    nodes = [0]
    sols = []

    def rec(i):
        if i == L:
            if any(assign[ch] == 0 for ch in leading):
                return
            if sum(coeff[ch] * assign[ch] for ch in order) == 0:
                sols.append(dict(assign))
            return
        ch = order[i]
        for d in range(10):
            if used[d]:
                continue
            assign[ch] = d
            used[d] = True
            nodes[0] += 1
            rec(i + 1)
            used[d] = False
        assign.pop(ch, None)

    rec(0)
    return sols, nodes[0]
```

Gambar 1. Implementasi strategi brute force

Strategi kedua, yaitu backtracking dasar, menambahkan satu bentuk pemangkasan, yaitu kendala digit paling depan. Begitu sebuah huruf yang menempati posisi paling signifikan hendak diisi dengan nilai nol, cabang tersebut langsung ditolak tanpa diteruskan, sehingga seluruh subpohon di bawahnya tidak pernah dibangkitkan. Akan tetapi, kendala aritmatika tetap baru diperiksa pada simpul daun.

```python
def solve_bt_basic(meta):
    letters, leading, m, cols, coeff, order = meta
    L = len(order)
    assign = {}
    used = [False] * 10
    nodes = [0]
    sols = []

    def rec(i):
        if i == L:
            if sum(coeff[ch] * assign[ch] for ch in order) == 0:
                sols.append(dict(assign))
            return
        ch = order[i]
        is_lead = ch in leading
        for d in range(10):
            if used[d]:
                continue
            if d == 0 and is_lead:# pangkas: huruf terdepan != 0
                continue
            assign[ch] = d
            used[d] = True
            nodes[0] += 1
            rec(i + 1)
            used[d] = False
        assign.pop(ch, None)

    rec(0)
    return sols, nodes[0]
```

Gambar 2. Implementasi strategi backtracking dasar

Strategi ketiga, yaitu backtracking dengan pemangkasan kolom, memperketat fungsi pembatas dengan turut memeriksa kendala aritmatika seawal mungkin melalui fungsi promising. Fungsi ini menelusuri kolom-kolom penjumlaha dari arah kanan ke kiri, selama seluruh huruf penyusunan sebuah kolom telah memiliki nilai, konsistensi penjumlahan beserta nilai carry nya langsung diperiksa, dan penelusuran berhenti pada kolom pertama yang belum lengkap karena carry untuk kolom berikutnya belum dapat ditentukan. Apabila terdapat kolom yang aritmatikanya dilanggar, simpul langsung dimatikan sehingga seluruh penetapan huruf pada kolom-kolom berikutnya ikut terpangkas.

```python
def solve_bt_col(meta):
    letters, leading, m, cols, coeff, order = meta
    L = len(order)
    assign = {}
    used = [False] * 10
    nodes = [0]
    sols = []

    def promising():
        carry = 0
        for (a, b, c) in cols:
            needed = [x for x in (a, b, c) if x is not None]
            if not all(x in assign for x in needed):
                return True          # kolom belum lengkap, berhenti memeriksa
            s = carry + (assign[a] if a else 0) + (assign[b] if b else 0)
            if assign[c] != s % 10:  # aritmetika kolom dilanggar
                return False
            carry = s // 10
        return carry == 0            # seluruh kolom lengkap: carry akhir harus 0

    def rec(i):
        if i == L:
            sols.append(dict(assign))
            return
        ch = order[i]
        is_lead = ch in leading
        for d in range(10):
            if used[d]:
                continue
            if d == 0 and is_lead:
                continue
            assign[ch] = d
            used[d] = True
            nodes[0] += 1
            if promising():
                rec(i + 1)
            used[d] = False
        assign.pop(ch, None)

    rec(0)
    return sols, nodes[0]
```

Gambar 3. Implementasi strategi backtracking dengan pemangkasan kolom

C. Hasil Pengujian

| Puzzle | L | BF nodes | BF ms | BT nodes | BT ms | COL nodes | COL ms | sol |
|---|---|---|---|---|---|---|---|---|
| SO + SO = TOO | 3 | 820 | 0.6 | 667 | 0.3 | 27 | 0.04 | n=1 |
| AS + A = MOM | 4 | 5,860 | 3.5 | 4,689 | 2.1 | 1,105 | 0.98 | n=1 |
| ODD + ODD = EVEN | 5 | 36,100 | 25.1 | 28,972 | 17.0 | 339 | 0.69 | n=2 |
| TWO + TWO = FOUR | 6 | 187,300 | 143.4 | 154,036 | 90.3 | 1,084 | 1.51 | n=7 |
| SEND + MORE = MONEY | 8 | 2,606,500 | 1874.2 | 2,183,140 | 1275.3 | 7,956 | 11.97 | n=1 |
| CROSS + ROADS = DANGER | 9 | 6,235,300 | 3839.6 | 4,625,370 | 2545.8 | 8,681 | 14.60 | n=1 |

Gambar 4. Hasil perbandingan ketiga strategi

Gambar 4 merangkum hasil pengujian ketiga strategi terhadap keenam persoalan uji. Kolom “sol” menyatakan banyaknya solusi yang ditemukan, sedangkan tiap pasangan nilai pada kolom strategi menyatakan jumlah simpul yang dibangkitkan dan waktu eksekusi dalam milidetik.

Dari gambar 4 terlihat bahwa untuk seluruh persoalan, strategi backtracking dengan pemangkasan secara konsisten membangkitkan simpul paling sedikit dan menyelesaikan

persoalan paling cepat, sedangkan brute force selalu menjadi yang paling boros. Selisih ini relatif kecil pada persoalan berukuran tiga hingga empat huruf, tetapi membesar dengan sangat tajam pada persoalan berukuran besar. Sebagai gambaran, pada persoalan SEND + MORE = MONEY, brute force membangkitkan lebih dari 2,6 juta simpul, sementara backtracking dengan pemangkasan kolom hanya membangkitkan kurang dari delapan ribu simpul.

### D. Pembahasan

Hasil pengujian memperlihatkan beberapa hal. Pertama, jumlah simpul pada brute force dan backtracking dasar tumbuh secara faktorial seiring bertambahnya jumlah huruf. Pada gambar 4, jumlah simpul kedua strategi tersebut selalu berdekatan untuk setiap persoalan, yang menunjukkan bahwa pemangkasan kendala digit paling depan hanya memberikan penghematan yang kecil. Penghematan tersebut berkisar sekitar 16% pada persoalan SEND + MORE = MONEY dan sekitar 26% pada CROSS + ROADS = DANGER. Hal ini dapat dipahami karena kendala digit terdepan hanya memangkas sebagian kecil cabang yang terletak dekat dengan akar pohon ruang status, sehingga sebagian besar ruang pencarian tetap dieksplorasi.

Kedua, strategi backtracking dengan pemangkasan kolom memberikan penghematan yang berkali-kali lipat lebih besar. Pada persoalan SEND + MORE = MONEY, jumlah simpul menyusut dari sekitar 2,6 juta menjadi hanya 7.956 simpul, yaitu sekitar 328 kali lebih sedikit, dengan waktu eksekusi sekitar 157 kali lebih cepat. Penghematan yang lebih dramatis terjadi pada persoalan terbesar, CROSS + ROADS = DANGER, yang mengalami pengurangan jumlah simpul sekitar 718 kali dan percepatan waktu sekitar 263 kali. Penyebabnya adalah pelanggaran aritmatika pada cryparithm sering kali sudah dapat dideteksi setelah hanya beberapa huruf pertama ditetapkan, terutapa pada kolom satuan. Begitu sebuah kolom terbukti tidak konsisten, subpohon besar yang merepresentasikan seluruh kemungkinan penetapan huruf sisanya langsung terpangkas tanpa perlu dibangkitkan sama sekali.

Ketiga, keunggulan pemangkasan kolom semakin melebar seiring membesarnya ukuran persoalan. Sebagaimana yang ada pada gambar 4, jumlah simpul pada strategi backtracking dengan pemangkasan kolom jauh lebih lambat dibandingkan kedua strategi lainnya, sehingga selisihnya menjadi semakin besar pada nilai L yang tinggi. Perlu dicatat pula bahwa jumlah simpul pada strategi ini tidak sepenuhnya monoton terhadap L, sebagai contoh, persoalan ODD + ODD = EVEN dengan lima huruf justru hanya membangkitkan 339 simpul, lebih sedikit daripada AS + A = MOM dengan empat huurf. Hal ini menunjukkan bahwa setelah pemangkasan yang kuat diterapkan, jumlah simpul yang dibangkitkan tidak lagi ditentukan semata-mata oleh jumlah huruf, melainkan juga oleh seberapa ketat struktur kendala persoalan tersebut dalam memangkas cabang sejak dini.

Secara keseluruhan, hasil eksperimen menegaskan bahwa efektivitas algoritma backtracking tidak ditentukan oleh strukturnya semata, melainkan diutamakan oleh kualitas fungsi pembatas yang diterapkan. Meskipun fungi pembatas pada strategi ketiga membutuhkan komputasi tambahan pada setiap simpul untuk memeriksa konsistensi kolom, biaya tambahan tersebut terbukti jauh lebih kecil dibandingkan penghematan dari pemangkasan yang dihasilkannya. Fugnsi pembatas yang memanfaatkan struktur aritmatika persoalan berhasil mengubah pencarian yang semula praktis tidak mangkus menjadi sangat efisien.

## V. KESIMPULAN

Makalah ini telah memodelkan persoalan cryparithm ke dalam kerangka formal algoritma backtracking, yaitu melalui penetapan vektor solusi berupa pemetaan huruf-ke-digit, fungsi pembangkit yang menetapkan digit secara bertahap, serta fungsi pembatas yang memeriksa pemenuhan kendala keunikan, digit paling depan, dan aritmatika. Berdasarkan pemodelan tersebut, tiga strategi telah diimplementasikan dan dibandingkan, yakni brute force, backtracking dasar, serta backtracking dengan pemangkasan aritmatika per kolom.

Hasil eksperimen menunjukan bahwa pemangkasan kendala digit paling depan hanya memberikan penghematan kecil, yaitu sekitar 16% hingga 26% terhadap jumlah simpul brute force, karena hanya memangkas sebagian cabang di dekat akar pohon ruang status. Sebaliknya, pemangkasan aritmatika per kolom memberikan penghematan yang sangat besar, mencapai sekitar 328 kali lebih sedikit simpul pada persoalan SEND + MORE = MONEY dan sekitar 718 kali pada persoalan CROSS + ROADS = DANGER, disertai percepatan waktu eksekusi yang sebanding. Keunggulan ini juga semakin melebar seiring bertambahnya jumlah huruf yang terlibat. Dengan demikian, dapat disimpulkan bahwa efektivitas algoritma backtracking dalam menyelesaikan persoalan cryptarithm tidak ditentukan oleh strukturnya semata, melainkan diutamakan oleh kualitas fungsi pembatas yang memanfaatkan struktur persoalan untuk memangkas ruang pencarian seawal mungkin.

Sebagai pengembangan lebih lanjut, penelitian ini dapat diperluas dengan menambahkan strategi pemilihan huruf yang lebih cerdas, misalnya mendahulukan huruf yang paling membatasi, serta dengan memperluas cakupan persoalan ke penjumlahan lebih dari dua suku, operasi selain penjumlahan, atau basis bilangan selain sepuluh, Selain itu, kinerja algoritma backtracking dapat dibandingkan dengan teknikpenyelesaian *constraint satisfaction problem* lain seperti propagasi kendala penuh untuk memperoleh gambaran yang lebih menyeluruh.

## VI. UCAPAN TERIMA KASIH

Puji Syukur serta rasa terima kasih selaku penulis kepada Allah SWT atas segala berkat, rahmat, dan karunia-Nya sehingga penulis dapat menyelesaikan makalah ini dengan lancar. Selain itu penulis juga berterima kasih kepada :

1. Orang tua penulis, yang selalu mendukung, dan mendoakan penulis.

2. Ibu Tricya Esterina Widagdo, S.T, M.Sc atas bimbingannya dalam mengajar mata kuliah Strategi Algoritma, khususnya dalam materi backtracking

3. Bapak Dr. Ir. Rinaldi Munir, MT., yang telah memberikan sumber materi pembelajaran.

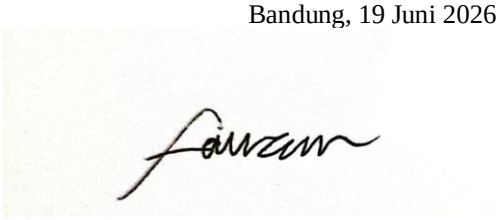
## DAFTAR PUSTAKA

[1] R. Munir, "Algoritma Brute Force (Bagian 1 dan 2)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2026.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/02-Algoritma-Brute-Force-(2026)-Bag1.pdf
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/03-Algoritma-Brute-Force-(2026)-Bag2.pdf
[2] R. Munir, "Algoritma Backtracking (Bagian 1 dan 2)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2026.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/15-Algoritma-backtracking-(2026)-Bagian1.pdf
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/16-Algoritma-backtracking-(2026)-Bagian2.pdf
[3] A. Levitin, Introduction to the Design and Analysis of Algorithm, 3rd ed. Boston, MA, USA: Pearson, 2012

## 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 2026



Fauzan Mohamad Abdul Ghani 13524113