Implementasi Algoritma Greedy untuk Pengambilan Keputusan Optimal Jangka Pendek pada Papan Congklak

Beni Lesmana - 10122043
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : benilesmana48@gmail.com, 10122043@mahasiswa.itb.ac.id

***Abstract*—Congklak adalah permainan papan tradisional yang membutuhkan strategi untuk mengumpulkan jumlah biji terbanyak di lumbung. Karena sifat permainan yang memiliki probabilitas percabangan langkah yang besar, pencarian langkah optimal mutlak secara komprehensif sangat sulit dilakukan dan memakan waktu komputasi yang tinggi. Makalah ini mengusulkan penggunaan pendekatan Algoritma Greedy untuk menentukan langkah optimal jangka pendek pada setiap giliran. Fungsi heuristik yang dirancang memprioritaskan langkah yang menghasilkan giliran tambahan (*free turn*) serta perolehan jumlah poin terbanyak pada satu putaran, tanpa memikirkan langkah lawan berikutnya. Pengujian dilakukan dengan menggunakan metode simulasi Monte Carlo sebanyak 1000 iterasi dengan mempertandingkan agen *Greedy* melawan agen *Random*. Hasil simulasi menunjukkan bahwa pendekatan heuristik *Greedy* dasar ini berhasil memberikan kemenangan yang dominan melawan langkah acak dengan rata-rata skor akhir stabil di atas target aman, membuktikan bahwa peninjauan lokal-optimal sudah sangat mumpuni untuk permainan Congklak.**

***Index Terms*—Congklak, Algoritma Greedy, Heuristik, Pengambilan Keputusan, Monte Carlo, Kecerdasan Buatan, Teori Permainan**

I. Pendahuluan

Seiring dengan perkembangan pesat dalam bidang kecerdasan buatan (*Artificial Intelligence*) dan teori permainan (*Game Theory*), pencarian algoritma pemecahan masalah untuk permainan tradisional kembali mendapat sorotan tajam dari para peneliti. Permainan tradisional tidak hanya berfungsi sebagai warisan budaya, tetapi juga menawarkan model simulasi diskret yang sangat kaya akan variasi matematis. Congklak, atau sering juga disebut dakon, merupakan salah satu permainan papan tertua yang diyakini berasal dari wilayah Timur Tengah dan telah berasimilasi menjadi bagian dari tradisi dan budaya lokal di Indonesia selama berabad-abad [1]. Permainan ini menuntut pemain untuk tidak sekadar memindahkan biji secara acak, melainkan harus menyusun sebuah strategi supaya akumulasi biji di lumbung milik mereka pada akhir permainan lebih banyak daripada milik lawan.

Meskipun aturan dasarnya tampak sederhana secara kasat mata, yaitu hanya memindahkan biji dari satu lubang ke lubang lainnya secara berurutan, permainan ini memiliki tingkat kompleksitas ruang pencarian (*state space*) yang sangat tinggi jika dianalisis dari kacamata ilmu komputer. Setiap pilihan lubang awal yang dieksekusi oleh seorang pemain akan memicu reaksi berantai (*chain reaction*) perpindahan biji yang dapat mengubah susunan papan secara drastis dalam satu waktu. Jumlah percabangan kemungkinan papan baru (yang sering disebut sebagai faktor percabangan) berkisar hingga 7 opsi per giliran. Mengingat satu permainan Congklak dapat berlangsung selama puluhan hingga ratusan giliran, pohon permainan (*game tree*) yang terbentuk akan tumbuh secara eksponensial. Hal ini membuat pencarian jalur kemenangan absolut menggunakan algoritma penelusuran komprehensif (*exhaustive search*) menjadi sangat tidak efisien secara waktu maupun alokasi memori komputer.

Oleh karena itu, sangat dibutuhkan perancangan model heuristik untuk mengatasi masalah batasan komputasi eksponensial tersebut. Salah satu alternatif penyelesaian komputasi heuristik yang paling baik dan efisien adalah dengan memetakan papan congklak ke dalam struktur memori linear yang ringan dan menggunakan pendekatan Algoritma *Greedy*. Algoritma *Greedy* adalah keluarga algoritma optimasi yang prinsip kerjanya didasarkan pada filosofi mengambil apa yang bisa didapatkan sekarang, di mana agen komputasional akan selalu mengambil pilihan lokal terbaik yang dapat dievaluasi pada saat itu juga, tanpa memedulikan konsekuensi atau jebakan yang mungkin menanti pada masa depan [2]. Walaupun tidak selalu menghasilkan optimum secara global, kinerja kecepatan proporsional konstan yang dimilikinya menjadikannya andalan dalam implementasi di dunia nyata.

Makalah teknis ini secara khusus bertujuan untuk mengimplementasikan Algoritma *Greedy* sebagai agen pencari keputusan untuk bermain Congklak. Di samping itu, makalah ini juga akan menyajikan pemodelan matematis permainan, penjabaran struktur kode representatif, dan diakhiri dengan analisis komprehensif terkait sejauh mana tingkat efektivitas kinerjanya melalui pengujian pertandingan tunggal maupun pengujian simulasi menggunakan metode Monte Carlo.

II. Kajian Pustaka dan Dasar Teori

A. Latar Belakang

Secara historis, Congklak merupakan varian dari keluarga permainan papan Mancala yang lazim ditemukan di wilayah Afrika dan Timur Tengah. Perdagangan dan akulturasi budaya membawa permainan ini ke Nusantara di mana ia dinamai berbeda-beda pada setiap daerah (misalnya Dakon di Jawa, Mokaotan di Sulawesi, dan Congkak di Sumatera) [1]. Pada masa lalu, permainan ini seringkali dimanfaatkan oleh masyarakat agraris sebagai medium untuk melatih kemampuan berhitung cepat serta strategi manajemen sumber daya alam (yang disimbolkan melalui pemindahan biji-bijian).

Bagi dunia informatika, permainan papan seperti Congklak merupakan medan uji coba yang ideal untuk menguji kapabilitas algoritma kecerdasan buatan karena statusnya sebagai permainan informasi sempurna (*perfect information game*) di mana tidak ada faktor tersembunyi seperti dalam permainan kartu, sehingga seluruh ruang solusi secara teoretis bisa dipetakan meskipun jumlahnya sangat masif.

B. Aturan Permainan Congklak Secara Formal

Permainan Congklak di Indonesia memiliki berbagai ragam variasi aturan lokal. Oleh karena itu, dalam makalah komputasional ini, kita menyepakati sebuah aturan baku yang terstandarisasi agar agen dapat diprogram secara presisi. Implementasi standar yang digunakan mengadopsi aksioma berikut:

1) Papan congklak terbagi secara simetris atas dua wilayah. Setiap pemain memegang kendali atas 7 lubang kecil di area terdekatnya dan 1 lumbung besar di ujung kirinya.
2) Sebanyak 98 keping biji didistribusikan secara seragam ke dalam 14 lubang kecil, yang mana setiap lubang kecil akan diinisialisasi dengan tepat 7 keping biji di awal mula pertandingan.
3) Pada giliran berjalannya, pemain berhak mengangkat seluruh biji dari salah satu lubang kecil miliknya, dan kemudian menjatuhkannya satu per satu searah jarum jam. Alur distribusi akan melewati lumbung milik pemain yang sedang berjalan, namun wajib melompati lumbung milik musuh.
4) Jika biji terakhir mendarat di dalam lumbung milik sendiri, status permainan bergeser menguntungkan pemain tersebut dengan memberikannya **giliran tambahan** secara langsung.
5) Jika biji terakhir mendarat di sebuah lubang yang kosong di area kontrol pemain itu sendiri, dan lubang milik lawan yang berhadapan persis di seberangnya berisi biji, maka sebuah **tembakan** dipicu. Biji yang berada di kedua lubang tersebut dipindahkan ke dalam lumbung milik pemain secara serentak.
6) Permainan berakhir manakala seluruh 7 lubang di salah satu sisi papan kehabisan biji seutuhnya. Pemain dengan kuantitas biji terbanyak di lumbung akan menjadi pemenang.

C. Landasan Teori Algoritma Greedy

Algoritma *Greedy* merupakan salah satu teknik esensial dalam memecahkan permasalahan optimasi diskret dalam ranah strategi algoritma. Algoritma ini mencari solusi dengan cara merangkai keputusan demi keputusan secara iteratif. Pada tahap manapun dari algoritma, pencari keputusan akan selalu mengeksekusi pilihan yang terlihat mengisyaratkan keuntungan paling besar pada momen tersebut (yang disebut sebagai pencarian lokal optimal), dilandasi oleh asumsi bahwa rentetan optimasi lokal ini nantinya akan mengerucut pada pencapaian optimal secara global [3].

Secara konseptual, algoritma *Greedy* dibangun oleh beberapa elemen inti, yakni:

- **Himpunan Kandidat:** Himpunan yang mendefinisikan sekumpulan pilihan yang dapat dieksekusi secara legal pada satu interval waktu.
- **Himpunan Solusi:** Tempat penyimpanan elemen-elemen langkah yang telah terpilih.
- **Fungsi Kelayakan:** Modul fungsi logis yang bertugas memeriksa secara preventif apakah sebuah kandidat melanggar konstrain atau aturan hukum (sistem).
- **Fungsi Objektif:** Metrik kalkulasi heuristik yang digunakan untuk menyematkan skor evaluasi ke dalam masing-masing langkah kandidat guna keperluan pemeringkatan dan seleksi.

Terdapat satu pedoman paling fundamental dari implementasi algoritma ini: Keputusan yang telah diambil pantang untuk dievaluasi ulang (tidak ada mekanisme penelusuran balik). Begitu suatu langkah dipilih, tidak ada proses perbaikan di kemudian giliran.

III. Pemodelan Sistem dan Ruang Solusi

A. Model Struktur Data Linear Papan

Salah satu kontribusi utama pada perancangan komputasional ini adalah efisiensi pemetaan struktur papan. Alih-alih mendefinisikan papan sebagai matriks dua dimensi yang menyulitkan pergerakan iteratif searah jarum jam, ruang sirkuler papan dipetakan menjadi sebuah senarai linear (*linear array*) satu dimensi berkapasitas statis sebesar 16 alamat memori.

Secara perinci, arsitektur data tersebut adalah sebagai berikut:

- **Indeks 0 hingga 6:** Rentetan alamat memori yang mempresentasikan ketujuh lubang kecil milik Pemain 1 secara berurutan.
- **Indeks 7:** Titik lumbung milik Pemain 1.
- **Indeks 8 hingga 14:** Rentetan alamat memori yang mempresentasikan ketujuh lubang kecil milik Pemain 2.
- **Indeks 15:** Titik lumbung milik Pemain 2.

Pemetaan linear ini didemonstrasikan pada Gambar 1. Desain ini sangat hemat secara konsumsi memori komputer karena setiap mutasi pergerakan biji hanyalah operasi penambahan indeks modulus 16.

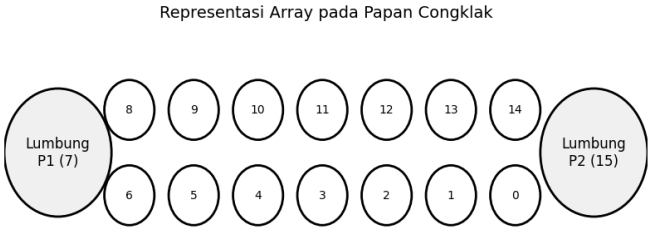

Fig. 1. Model Struktur Indeks Array pada Papan Congklak.

B. Logika Komputasi Berdasarkan Aturan Permainan

Untuk mereplikasi sifat perputaran tanpa batas di papan permainan secara virtual, logika matematika modulo akan digunakan. Sebagai contoh, pergerakan dari lubang milik Pemain 1 indeks 6 akan melangkah menuju indeks 7 (lumbung). Dari lumbung indeks 7, akan melanjutkan ekspansi ke indeks 8 yang merupakan teritori milik lawan. Ketika pergerakan biji pemain menyeberang dan mendarat pada indeks lumbung musuh, kontroler logika bertugas melewatkan indeks tersebut dengan klausa kondisional agar tidak menambahkan poin kepada musuh.

Sistem penembakan lawan didesain menggunakan formula simetris matematika sederhana. Jika biji jatuh di indeks X milik sendiri dan kosong, maka indeks lawan yang menjadi target rampasan selalu dapat ditemukan secara absolut dengan perhitungan matematis $14 - X$.

IV. Perancangan Algoritma Greedy

A. Pemetaan Elemen Greedy

Implementasi Algoritma *Greedy* memerlukan parameter logika yang kuat. Parameter ini diklasifikasikan menjadi rancangan sebagai berikut:

1) **Himpunan Kandidat:** Semua indeks yang mewakili lubang di sisi pemain saat ini, yang syarat nilainya tidak boleh kosong (lebih dari 0).
2) **Fungsi Heuristik:** Menilai skor sebuah langkah kandidat berdasarkan kalkulasi maksimalisasi perolehan poin murni dan potensi hak giliran tambahan.

B. Strategi Perancangan Heuristik

Agar agen *Greedy* memiliki kualitas yang baik, dirancanglah sebuah fungsi evaluasi objektif. Fungsi heuristik ini mengeksploitasi salah satu aturan bermain Congklak, yaitu mekanisme giliran tambahan. Mendapatkan giliran tambahan berulang-ulang artinya pemain dapat me-reset papan dan meraih poin mutlak tanpa merelakan kontrol kendali ke tangan lawan.

Alur perhitungan Fungsi Heuristik untuk setiap langkah dievaluasi dengan alur:

1) Eksekusi penyalinan variabel papan permainan agar simulasi tidak merusak nilai asli.
2) Terapkan fungsi simulasi pergerakan.
3) Hitung selisih jumlah biji di lumbung dari sebelum dan sesudah langkah berjalan.
4) Apabila di penghujung simulasi terdeteksi bahwa biji mendarat di lumbung sendiri, maka heuristik langkah disuntikkan bonus $+100$.
5) Rumus akhir merupakan penjumlahan antara selisih dari hasil simulasi dengan nilai bonus ketika mendapatkan giliran tambahan.

Algoritma kemudian mengurutkan nilai kandidat dan memilih pergerakan dengan nilai heuristik tertinggi. Bila terjadi nilai seri, langkah dipilih secara acak dari yang terbaik.

C. Pseudocode Algoritma Seleksi

Konsep heuristik tersebut diformulasikan ke dalam bentuk abstraksi kode semu yang tertera pada Algoritma 1.

Algorithm 1 Fungsi Seleksi Langkah Greedy

Require: $gameState$: representasi papan Congklak
Ensure: $bestMove$: indeks lubang optimal

$validMoves \leftarrow$ get_valid_moves($gameState$)
$bestScore \leftarrow -\infty$
$bestMoves \leftarrow [\,]$
for each $move \in validMoves$ **do**
  $simState \leftarrow$ deepcopy($gameState$)
  $scoreBefore \leftarrow simState.getStore()$
  $extraTurn \leftarrow simState.playMove(move)$
  $\Delta Poin \leftarrow simState.getStore() - scoreBefore$
  $heuristic \leftarrow \Delta Poin$
  if $extraTurn ==$ True **then**
    $heuristic \leftarrow heuristic + 100$
  end if
  if $heuristic > bestScore$ **then**
    $bestScore \leftarrow heuristic$
    $bestMoves \leftarrow [move]$
  else if $heuristic == bestScore$ **then**
    $bestMoves$.append($move$)
  end if
end for
return random_choice($bestMoves$)

D. Evaluasi Kompleksitas Teoretis

Pendekatan *Greedy* ini dijamin beroperasi pada batas kecepatan ekstrim. Misalkan b mewakili jumlah kandidat lubang pilihan (batas 7 buah) dan k adalah total biji yang ditransisikan. Algoritma *Greedy* ini akan menyimulasikan 7 cabang di memori, di mana tiap simulasi berjalan secara linier dan memakan pergerakan iteratif senilai jumlah k. Maka kompleksitas waktu eksekusi hanyalah $\mathcal{O}(b \times k)$.

Dari sisi pengunaan memori, karena algoritma tidak membangun penyimpanan struktur graf raksasa ke depan, ia hanya menyimpan salinan sementara array sebesar 16 elemen sebanyak jumlah cabang. Oleh karena itu, batasan konsumsi kompleksitas memori maksimalenada angka $\mathcal{O}(b)$ yang secara ekuivalen adalah batasan konstan $\mathcal{O}(1)$. Performa konstan ini mengeliminasi masalah eksponensial.

V. Pengujian Eksperimental dan Diskusi

A. Analisis Eksekusi Pertandingan Tunggal

Untuk menguji kebenaran algoritma, sebuah program komputasi telah dikembangkan menggunakan bahasa Python. Pengujian tahap awal dilakukan dengan menjalankan simulasi eksekusi satu pertandingan penuh antara algoritma *Greedy* dan agen Langkah Acak.

Gambar 2 menampilkan hasil eksekusi program melalui terminal yang mensimulasikan sesi bermain tunggal tersebut. Program mencatat bahwa keseluruhan proses permainan selesai dalam waktu komputasi murni yang sangat cepat, yaitu sebesar 4 milidetik. Hal ini memverifikasi bahwa kompleksitas $\mathcal{O}(b \times k)$ dari algoritma *Greedy* memungkinkan sistem merespon secara waktu-nyata (*real-time*).

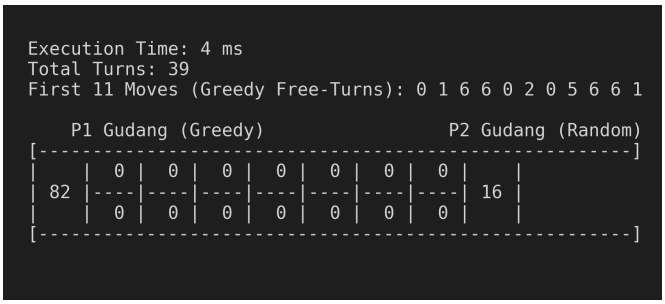

Fig. 2. Visualisasi Terminal: Hasil dan Statistik Akhir Pertandingan *Greedy* vs *Random*.

Untuk memberikan gambaran yang lebih transparan mengenai cara kerja fungsi heuristik di balik layar, berikut adalah dekonstruksi (*breakdown*) langkah demi langkah (*step-by-step*) dari mekanisme pengambilan keputusan agen *Greedy* pada beberapa putaran (*turn*) krusial di awal permainan:

1. Putaran Pertama (Kondisi Awal Papan)

Pada saat permainan baru dimulai, seluruh 14 lubang kecil terisi penuh oleh 7 biji. Agen *Greedy* (Pemain 1) melakukan simulasi pemindahan biji untuk ketujuh lubang miliknya (indeks 0 hingga 6).

- Jika agen mensimulasikan pilihan lubang indeks 0 (berisi 7 biji), biji ketujuh (terakhir) akan jatuh tepat pada indeks 7 (lumbung miliknya sendiri). Kondisi fisika papan ini mengaktifkan hak *extra turn*. Langkah ini langsung diberi tambahan bonus skor $+100$.
- Sementara itu, pilihan lubang lain (indeks 1 sampai 6) pada putaran pertama hanya memberikan poin deposit standar 1 biji tanpa memicu giliran tambahan maupun efek tembakan.
- **Keputusan Akhir:** Algoritma *Greedy* secara mutlak memilih mengeksekusi lubang indeks 0 dan mendapatkan hak istimewa untuk bermain kembali.

2. Putaran Kedua (Eksploitasi Beruntun)

Karena berhasil mendarat di lumbung, agen kembali mengevaluasi papan di putaran kedua. Kini lubang indeks 0 telah kosong, sehingga himpunan kandidat legal tersisa indeks 1 hingga 6.

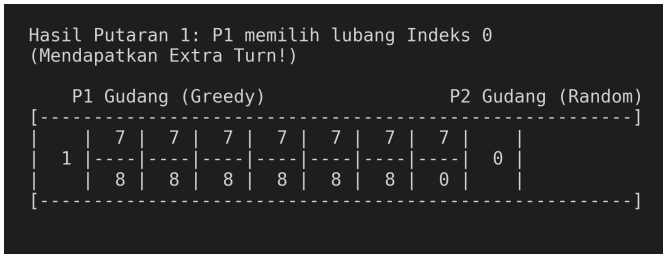

Fig. 3. Kondisi Papan Terminal Setelah Putaran 1 (Mendapat *Extra Turn*).

- Agen melakukan penelusuran heuristik pada lubang indeks 1 (yang kini berisi 8 biji akibat sebaran amunisi dari langkah pertama). Biji terakhir akan jatuh di teritori musuh dan memicu pergerakan sirkuler berantai (karena menimpa lubang berisi). Setelah disimulasikan hingga biji terakhir benar-benar berhenti mendarat, ternyata jalur rantai pergerakan tersebut lagi-lagi berujung pada pendaratan di lumbung agen.
- **Keputusan Akhir:** Karena algoritma kembali menemukan jalur komputasi yang memicu *extra turn* (mendapatkan bonus $+100$), agen langsung mengeksekusi lubang indeks 1 tanpa ragu.

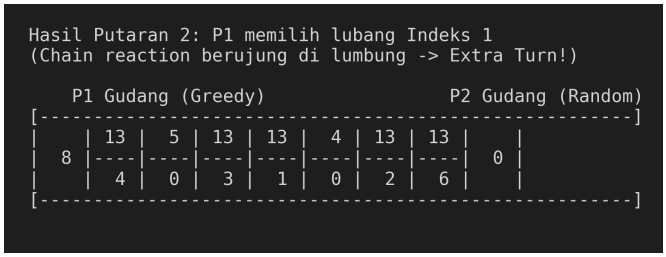

Fig. 4. Kondisi Papan Terminal Setelah Putaran 2 (*Chain-reaction* berujung ke lumbung).

3. Kondisi Kritis Pemicu Penembakan (Capture)

Pada suatu titik di pertengahan simulasi permainan ketika *extra turn* sudah tidak lagi tersedia di cabang kandidat manapun, agen otomatis beralih murni mengevaluasi kandidat langkah berdasarkan pertambahan panen terbesar.

- Agen mendeteksi secara matematis bahwa memindahkan biji dari lubang indeks 4 akan berakhir tepat menempati lubang kosong indeks 5 miliknya sendiri.
- Di seberang lubang indeks 5 tersebut (indeks 9 milik lawan), terdapat tumpukan akumulasi 8 biji milik musuh.
- Mengingat *extra turn* tidak bisa dicapai (ketiadaan bonus $+100$), agen murni mengevaluasi berdasarkan total pertambahan poin terbesar.
- **Keputusan Akhir:** Karena lubang indeks 4 memanen total 9 biji (1 milik sendiri + 8 rampasan musuh), agen secara mutlak mengorbankan opsi lubang lain yang mungkin hanya mendepositkan 2-3 biji secara lambat. Agen memprioritaskan eksekusi lubang indeks 4 untuk merampas seketika (menembak) seluruh tumpukan milik

lawan, memberikan asupan panen masif ke dalam lumbungnya.

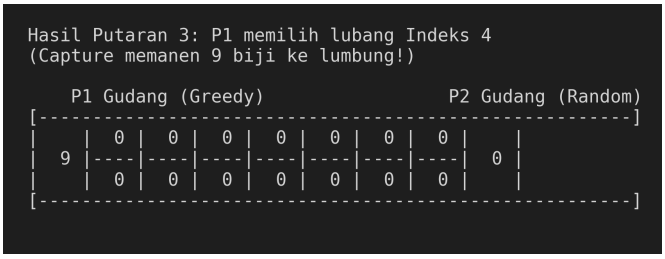

Fig. 5. Kondisi Papan Terminal Setelah Putaran 3 (Berhasil *Capture* 9 biji ke lumbung).

Dekonstruksi ketiga sampel langkah di atas membuktikan bahwa meskipun algoritma ini tidak menelusuri atau memprediksi pohon permainan di masa depan layaknya algoritma *Minimax*, fungsi insting rakusnya (*myopic function*) sudah didesain cukup presisi dalam mengamankan keunggulan materi poin secara cepat.

Permainan berakhir dalam total 39 giliran (*turns*). Dari data lintasan eksekusi yang ditampilkan pada terminal, agen *Greedy* mendemonstrasikan keunggulan absolutnya dengan mencetak 11 kali giliran tambahan secara beruntun di fase awal permainan, mengeksploitasi fitur *free-turn* secara sempurna berkat heuristik evaluasi pendeknya. Di akhir pertandingan, Pemain *Greedy* menumbangkan pemain Acak dengan telak, mencetak total 82 biji berbanding 16 biji.

B. Simulasi Statistik Dengan Monte Carlo

Untuk menyingkirkan kemungkinan bias acak, akan digunakan instrumen *Monte Carlo*. Algoritma diputar sebanyak 1000 iterasi pertandingan secara berturut-turut, dibagi rata menjadi posisi pemain awalan dan pemain akhiran. Grafik 6 memberikan ilustrasi distribusi kemenangan telak. Agen *Greedy* memenangkan 995 kali, seri 2 kali, dan mengalami kekalahan sebanyak 3 kali.

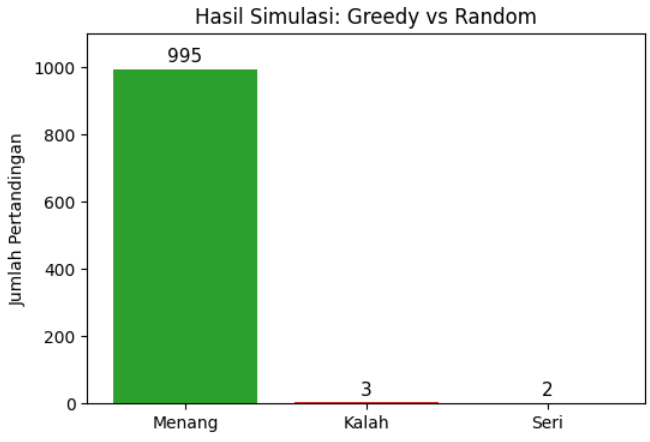

Fig. 6. Grafik Hasil Kemenangan Simulasi (1000 Iterasi).

Tabel I merangkum distribusi skor penangkapan deposit di lumbung.

TABLE I
STATISTIK PEMUSATAN SKOR LUMBUNG AGEN GREEDY

Metrik Pengujian	**Jumlah Biji**
Expected Value (Rata-rata)	75.87
Nilai *Best Case* (Tertinggi)	94
Nilai *Worst Case* (Terendah)	44

C. Diskursus Signifikansi Kinerja

Pengaplikasian pengondisian heuristik deterministik lokal dengan penambahan **bonus nilai** giliran tambahan telah menghasilkan kekalahan mutlak bagi agen musuh dengan persentase kemenangan di angka 99.5%. Berhasil mendapatkan hak untuk memperoleh giliran transisi yang beruntun menjadi pemicu anomali *Snowball Effect* yang menghancurkan ritme acak.

Nominal *Expected Value* yang tinggi pada nilai rata-rata 75.87 biji sangat krusial, mengingat syarat sah memenangkan pertandingan Congklak hanyalah menguasai separuh lebih biji keseluruhan (melebihi 49 biji).

Meskipun hasil simulasi sangat dominan, agen *Greedy* tercatat mengalami beberapa kali margin poin tipis atau kekalahan. Hal ini membuktikan bahwa algoritma penelusuran satu kedalaman (*1-depth search*) masih memiliki kelemahan struktural bawaan.

D. Analisis Post-Mortem dan Keterbatasan Greedy

Berdasarkan data statistik simulasi Monte Carlo yang mencatat raupan skor terendah (*Worst Case*) di angka kritis 44 hingga 49 poin, kemunduran agen secara fundamental diakibatkan oleh fenomena *short-sightedness* (rabun dekat) yang secara alami memang melekat pada definisi algoritma *Greedy* konvensional. Algoritma ini dirancang secara eksklusif untuk hanya mengevaluasi parameter profitabilitas pada saat transisi kondisi papan sekarang ke kondisi $T+1$ (giliran sendiri), sama sekali tanpa membekali dirinya dengan instrumen proyeksi visibilitas untuk melihat probabilitas kondisi $T+2$ (giliran balasan lawan).

Kelemahan ”kebutaan masa depan” ini memicu efek bumerang yang dapat didekonstruksi dan dibuktikan kelemahannya melalui skenario simulasi konkret di atas papan:

- **Pengorbanan Posisi Pertahanan Demi Poin Marginal:** Misalkan agen *Greedy* dihadapkan pada dua pilihan lubang yang sah. Lubang pertama memberikan panen aman ke lumbung sebesar 2 biji. Lubang kedua memberikan panen 3 biji, namun kalkulasi perputaran mensyaratkan biji terakhir dari lubang tersebut akan mendarat di sebuah lubang yang persis berhadapan dengan lubang musuh yang sedang kosong dan siap menembak.
- **Eksekusi Kebutaan Heuristik:** Karena metrik evaluasi objektif pada heuristik *Greedy* diatur untuk membandingkan selisih murni Δ poin lumbung sesaat sesudah langkah dieksekusi ($3 > 2$), algoritma akan secara deterministik dan mutlak memilih lubang kedua.
- **Anatomi Kegagalan:** Keputusan instan tersebut memang mutlak lebih menguntungkan secara matematis pada detik itu (karena ada untung bersih margin +1 biji lebih besar).

Namun, dengan memaksakan kehendak menjatuhkan biji terakhirnya di posisi yang secara taktis terekspos, agen secara sukarela menyerahkan kontrol papan dan meletakkan poinnya tepat di zona tembak musuh. Pada detik berikutnya ketika giliran beralih, musuh dapat dengan mudah melakukan manuver ”tembakan”, merampas tidak hanya 1 biji umpan yang baru saja diletakkan agen, tetapi juga seluruh akumulasi tumpukan biji agen yang telah dikumpulkan di sisi papan tersebut.

Dengan kata lain, agen *Greedy* rela menukar sebuah keuntungan jangka pendek yang nilainya sangat sepele (hanya sebesar 1 poin marginal) dengan sebuah kerugian masif yang berpotensi merenggut belasan poin dari tangannya di giliran kelak.

Fenomena kegagalan akibat *blind-baiting* (umpan buta) ini sangat berbahaya jika agen *Greedy* tersebut diadu melawan agen cerdas tingkat lanjut (seperti agen *Minimax* atau agen manusia) yang dapat dengan sengaja mengosongkan lubang-lubang strategisnya sejak awal hanya untuk memancing *Greedy* masuk ke dalam perangkap. Dalam percobaan ini, karena uji coba dilakukan melawan agen langkah acak (*Random*), probabilitas musuh memanfaatkan celah ini secara terstruktur sangatlah rendah, sehingga tingkat kemenangan keseluruhan agen *Greedy* masih kokoh berdiri di atas 99.5%.

Namun demikian, analisis *post-mortem* ini telah menunjukkan dan memvalidasi dalil teoretis ilmu komputer bahwa tidak ada algoritma heuristik optimasi satu lapis (*single-layer*) yang dapat luput sepenuhnya dari kecacatan fatal jebakan *local optimum*. Algoritma *Greedy* sangat cepat, namun sangat naif.

VI. Kesimpulan

Pendekatan resolusi permasalahan optimasi menggunakan algoritma *Greedy* terbukti sebagai solusi yang baik dan sangat efisien dalam menyelesaikan polemik komputasi eksponensial di permainan Congklak. Evaluasi heuristik satu-kedalaman yang fokus mengeksploitasi giliran ekstra sanggup menembus *win rate* sebesar 99.5% melawan sistem langkah acak dengan komputasi asimtotik waktu linier dan memori statis $\mathcal{O}(1)$. Untuk pengembangan ke depan (*future work*), disarankan penambahan algoritma *Minimax* sedalam satu atau dua lapis turunan untuk menutup celah titik buta yang memicu kekalahan agen.

References

[1] B. W. Kusumo, dkk., *Nilai dan Potensi Kognitif Permainan Tradisional Nusantara*. Jurnal Pelestarian Sejarah dan Nilai Tradisional, vol. 7, no. 2, hal. 11-23, 2023.

[2] D. Santoso, “Pendekatan Algoritma Greedy pada Penyelesaian Masalah Optimasi Keputusan Lokal,” *Jurnal Ilmu Komputer dan Informatika*, vol. 12, no. 1, hal. 45-56, 2022.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, dan C. Stein, *Introduction to Algorithms*, edisi ke-3. Cambridge, MA, USA: MIT Press, hal. 414-416, 2009.

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

(Beni Lesmana - 10122043)