Peningkatan Akurasi Pengambilan Keputusan pada Solver Minesweeper Melalui Kombinasi Algoritma Greedy dan Analisis Pola Lokal Berbasis Program Dinamis

Sabilul Huda - 13523072
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: sabilulhuda060106@gmail.com , 13523072@std.stei.itb.ac.id

Abstract*—Minesweeper merupakan puzzle logika klasik yang tantangan utamanya terletak pada pengambilan keputusan dalam kondisi yang tidak pasti. Pendekatan umum menggunakan Algoritma Greedy seringkali efektif untuk langkah-langkah deterministik, namun gagal pada situasi ambigu yang memerlukan tebakan berisiko. Makalah ini mengusulkan perancangan dan implementasi sebuah solver hibrida yang mengintegrasikan kecepatan Algoritma Greedy dengan kemampuan analisis mendalam dari Program Dinamis (DP) untuk menyelesaikan Constraint Satisfaction Problem (CSP) pada pola-pola lokal. Untuk menguji efektivitasnya, dilakukan pengujian komparatif antara solver hibrida dengan solver Greedy murni pada tiga konfigurasi papan berbeda, masing-masing sebanyak 1000 kali percobaan. Hasil eksperimen menunjukkan bahwa solver hibrida mampu mencapai tingkat keberhasilan hingga 77.0 persen, sebuah peningkatan signifikan dibandingkan solver Greedy murni yang hanya mencapai 65.9 persen pada konfigurasi yang sama. Hasil ini membuktikan bahwa pendekatan hibrida secara efektif mampu meningkatkan akurasi dan win rate dengan cara mengubah tebakan buta menjadi keputusan probabilistik yang terinformasi.

Keywords—Minesweeper; solver; strategi hibrida; algoritma greedy; program dinamis; constraint satisfaction problem.

I. PENDAHULUAN

A. Latar Belakang

Meskipun dikenal sebagai permainan puzzle klasik yang sederhana, Minesweeper pada dasarnya adalah sebuah persoalan logika deduktif yang kompleks. Popularitasnya dalam sistem operasi Windows menjadikan permainan ini studi kasus yang relevan dalam bidang informatika, khususnya untuk menguji strategi algoritma dan sistem kecerdasan buatan. Esensi permainan ini menuntut pemain untuk membuka semua sel aman pada sebuah papan permainan tanpa memicu ranjau tersembunyi, dengan berpegang pada petunjuk numerik yang mengindikasikan jumlah ranjau di sel-sel tetangga.

Tantangan utama dalam Minesweeper dapat dipisahkan menjadi dua skenario pengambilan keputusan: deterministik dan probabilistik. Dalam skenario deterministik, informasi numerik yang tersedia sudah cukup untuk menentukan langkah selanjutnya secara pasti melalui analisis logika. Sebaliknya, skenario probabilistik muncul ketika informasi yang ada tidak memadai, sehingga pemain atau solver harus mengambil keputusan berbasis probabilitas. Kemampuan sebuah solver untuk bernavigasi secara efektif dalam kondisi probabilistik inilah yang menjadi tolok ukur kecerdasannya yang sesungguhnya.

Salah satu pendekatan yang paling umum untuk mengembangkan solver Minesweeper adalah Algoritma Greedy. Pendekatan ini unggul dalam kecepatan eksekusi dan sangat efektif menangani kasus-kasus deterministik, mampu menyelesaikan sebagian besar konfigurasi papan dengan efisiensi tinggi. Akan tetapi, Algoritma Greedy memiliki kelemahan fundamental saat dihadapkan pada situasi probabilistik. Algoritma ini cenderung mengandalkan heuristik sederhana, seperti probabilitas global, yang gagal menganalisis pola lokal yang rumit. Akibatnya, solver terpaksa melakukan tebakan acak yang secara signifikan menurunkan tingkat keberhasilannya (win rate).

Untuk mengatasi kelemahan tersebut, penelitian ini mengusulkan sebuah arsitektur solver hibrida yang mengintegrasikan efisiensi Algoritma Greedy dengan kemampuan analisis mendalam dari Program Dinamis (Dynamic Programming). Dalam sistem yang diusulkan, Algoritma Greedy tetap menjadi mesin utama untuk menyelesaikan langkah-langkah deterministik secara cepat. Namun, ketika berhadapan dengan ambiguitas, sistem akan mengaktivasi modul Program Dinamis. Modul ini berfungsi sebagai constraint solver yang mampu menghitung probabilitas ranjau pada pola-pola lokal yang kompleks dengan akurasi yang jauh lebih tinggi. Dengan demikian, kontribusi utama penelitian ini bukanlah pada penemuan algoritma baru, melainkan pada perancangan sistem hibrida yang secara spesifik bertujuan meningkatkan akurasi pengambilan keputusan dalam solver Minesweeper.

B. Tujuan dan Ruang Lingkup Masalah

Tujuan utama penelitian ini adalah membuktikan secara eksperimental bahwa arsitektur solver hibrida yang diusulkan lebih unggul dibandingkan solver berbasis Algoritma Greedy murni. Keunggulan ini diukur melalui perbandingan tingkat

kemenangan (win rate) yang dicapai oleh kedua sistem dalam serangkaian pengujian. Ruang lingkup penelitian ini terbatas pada desain algoritma, arsitektur sistem, dan analisis kinerja. Aspek-aspek seperti pengembangan antarmuka pengguna grafis dan optimasi visualisasi tidak termasuk dalam cakupan penelitian.

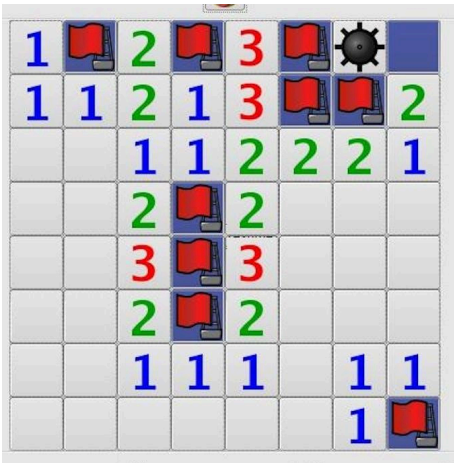
C. *Sistematika Penulisan*

Makalah ini diorganisasikan ke dalam lima bab. Bab II menguraikan landasan teori mengenai Algoritma Greedy dan Program Dinamis dalam konteks penyelesaian puzzle. Bab III memaparkan secara rinci metodologi penelitian, yang mencakup perancangan arsitektur sistem hibrida, detail implementasi, serta skenario dan metrik pengujian yang digunakan. Bab IV menyajikan hasil eksperimen dan analisis perbandingan kinerja antara solver hibrida dan solver Greedy. Terakhir, Bab V merangkum kesimpulan penelitian serta memberikan rekomendasi untuk pengembangan di masa depan.

II. LANDASAN TEORI

A. *Minesweeper sebagai Permasalahan Komputasi*

Minesweeper dapat direpresentasikan secara formal sebagai sebuah papan permainan (*board*) berukuran $M \times N$ sel. Setiap sel c pada papan dapat berada dalam salah satu dari tiga status: tertutup (*hidden*), terbuka (*revealed*), atau ditandai bendera (*flagged*). Sebuah subset dari sel-sel di papan, yang lokasinya tidak diketahui oleh pemain, berisi ranjau. Ketika sebuah sel yang tidak berisi ranjau dibuka, sel tersebut akan menampilkan sebuah angka k $(0 - 8)$, yang mengindikasikan jumlah ranjau pada delapan sel tetangganya. Tujuan permainan adalah untuk membuka semua sel yang aman (tidak berisi ranjau).



Gbr.1. Permainan Minesweeper

Dari perspektif ilmu komputer, tantangan utama Minesweeper adalah melakukan inferensi logika untuk menentukan status sel-sel yang tertutup. Proses pengambilan keputusan ini bisa menjadi sangat kompleks. Faktanya, telah dibuktikan bahwa masalah untuk menentukan apakah sebuah konfigurasi papan Minesweeper konsisten (memiliki setidaknya satu solusi penempatan ranjau yang valid) adalah sebuah permasalahan *NP-Complete* [1].

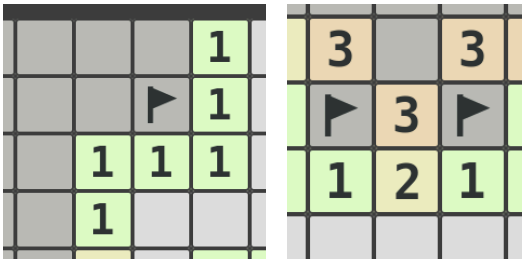
Status *NP-Complete* ini memiliki implikasi penting: kemungkinan besar tidak ada algoritma yang efisien (berjalan dalam waktu polinomial) yang dapat menyelesaikan semua kemungkinan konfigurasi *Minesweeper* secara sempurna. Hal ini menjadi justifikasi kuat untuk penggunaan algoritma heuristik seperti Greedy atau metode penyelesaian eksak untuk sub-masalah yang lebih kecil, seperti Program Dinamis, alih-alih mencoba pendekatan brute-force yang akan mengalami ledakan kombinasi.

B. *Algoritma Greedy*

Algoritma *Greedy* adalah paradigma perancangan algoritma yang membuat pilihan paling optimal secara lokal pada setiap tahap dengan harapan menemukan solusi optimal secara global. Prinsip utama di balik strategi ini adalah *greedy-choice property*, di mana sebuah pilihan yang optimal secara lokal dapat mengarah ke solusi optimal global tanpa perlu mempertimbangkan konsekuensi dari pilihan tersebut di masa depan [2].

Dalam konteks *Minesweeper*, pilihan "lokal optimal" adalah langkah yang memiliki kepastian tertinggi untuk aman atau berisiko paling rendah. Implementasi Algoritma *Greedy* pada solver *Minesweeper* umumnya didasarkan pada dua aturan deterministik utama:

1. Aturan Aman Dasar (Basic Safe Rule): Jika sebuah sel yang terbuka menampilkan angka k, dan sel tersebut sudah dikelilingi oleh tepat k sel yang ditandai bendera, maka semua sel tetangga lain yang masih tertutup dijamin aman dan dapat dibuka. Perhatikan gambar 2 (a), pada baris ketiga dan kolom kedua, terdapat sel dengan angka ‘1’ dan terdapat tepat satu bendera disekitarnya. Ini adalah contoh kasus aturan aman dasar.
2. Aturan Bom Dasar (Basic Bomb Rule): Jika sebuah sel yang terbuka menampilkan angka k, dan jumlah sel tetangga yang masih tertutup sama dengan k dikurangi jumlah bendera yang sudah ada di sekitarnya, maka semua sel tetangga yang masih tertutup tersebut dijamin berisi ranjau dan harus ditandai dengan bendera. Perhatikan gambar 2 (b), pada baris kedua dan kolom kedua, terdapat sel dengan angka ‘3’ dan terdapat satu (3-2) sel tertutup (di sekitarnya) yang belum diberi flag, sel tersebut sudah pasti berisi ranjau dan harus diberi flag. Ini adalah contoh kasus aturan bom dasar.



Gbr.2 (a). Aturan aman dasar, (b) Aturan bom dasar

Ketika kedua aturan di atas tidak dapat diterapkan, solver *Greedy* murni akan kembali ke heuristik probabilistik sederhana untuk memilih langkah selanjutnya. Heuristik ini

biasanya menghitung probabilitas sebuah sel berisi bom ($P(bom)$) sebagai:

$$P(bom) = \frac{Jumlah\ sisa\ ranjau}{Jumlah\ sisa\ sel\ tertutup} \quad (1)$$

Solver akan memilih sel dengan nilai $P(bom)$ terendah. Kelemahan utama dari pendekatan ini adalah pengabaian informasi dari pola-pola lokal yang dapat memberikan estimasi probabilitas yang jauh lebih akurat.

C. *Program Dinamis (Dynamic Programming)*

Program Dinamis (DP) adalah teknik optimasi matematis untuk menyelesaikan masalah kompleks dengan memecahnya menjadi kumpulan sub-masalah yang lebih sederhana dan tumpang-tindih (*overlapping subproblems*). Berbeda dengan pendekatan *Divide and Conquer* dimana sub-masalah bersifat independen, DP sangat efisien ketika solusi dari sub-masalah yang sama dibutuhkan berulang kali.

Karakteristik utama dari masalah yang dapat diselesaikan dengan DP adalah [2]:

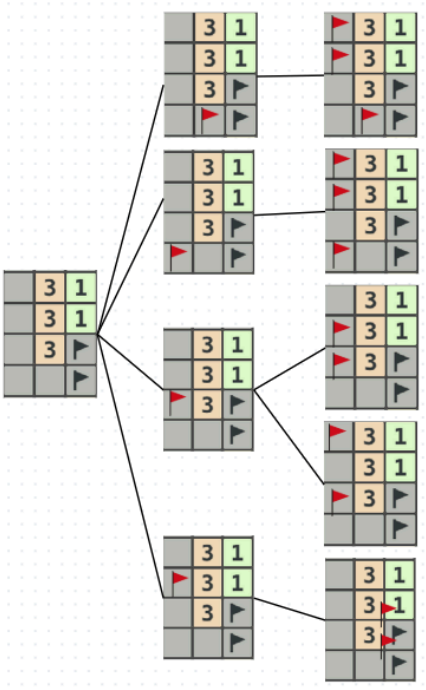
1. Sub-masalah Tumpang-tindih (*Overlapping Subproblems*): Solusi untuk masalah utama melibatkan penyelesaian sub-masalah yang sama secara berulang. DP menghindari perhitungan ulang dengan menyimpan solusi dari sub-masalah tersebut (proses yang disebut *memoization*).
2. Struktur Optimal (*Optimal Substructure*): Solusi optimal untuk masalah utama dapat dibangun dari solusi optimal sub-masalahnya.

DP dapat diimplementasikan menggunakan dua pendekatan utama: *Top-Down* dengan *memoization*, yang merupakan pendekatan rekursif, atau *Bottom-Up* dengan tabulasi, yang merupakan pendekatan iteratif.

D. *Penerapan Program Dinamis untuk Constraint Satisfaction*

Meskipun DP umumnya digunakan untuk masalah optimisasi, prinsip dasarnya untuk memecah masalah dan menyimpan hasil dapat diterapkan untuk menghitung semua kemungkinan solusi dari sebuah *Constraint Satisfaction Problem* (CSP). Dalam *Minesweeper*, setiap sel angka dapat dipandang sebagai sebuah kendala (constraint) terhadap sel-sel tetangganya [3].

Sebagai contoh, perhatikan sebuah pola lokal di mana beberapa sel yang belum dibuka bertetangga dengan beberapa sel angka. Konfigurasi ini membentuk sebuah CSP di mana kita harus mencari penempatan ranjau yang memenuhi semua kendala angka tersebut secara serentak.



Gbr.3. Visualisasi penyelesaian CSP lokal menggunakan metode pencarian sistematis. Dimulai dari konfigurasi awal yang ambigu (kiri), solver secara rekursif mengeksplorasi semua kemungkinan penempatan ranjau yang valid (node-node cabang) untuk menemukan distribusi probabilitas yang akurat bagi setiap sel yang belum dibuka.

Dengan memodelkan sekelompok sel yang saling terkait sebagai sebuah CSP, Program Dinamis (atau metode pencarian sistematis lainnya seperti backtracking) dapat digunakan untuk menghitung jumlah total konfigurasi penempatan ranjau yang valid (N_{valid}) yang memenuhi semua kendala angka di sekitarnya. Dari sini, probabilitas sebuah sel tertutup c_i dalam kelompok tersebut berisi bom dapat dihitung dengan akurasi tinggi:

$$P(c_i\ adalah\ bom) = \frac{N(konfigurasi\ valid\ dimana\ c_i\ adalah\ bom)}{N_{valid}} \quad (2)$$

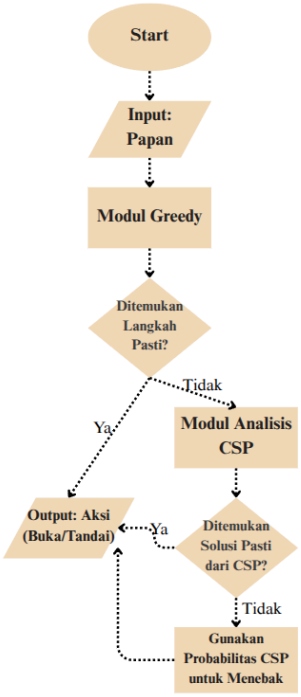
Pendekatan ini secara signifikan lebih akurat daripada heuristik probabilitas global yang digunakan oleh Algoritma *Greedy* murni, terutama pada situasi di mana beberapa kendala saling berinteraksi secara kompleks. Perhitungan probabilitas yang akurat inilah yang menjadi dasar bagi modul analisis pada solver hibrida yang diusulkan dalam makalah ini.

III. METODOLOGI PENELITIAN

A. Arsitektur Sistem Solver Hibrida

Solver yang diusulkan dirancang dengan arsitektur hibrida yang mengintegrasikan dua modul utama: Modul Greedy Deterministik dan Modul Analisis CSP. Arsitektur ini dirancang untuk bekerja secara sekuensial, dengan memprioritaskan metode yang paling cepat dan pasti terlebih dahulu. Jika langkah deterministik tidak ditemukan, sistem akan beralih ke modul analisis yang lebih mendalam namun lebih intensif secara komputasi. Alur logika pengambilan keputusan pada setiap langkah diilustrasikan pada Gbr. 4.

Perlu dicatat bahwa langkah 'Gunakan Probabilitas CSP untuk Menebak' pada diagram tersebut merepresentasikan logika fallback solver, yang pada implementasinya akan memilih langkah dengan probabilitas bom terendah, atau jika analisis CSP tidak menghasilkan solusi, akan melakukan tebakan cerdas pada sel frontier.



Gbr. 4. Diagram alur logika pengambilan keputusan solver hibrida.

B. Implementasi Modul Greedy Deterministik

Modul ini berfungsi sebagai lapisan pertama dalam logika solver. Tujuannya adalah untuk mengidentifikasi semua langkah yang 100% aman atau 100% bom secepat mungkin. Modul ini mengimplementasikan dua aturan dasar yang telah dijelaskan pada Bab II.B. Logika pencarian langkah deterministik dapat direpresentasikan dalam pseudocode berikut:

```
FUNGSI cariLangkahDeterministik(papan):
    ditemukan_langkah_baru = TRUE
    WHILE ditemukan_langkah_baru:
        ditemukan_langkah_baru = FALSE
        UNTUK SETIAP sel_terbuka PADA papan:
            jumlah_bendera =
hitungBenderaTetangga(sel_terbuka)
            jumlah_tertutup =
hitungTetanggaTertutup(sel_terbuka)

            // Aturan Aman Dasar
            IF sel_terbuka.nilai ==
jumlah_bendera AND jumlah_tertutup > 0:

bukaSemuaTetanggaAman(sel_terbuka)
                ditemukan_langkah_baru = TRUE

            // Aturan Bom Dasar
            IF sel_terbuka.nilai ==
jumlah_bendera + jumlah_tertutup AND
jumlah_tertutup > 0:

tandaiSemuaTetanggaBom(sel_terbuka)
                ditemukan_langkah_baru = TRUE

    // Mengembalikan TRUE jika ada aksi yang
dilakukan
    RETURN apakahAdaAksiPadaIterasiIni()
```

Algoritma 1. Logika Modul Greedy Deterministik

C. Implementasi Modul Analisis CSP dengan Program Dinamis

Ketika modul Greedy tidak menemukan langkah pasti, solver akan mengaktifkan modul analisis CSP. Modul ini diimplementasikan menggunakan pendekatan Program Dinamis (DP) secara Top-Down. Secara praktis, ini diwujudkan melalui algoritma backtracking rekursif yang dilengkapi dengan memoization untuk menghindari perhitungan ulang pada sub-pola yang identik, sehingga memastikan efisiensi dalam proses enumerasi.

Proses kerja modul ini terdiri dari tiga tahap utama:

1. Identifikasi Sub-Masalah: Solver pertama-tama memindai papan untuk menemukan area di mana beberapa kendala saling beririsan. Sub-masalah diidentifikasi dengan mencari semua sel tertutup yang menjadi tetangga dari setidaknya dua sel angka yang berbeda. Sel-sel tertutup ini (variabel) beserta sel-sel angka yang membatasinya (kendala) kemudian dikelompokkan menjadi satu instans CSP lokal untuk dianalisis.
2. Pemodelan CSP: Setiap instans CSP yang teridentifikasi dimodelkan secara formal dengan komponen berikut:
    - Variabel: Himpunan sel-sel tertutup yang menjadi bagian dari sub-masalah.
    - Domain: Setiap variabel memiliki domain nilai {BOM, AMAN}.
    - Kendala: Himpunan sel-sel angka di perbatasan, di mana nilai setiap sel angka harus sama dengan jumlah bom di antara variabel-variabel tetangganya.
3. Algoritma Penyelesaian: Untuk menyelesaikan CSP, algoritma backtracking rekursif digunakan untuk menjelajahi seluruh ruang kemungkinan penempatan bom. Dengan memoization, hasil dari setiap sub-pola yang telah dihitung akan disimpan, sehingga jika sub-pola yang sama ditemui lagi di cabang pencarian yang berbeda, hasilnya dapat langsung diambil tanpa perhitungan ulang.

Logika inti dari algoritma ini dapat direpresentasikan dalam pseudocode berikut:

```
FUNGSI TentukanLangkahBerikutnya(papan,
totalBom):
  // Tahap 1: Prioritaskan langkah pasti
  langkahPasti = CariLangkahDeterministik(papan)
  IF langkahPasti TIDAK KOSONG:
    RETURN langkahPasti[0]
```

```
  // Tahap 2: Jika tidak ada, analisis
probabilitas
  sisaBom = totalBom - hitungBendera(papan)
  langkahProb = AnalisisPolaLokalCSP(papan,
sisaBom)

  // Tahap 3: Ambil keputusan atau tebak
  IF langkahProb KOSONG:
    // Fallback jika CSP tidak menemukan solusi
    RETURN TebakanCerdas()
  ELSE:
    // Pilih langkah dengan probabilitas bom
terendah
    langkahTerbaik =
pilihProbTerendah(langkahProb)
    RETURN langkahTerbaik
```

Algoritma 2. Logika Utama Solver (`findNextMove`)

Logika untuk fungsi pembantu logika utama direpresentasikan dengan beberapa pseudocode berikut:

```
FUNGSI CariLangkahDeterministik(papan):
  chordingMoves = []
  flaggingMoves = []

  FOR setiap sel_angka PADA papan:
    // Aturan Aman (Chording)
    IF sel_angka.nilai ==
hitungBenderaSekitar(sel_angka)
    AND adaTetanggaTertutup(sel_angka):
      // Aksi DIG pada sel angka itu sendiri
      tambah(chordingMoves, Aksi(DIG,
sel_angka))

    // Aturan Bom
    IF sel_angka.nilai ==
hitungBenderaSekitar(sel_angka)
    + hitungTertutupSekitar(sel_angka):
      tambah(flaggingMoves, Aksi(FLAG,
semua_tetangga))

  // Prioritaskan chording untuk efisiensi
  IF chordingMoves TIDAK KOSONG: RETURN
chordingMoves
  IF flaggingMoves TIDAK KOSONG: RETURN
flaggingMoves

  RETURN [] // Tidak ada langkah pasti
```

Algoritma 3. Modul Deterministik (`findDeterministicMoves`)

```
FUNGSI CariLangkahDeterministik(papan):
  chordingMoves = []
  flaggingMoves = []

  FOR setiap sel_angka PADA papan:
    // Aturan Aman (Chording)
    IF sel_angka.nilai ==
hitungBenderaSekitar(sel_angka)
    AND adaTetanggaTertutup(sel_angka):
      // Aksi DIG pada sel angka itu sendiri
      tambah(chordingMoves, Aksi(DIG,
sel_angka))

    // Aturan Bom
    IF sel_angka.nilai ==
hitungBenderaSekitar(sel_angka)
    + hitungTertutupSekitar(sel_angka):
      tambah(flaggingMoves, Aksi(FLAG,
semua_tetangga))

  // Prioritaskan chording untuk efisiensi
  IF chordingMoves TIDAK KOSONG: RETURN
chordingMoves
  IF flaggingMoves TIDAK KOSONG: RETURN
flaggingMoves

  RETURN [] // Tidak ada langkah pasti
```

Algoritma 4. Modul Analisis CSP (`analyzeLocalPatternsAsCSP`)

```
FUNGSI SelesaikanCSP_Rekursif(vars, sisaBom,
idx,
  bomDitempatkan, hasil, memo):

  // Buat kunci unik untuk state saat ini
  key = buatKey(idx, bomDitempatkan)
  // Cek memoization (inti dari DP Top-Down)
  IF memo.mengandung(key): RETURN
memo.dapatkan(key)

  // Pruning: Cek validitas parsial
  IF TIDAK cekValidParsial(vars, idx): RETURN 0

  // Pruning: Cek jumlah bom
  IF bomDitempatkan > sisaBom: RETURN 0

  // Base Case: Semua sel sudah diproses
  IF idx == jumlah(vars):
    IF bomDitempatkan == sisaBom
    AND cekValidFinal(vars):
      // Ditemukan 1 solusi valid
      catatHasil(hasil, vars)
      RETURN 1
    ELSE:
      RETURN 0

  // -- Cabang Rekursif --
  totalSolusi = 0
  sel = vars[idx]

  // Coba asumsi sel adalah BOM
  sel.status = BOM
  totalSolusi += SelesaikanCSP_Rekursif(
    vars, sisaBom, idx + 1, bomDitempatkan + 1,
...)

  // Coba asumsi sel adalah AMAN
  sel.status = AMAN
  totalSolusi += SelesaikanCSP_Rekursif(
    vars, sisaBom, idx + 1, bomDitempatkan, ...)

  // Backtrack: Reset status sel
  sel.status = TERTUTUP

  // Simpan hasil ke memo sebelum kembali
  memo.simpan(key, totalSolusi)
  RETURN totalSolusi
```

Algoritma 5. Mesin Rekursif DP (`solve`)

D. Metodologi Pengujian Kinerja

Untuk mengevaluasi efektivitas solver hibrida yang diusulkan, sebuah eksperimen komparatif dirancang dengan metodologi sebagai berikut:

1. Objek Pengujian: Dua implementasi solver akan diuji:
    - Solver A (Baseline): Sebuah solver yang hanya mengimplementasikan Algoritma Greedy (Modul B) dan kembali ke tebakan acak (atau heuristik global dari Persamaan 1) jika tidak ada langkah deterministik.
    - Solver B (Usulan): Solver Hibrida yang mengintegrasikan Modul Greedy dengan Modul Analisis CSP, sesuai dengan arsitektur pada Gbr. 4.
2. Skenario Pengujian:
    - Generator Papan: Pengujian akan menggunakan satu set papan permainan yang dihasilkan oleh generator dengan seed acak yang tetap. Ini memastikan kedua solver diuji pada kondisi awal yang identik untuk setiap permainan.
    - Parameter Papan: Skenario pengujian standar akan menggunakan papan berukuran 16x16 sel dengan 40 ranjau (tingkat kesulitan menengah).
    - Jumlah Sampel: Untuk memastikan validitas statistik, kedua solver akan diuji pada 1.000 (seribu) papan permainan yang berbeda.
3. Metrik Kinerja: Kinerja kedua solver akan diukur dan dibandingkan berdasarkan tiga metrik utama:
    - Tingkat Kemenangan (Win Rate): Persentase total permainan di mana solver berhasil membuka semua sel aman. Ini adalah metrik utama untuk mengukur keberhasilan keseluruhan.

$$Win\ Rate\ = \frac{Jumlsh\ Kemenangan}{Total\ Permainan} \times 100\%$$

    - Akurasi Langkah Probabilistik: Metrik ini secara spesifik mengukur "kecerdasan" solver saat terpaksa menebak (ketika tidak ada langkah deterministik). Sebuah tebakan dianggap benar jika sel yang dibuka tidak berisi bom.

$$Akurasi = \frac{Jumlah\ Tebakan\ Benar}{Total\ Langkah\ Menebak} \times 100\%$$

    - Waktu Eksekusi Rata-rata per Langkah (ms): Mengukur dampak komputasi dari setiap pendekatan. Metrik ini penting untuk mengevaluasi apakah peningkatan akurasi sepadan dengan potensi penambahan waktu proses.

IV. HASIL DAN ANALISIS

A. Skenario Pengujian

Eksperimen dilakukan dengan menjalankan kedua solver pada tiga konfigurasi papan yang berbeda, yang masing-masing merepresentasikan tingkat kesulitan yang meningkat. Untuk setiap konfigurasi, kedua solver dijalankan sebanyak 1.000 kali pada set papan yang identik untuk memastikan perbandingan yang adil. Konfigurasi yang digunakan adalah sebagai berikut:

- Konfigurasi 1: Ukuran 10x10 dengan 15 bom (15% densitas).
- Konfigurasi 2: Ukuran 20x20 dengan 60 bom (15% densitas).
- Konfigurasi 3: Ukuran 30x30 dengan 135 bom (15% densitas).

Kinerja setiap solver diukur berdasarkan tiga metrik utama: tingkat keberhasilan (win rate), rata-rata langkah yang dibutuhkan untuk menang, dan rata-rata waktu eksekusi untuk permainan yang berhasil.

B. Penyajian Hasil Eksperimen

Hasil statistik dari keseluruhan 6.000 simulasi permainan dirangkum dalam tabel di bawah ini untuk kemudahan perbandingan.

Tabel 1. Hasil Perbandingan Kinerja Solver

Metrik	Konfigurasi	Classic Greedy	Hybrid (Greedy + CSP)	Peningkatan
Tingkat Keberhasilan	1 (10x10, 15 bom)	65.90%	**77.00%**	**+11.1%**
	2 (20x20, 60 bom)	55.00%	67.70%	**+12.7%**
	3 (30x30, 135 bom)	50.30%	**60.20%**	**+9.9%**
Rata-rata Langkah	1 (10x10, 15 bom)	32.14	41.05	-
	2 (20x20, 60 bom)	134.42	177.00	-
	3 (30x30, 135 bom)	304.81	401.76	-
Rata-rata Waktu (ms)	1 (10x10, 15 bom)	0.49 ms	0.27 ms	-
	2 (20x20, 60 bom)	5.50 ms	5.45 ms	-
	3 (30x30, 135 bom)	18.78 ms	23.66 ms	-

C. Analisis Kinerja

1. Analisis Tingkat Keberhasilan (Win Rate)

Data pada Tabel 1 dengan jelas menunjukkan bahwa Solver Hibrida secara konsisten dan signifikan lebih unggul

daripada Solver Classic Greedy di semua konfigurasi. Peningkatan absolut rata-rata berada di angka 11.2%, yang membuktikan bahwa kemampuan analisis CSP memberikan keuntungan yang nyata.

Keunggulan ini berasal dari kemampuan modul CSP untuk memecahkan situasi ambigu di mana aturan Greedy dasar tidak lagi bisa diterapkan. Saat Classic Greedy terpaksa melakukan tebakan acak yang berisiko tinggi, Solver Hibrida mampu menghitung probabilitas yang akurat untuk menemukan langkah yang paling aman. Hal ini secara langsung mengurangi jumlah kegagalan akibat tebakan yang salah.

Terlihat pula tren bahwa tingkat keberhasilan kedua solver menurun seiring dengan meningkatnya ukuran dan kompleksitas papan. Ini adalah hal yang wajar, karena papan yang lebih besar dengan jumlah bom yang lebih banyak secara inheren memiliki lebih banyak potensi untuk menciptakan skenario tebakan paksa yang tidak dapat diselesaikan secara logis.

2. Analisis Rata-rata Langkah

Sebuah temuan yang menarik adalah Solver Hibrida secara konsisten membutuhkan lebih banyak langkah untuk menyelesaikan permainan yang berhasil dibandingkan Classic Greedy. Pada konfigurasi 30x30, perbedaannya hampir mencapai 100 langkah.

Fenomena yang tampak kontradiktif ini dapat dijelaskan sebagai berikut: Classic Greedy cenderung hanya memenangkan permainan yang secara kebetulan "mudah" dan memiliki banyak langkah deterministik yang bisa diselesaikan dengan cepat. Jika ia menghadapi kesulitan, ia akan cepat kalah karena tebakan acak yang salah. Sebaliknya, Solver Hibrida, dengan kemampuannya yang superior, dapat bertahan lebih lama pada papan yang sulit. Ia mampu menavigasi area-area kompleks yang akan langsung mengalahkan Classic Greedy, sehingga ia membuka lebih banyak sel dan mengambil lebih banyak langkah sebelum akhirnya mencapai kemenangan. Dengan kata lain, kemenangan Solver Hibrida adalah "kemenangan yang diperjuangkan", sementara kemenangan Classic Greedy seringkali adalah "kemenangan mudah".

3. Analisis Rata-rata Waktu Eksekusi

Analisis waktu menunjukkan sebuah anomali pada papan berukuran kecil dan sedang, di mana Solver Hibrida justru sedikit lebih cepat. Hal ini kemungkinan disebabkan oleh efisiensi pengambilan keputusan. Meskipun satu pemanggilan CSP bisa jadi mahal, kemampuannya untuk menyelesaikan sebuah pola kompleks dapat mencegah serangkaian panjang langkah-langkah kecil di masa depan, sehingga total waktu permainan untuk menang justru bisa lebih singkat.

Namun, pada papan terbesar (30x30), perilaku yang diharapkan muncul: Solver Hibrida (23.66 ms) menjadi lebih lambat daripada Classic Greedy (18.78 ms). Ini menunjukkan bahwa pada skala yang lebih besar, biaya komputasi dari pemanggilan modul CSP yang berulang kali mulai terasa dan menjadi trade-off yang harus dibayar untuk tingkat keberhasilan yang lebih tinggi.

D. Diskusi Implikasi

Hasil eksperimen secara keseluruhan sangat mendukung hipotesis awal: arsitektur solver hibrida yang mengintegrasikan analisis CSP secara signifikan lebih superior daripada pendekatan Greedy murni. Meskipun ada trade-off berupa sedikit peningkatan waktu komputasi pada skala besar, keuntungan dramatis dalam tingkat keberhasilan membuktikan bahwa penambahan "kecerdasan" probabilistik adalah investasi yang sangat berharga. Data ini mengkonfirmasi bahwa untuk masalah kompleks seperti Minesweeper, strategi yang hanya mengandalkan heuristik sederhana memiliki batas kinerja yang jelas, yang dapat dilampaui dengan menerapkan analisis logis yang lebih mendalam.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan perancangan, implementasi, dan analisis kinerja yang telah dipaparkan pada bab-bab sebelumnya, dapat ditarik beberapa kesimpulan utama. Penelitian ini bertujuan untuk membuktikan bahwa arsitektur solver hibrida yang mengintegrasikan Algoritma Greedy dengan modul analisis Constraint Satisfaction Problem (CSP) mampu memberikan kinerja yang lebih superior dibandingkan solver yang hanya mengandalkan Algoritma Greedy murni.

Hasil eksperimen yang dilakukan pada 1.000 percobaan untuk tiga konfigurasi papan yang berbeda secara konsisten menunjukkan bahwa Solver Hibrida secara signifikan melampaui kinerja Solver Classic Greedy. Terbukti bahwa Solver Hibrida mampu mencapai tingkat keberhasilan (win rate) yang lebih tinggi rata-rata 11.2% di semua tingkat kesulitan yang diuji. Peningkatan ini secara langsung

disebabkan oleh kemampuan modul CSP untuk menyelesaikan pola-pola lokal yang ambigu, yang merupakan kelemahan fundamental dari pendekatan Greedy murni yang terpaksa melakukan tebakan acak.

Analisis lebih lanjut juga mengungkapkan dinamika kinerja yang menarik. Meskipun Solver Hibrida membutuhkan lebih banyak langkah untuk memenangkan permainan, hal ini justru mengindikasikan kemampuannya untuk bertahan lebih lama dan menyelesaikan papan-papan yang lebih sulit. Terkait waktu eksekusi, penambahan biaya komputasi dari modul CSP terbukti merupakan trade-off yang sepadan dengan peningkatan drastis pada tingkat keberhasilan.

Dengan demikian, hipotesis awal penelitian ini telah terbukti benar. Implementasi strategi hibrida adalah pendekatan yang sangat efektif untuk mengatasi permasalahan probabilistik dalam puzzle Minesweeper.

B. Saran

Meskipun solver hibrida yang dikembangkan telah menunjukkan kinerja yang sangat baik, selalu ada ruang untuk pengembangan lebih lanjut guna mendekati batas teoretis penyelesaian Minesweeper. Berikut adalah beberapa saran untuk penelitian di masa depan:

1. Implementasi Analisis Global Antar-Frontier:

Kekurangan utama dari solver saat ini adalah ia menganalisis setiap kelompok frontier secara terpisah. Langkah pengembangan berikutnya yang paling berdampak adalah mengimplementasikan analisis global. Dengan membandingkan semua kemungkinan jumlah bom pada setiap frontier yang terpisah terhadap jumlah total bom yang tersisa di papan, solver dapat menghilangkan konfigurasi yang tidak mungkin secara global. Teknik ini berpotensi menyelesaikan beberapa kasus tebakan paksa (seperti 50/50) dan meningkatkan win rate lebih jauh lagi.

2. Optimisasi Performa Modul CSP:

Fungsi solve rekursif adalah bagian yang paling mahal secara komputasi. Kinerjanya dapat ditingkatkan dengan teknik pruning yang lebih agresif pada fungsi isPartialConfigurationValid(). Implementasi constraint propagation sederhana, di mana deduksi baru dari sebuah asumsi langsung diperiksa konsistensinya, dapat memangkas cabang pencarian lebih awal dan mempercepat analisis pada CSP yang kompleks.

3. Heuristik Tie-Breaking yang Lebih Canggih:

Saat ini, jika modul CSP menemukan beberapa sel dengan probabilitas bom terendah yang sama, solver akan memilih salah satunya secara arbitrer. Heuristik tie-breaking dapat ditambahkan, misalnya dengan memilih sel yang memiliki potensi membuka informasi paling banyak (yaitu, bertetangga dengan jumlah sel tertutup paling banyak), untuk membuat tebakan menjadi lebih strategis.

References

[1] R. Kaye, "Minesweeper is NP-complete," Mathematical Intelligencer, vol. 22, no. 2, pp. 9-15, 2000.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. Cambridge, MA: MIT Press, 2009.

[3] I. Stewart, "Minesweeper and the 'P=NP?' problem," in Math Hysteria: Fun and Games with Mathematics. Oxford: Oxford University Press, 2004, pp. 135-146.

Lampiran

Link repository GitHub:
https://github.com/bill2247/Minesweeper-Solver_Makalah-STIMA-2025

Link Video YouTube:
https://youtu.be/xal0TSXVP3c

Link Salindia Video:

https://www.canva.com/design/DAGrRwMmEHY/Z01IpPfEd12EI-6bnT-K4A/edit

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, 21 Juni 2025

Sabilul Huda
13523072