SOLUSI LANGKAH PADA GAME CONGKLAK MENGGUNAKAN METODE NEGASCOUT

Rangga Darmawan[1], Galih Hermawan[2]

Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-116 Bandung
E-mail : dar.rangga@gmail.com[1], galih.hermawan@email.unikom.ac.id[2]

ABSTRAK

Permainan congklak merupakan permainan tradisional Indonesia. Permainan ini sebenarnya bukanlah permainan asli Indonesia melainkan permainan dari luar Indonesia yang sudah diadaptasi berdasarkan budaya Indonesia. Pada permainan congklak diperlukan perhitungan matematis dalam memilih biji yang akan dimainkan dengan tepat dan dalam waktu yang cepat agar mendapatkan hasil yang optimal atau dalam kondisi yang paling menguntungkan.

Pada penelitian ini aplikasi game congklak yang menerapkan Algoritma Negascout, dibangun dengan tujuan untuk mengetahui kinerja Algoritma Negascout pada proses pencarian solusi game congklak. Kinerja algoritma akan dilihat dari segi tingkat kemenangan, waktu pemrosesan, dan jumlah langkah pencarian.

Berdasarkan hasil pengujian aplikasi secara keseluruhan, maka dapat disimpulkan bahwa akurasi dari perhitungan Negascout secara manual dan oleh aplikasi adalah 100%, tingkat kemenangan algoritma minimax lebih unggul dari algorima Negascout dengan presentase kemenangan 71% untuk minimax dan 57% untuk Negascout, tingkat kecepatan algoritma Negascout lebih baik dari algoritma minimax dengan perbandingan jumlah keseluruhan langkah pencariannya 2:5 dan perbandingan keseluruhan waktu proses pencariannya 1:10691,42.

Kata kunci : Congklak, Kecerdasan Buatan, Negascout.

1. PENDAHULUAN

Permainan congklak merupakan permainan tradisional Indonesia. Permainan ini sebenarnya bukanlah permainan asli Indonesia melainkan permainan dari luar Indonesia yang sudah diadaptasi berdasarkan budaya Indonesia. Congklak dimainkan oleh dua orang, dalam permainan ini biasanya menggunakan papan dengan 16 lubang yang terdiri dari 7 lubang kecil setiap sisinya dan 2 buah lubang besar. Setiap lubang kecil yang ada akan diisi dengan biji congklak masing-masing 7 buah.

Pada permainan congklak diperlukan perhitungan matematis dalam memilih biji yang akan dimainkan dengan tepat dan dalam waktu yang cepat agar mendapatkan hasil yang optimal atau dalam kondisi yang paling menguntungkan. Penelitian yang dilakukan oleh Galih Hermawan [1], teknik penentuan langkah menggunakan algoritma Greedy disertai dengan mekanisme pemilihan langkah secara dinamis. Yaitu pilihan untuk mengambil keuntungan sebesar-besarnya atau lebih menyelamatkan lubang yang berisi biji dengan jumlah relatif besar. Sedangkan pada penelitian Citra Anindya [2], metode penentuan langkah yang digunakan adalah algoritma Minimax. Dalam penelitian ini telah ditemukan solusi yang cukup optimal; dalam artian dari percobaan 21 kali bermain, agen komputer dapat memenangkan permainan sebanyak 15 kali dengan asumsi lawan *human* (manusia) adalah seorang yang cukup terampil dalam bermain congklak.

Adapun penelitian yang dilakukan oleh Sarah Nurhasanah [3], terfokus hanya pada mencari solusi yang optimal tanpa mempertimbangkan waktu komputasi yang dibutuhkan untuk mencari solusi. Dalam penelitian tersebut mencoba menerapkan Alpha-Beta Pruning optimasi dari algoritma Minimax untuk meningkatkan kecepatan komputasinya. Dari hasil penelitiannya menunjukkan bahwa kompleksitas algoritma Minimax dari penelitian Citra adalah T(n)=n, waktu pemrosesan yang dibutuhkan pada kondisi papan 7 lubang dengan isi lubang kecil masing-masing 3 biji adalah 27,8301 detik dan langkah pencarian yang dibutuhkan adalah 430 langkah. Dalam buku J. Mandziuk [4] ada algoritma lain yang dikemukakan oleh Alexander Reinefeld yaitu Algoritma Negascout yang merupakan optimasi dari algoritma Minimax. Dalam bukunya disebutkan Algoritma ini bisa melakukan pencarian solusi yang lebih cepat dan mendapatkan solusi yang lebih optimal.

Berdasarkan penjelasan tersebut, maka dalam penelitian ini akan mengimplementasikan algoritma Negascout pada game congklak. Diharapkan dengan adanya penelitian ini proses pencarian solusi bisa dilakukan dengan lebih cepat dalam artian kompleksitas algoritma, waktu pemrosesan dan langkah pencarian yang dibutuhkannya memiliki nilai yang lebih kecil dari algoritma Minimax, dan solusi yang didapatkan lebih optimal dalam artian

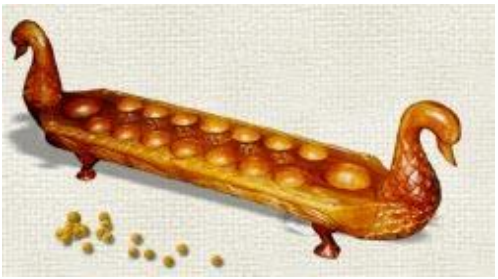
tingkat kemenangannya lebih besar dari algoritma Minimax.

2. ISI PENELITIAN

2.1 Game Congklak

Congklak merupakan suatu permainan tradional yang yang dimainkan oleh 2 orang dengan menggunakan media papan. Biasanya dalam permainan ini menggunakan batu atau sejenis kulit kerang yang digunakan sebagai alat permainan [5].

Papan congklak terdiri dari sejumlah lubang kecil dan dua lubang besar yang terdapat di masing-masing tiap ujung papan. Setiap pemain memiliki daerahnya sendiri. Pada awal permainan, semua lubang yang ada dipapan diisi dengan jumlah yang sama (batu, biji atau kerang). Jumlah isi di setiap lubang biasanya sama dengan jumlah lubang di salah satu sisi. Tujuan dari permainan ini adalah untuk mengumpulkan isi lubang sendiri dan mengosongkan isi lubang lawan yang ada di papan permainan. Pemain yang memenangkan permainan adalah pemain yang mengumpulkan batu / kulit kerang terbanyak.



Gambar 1. Papan Game Congklak

2.2 Kecerdasan Buatan

Definisi kecerdasan buatan atau *artificial intelligent* yang paling tepat untuk saat ini adalah *Acting rationally* dengan pendekatan *Rational agent*. Hal ini berdasarkan pemikiran bahwa komputer bisa melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut [6].

Kecerdasan buatan atau AI merupakan kegiatan membuat komputer agar dapat berpikir dan mengerjakan kegiatan yang dapat dilakukan oleh manusia maupun binatang. Saat ini dapat ditemukan program komputer yang memiliki kemampuan menangani masalah seperti *aritmatic*, *sorting*, *searching* [6]. Bahkan komputer juga dapat bermain beberapa *board game* seperti catur lebih baik dari pada manusia. Namun masih banyak hal yang tidak dapat dilakukan dengan baik oleh komputer. Seperti, mengenali wajah, berbicara bahasa manusia, menentukan sendiri apa yang harus dilakukan, dan bertingkah kreatif. Hal itu semua merupakan domain dari AI untuk mencoba menentukan algoritma apa yang dibutuhkan untuk memenuhi kebutuhan diatas.

Dalam bidang akademik, beberapa peneliti AI termotivasi oleh filosofi, yaitu memahami alam pikiran dan alam kecerdasan dan membangun program untuk memodelkan bagaimana proses berpikir. Beberapa juga termotivasi oleh psikologi, bertujuan untuk memahami mekanisme otak manusia dan proses mental. Dan lainya termotivasi oleh *engineering*, dengan tujuan membangun algoritma untuk melakukan kegiatan seperti manusia atau hewan. Dalam pembangunan Game, umumnya akan cenderung hanya pada sisi engineering yang bertujuan membangun algoritma yang dapat membuat Game karakter mengerjakan kegiatan seperti yang dilakukan manusia atau binatang [6].

2.3 Algoritma Negascout

Pada tahun 1982 dan 1983 dibuat dua algoritma yang mirip, yaitu algoritma PVS dan Algoritma Negascout. Algoritma negascout diperkenalkan oleh Alexander Reinefeld yang menggabungkan konsep algoritma SCOUT dan algoritma alpha-beta dalam notasi negamax [4].

Negascout game tree mirip dengan SCOUT, tetapi menggunakan notasi negamax dan menggunakan beberapa optimasi. Implementasinya mengasumsikan nilai integer dari fungsi evaluasi, seperti dalam gambar 2. Dalam memperkecil jendela pencarian agar mendapat jendela pencarian dengan lebar nol. Negascout akan memotong hampir semua cabang-cabang dari pohon, untuk membuat suatu pencarian yang sangat tepat. Tetapi sayangnya, ia akan memotong semua cabang-cabang berguna bersamaan dengan yang tidak berguna. Hal ini tidak berlaku jika algoritma dimulai dengan hasil yang benar. Satu jendela nol dapat dilihat sebagai uji coba. Uji coba dilakukan jika nilai sebenarnya sama dengan nilai yang ditebak.

```
1   function minimax(node,depth)
2   if depth <= 0 then
3       {positive values are good for the maximizing player}
4       {negative values are good for the minimizing player}
5       return objective_value(node)
6   end
7       {maximizing player is (+1)}
8       {minimizing player is (-1)}
9   local alpha = -node.player * INFINITY
10  local child = next_child(node,nil)
11  while child ~= nil do
12      local score = minimax(child,depth-1)
13      alpha = node.player==1 and math.max(alpha,score) or
        math.min(alpha,score)
14      child = next_child(node,child)
15  end
16  return alpha
17  end
```

Gambar 2. Algoritma Negascout

Keterangan pseudo code pada Gambar 2 :
s = kondisi papan
MAX_DEPTH = kedalaman maksimum pohon yang ditelusuri
successor si of s = lebar pohon(banyak langkah yang mungkin terjadi pada papan)
evaluate(s) = merupakan fungsi yang memberikan nilai evaluasi pada kondisi papan "s"

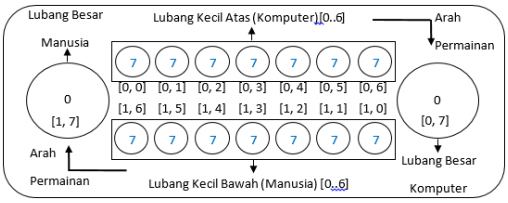
Gambar 2 merupakan *pseudocode* untuk algoritma Negascout, berdasarkan algoritma diatas dapat dijelaskan cara kerja algoritma Negascout sebagai berikut, statement pertama sama seperti alpha-beta, yaitu jika posisi s adalah *node leaf* maka Negascout akan mengembalikan nilai fungsi evaluasi (baris 2). Selain itu variabel m dan n diinisialisasi dengan -∞ dan beta (baris 3 dan 4). Kemudian Negascout akan melakukan "scout" pada node anak dari s dari kiri ke kanan. Node anak paling kiri akan dicari dengan menggunakan *interval(-beta,-alpha*) dan anak lainnya dicari dengan *zero-width window* (-m-1, -m) yang sudah diisi pada baris 11 sesudah melakukan pencarian anak paling kanan, karena null window ini tidak memiliki elemen, maka pencarian pasti akan gagal. Arah dari kegagalan ini menunjukkan apakah *node* tersebut dapat dipotong atau tidak [8].

Jika *null window* gagal karena t > m (baris 7), negascout harus mengecek kembali *node* tersebut dengan *search window* yang lebih lebar untuk mengetahui nilai aslinya. Hanya terdapat dua kasus dimana tidak diperlukan pencarian ulang yaitu, ketika n=beta (baris 8), dan negascout's "*fail-soft refinement*"selalu mengembalikan nilai Minimax yang benar pada dua level terbawah. Pada kasus lainnya pencarian ulang harus dilakukan dengan menggunakan *search window* baru yaitu (-beta,-t) (baris 9). Kondisi untuk pemotongan (baris 10) sama seperti *alpha-beta* : jika m ≥ *beta* maka semua *node* anak lain dapat diabaikan [7].

2.4 Analisis Masalah

Permainan ini merupakan permainan strategi matematik yang dapat mengasah keterampilan, daya hitung dan pola pikir pemain, permainan ini menggunakan papan untuk tempat permainannya.

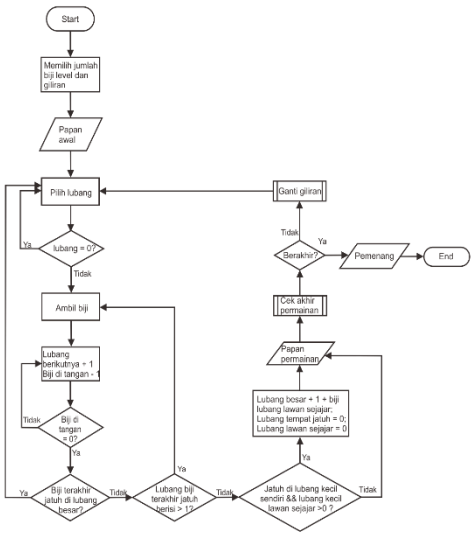
Pada permainan congklak terdapat dua sisi yaitu sisi pemain komputer dan sisi pemain manusia. Setiap sisi papan terdapat tujuh lubang kecil, satu lubang besar untuk penyimpanan dan 98 biji congklak. Lubang kecil akan diisi masing-masing tujuh buah pada awal permainan. Lubang besar merupakan tempat penyimpanan masing-masing pemain. Arah jalannya permainan berputar searah jarum jam seperti pada gambar 3. Permainan dilakukan secara bergiliran. Pada giliran pemain komputer akan digunakan algoritma Negascout dalam penentuan langkah yang akan diambil.



Gambar 3. Papan Permainan Congklak

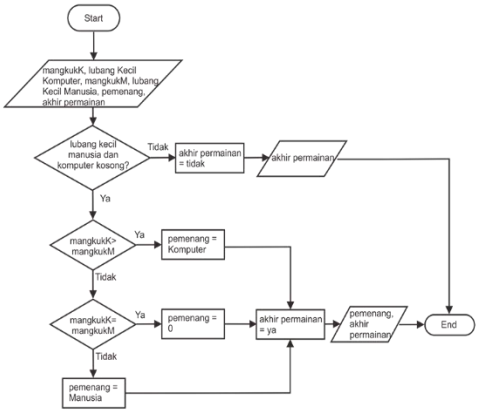
2.4.1 Analisis Game

Berikut ini adalah alur permainan secara keseluruhan berdasarkan aturan-aturan permainan congklak yang akan ditunjukan dalam *flowchart* pada gambar 4.



Gambar 4. Flowchart Permainan Congklak

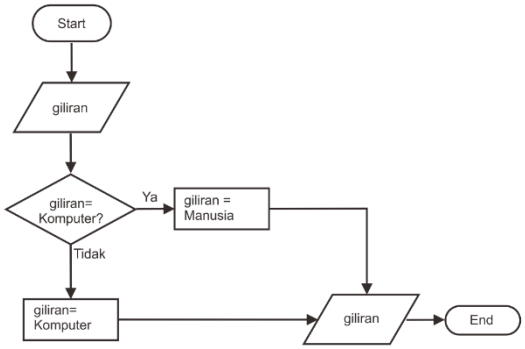
Untuk mengetahui akhir dari permainan maka diperlukan pemeriksaan pada lubang kecil papan permainan seperti dalam *flowchart* pada gambar 5. Jika kondisi akhir permainan dipenuhi, maka dapat ditentukan siapa pemenang permainannya.



Gambar 5. Flowchart Cek Akhir Permainan

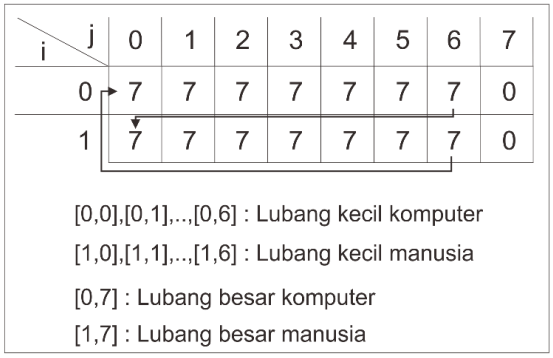
Jika kondisi akhir permainan tidak dipenuhi, maka langkah selanjutnya adalah menentukan

pemain yang akan bermain pada giliran selanjutnya seperti dalam *flowchart* pada gambar 6.



Gambar 6. Flowchart Ganti Giliran

Dalam penelitian kali ini papan akan direpresentasikan ke dalam sebuah matriks dua dimensi seperti pada gambar 7 di bawah ini.



Gambar 7. Matriks Papan Permainan

2.4.2 Analisis Algoritma

Tahap pertama yang dilakukan adalah pembangunan pohon pencarian atau kemungkinan-kemungkinan yang akan terjadi dalam jalannya permainan. Struktur pohon pencarian menggunakan general tree yaitu pohon yang setiap induk boleh memiliki anak lebih dari 2. Pohon yang dibuat terdiri dari 4 level. *Node* pada level 0 dan 2 berasal dari kemungkinan pergerakan manusia. *Node* pada level 1, 3 berasal dari kemungkinan pergerakan komputer.

Node pada pohon permainan di Gambar 8 memiliki beberapa variabel yaitu :

1. Nama *Node*

Nama *node* terdiri dari dua bagian, yaitu bagian kiri yang terdiri dari satu atau lebih digit angka dan bagian kanan yang hanya terdiri dari satu digit angka. Bagian kiri merupakan identitas dari induk *node* dan bagian kanan merpukanan identitas dari *node* itu sendiri.

2. Papan

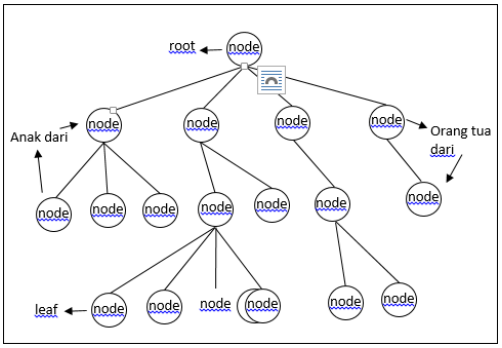
Papan ini merupakan array yang berisi informasi kondisi papan permainan pada kemungkinan yang terjadi.

3. Nilai evaluasi.

Nilai evaluasi adalah nilai yang akan dipakai pada proses pencarian solusi.

4. Anak

Variabel ini berisi *node* atau kemungkinan-kemungkinan yang bisa terjadi pada kondisi papan pada *node* ini.



Gambar 8. Contoh Pohon Permainan

Setelah pembuatan pohon langkah selanjutnya adalah menentukan nilai evaluasi dan proses pencarian solusi. Dalam penelitian ini akan dijelaskan cara menentukan nilai dan solusi menggunakan algoritma algoritma Negascout. Berikut adalah penjelasannya:

Pada algoritma Negascout cara membedakan kapan harus dilakukan pencarian minimum dan maksimum adalah dengan membuat negatif dari *node* anaknya dan dicari nilai maksimum. Pada saat kondisi mencari nilai maksimum maka nilai evaluasi dari *node* anaknya haruslah negatif nilai evaluasi, karena pada saat pengambilan nilai evaluasi nilainya akan dinegatifkan terlebih dahulu. Negatif dari negatif nilai evaluasi adalah nilai evaluasi, maka ketika dicari nilai maksimumnya akan didapatkan nilai maksimum dari nilai evaluasi. Pada saat kondisi mencari nilai minimum maka nilai evaluasi dari *node* anaknya haruslah positif nilai evaluasi, karena pada saat pengambilan nilai evaluasi nilainya akan dinegatifkan terlebih dahulu. Negatif dari positif nilai evaluasi adalah negatif nilai evaluasi, maka ketika dicari nilai maksimumnya akan didapatkan nilai minimum dari nilai evaluasi. Berdasarkan hal ini maka nilai evaluasi pada algoritma Negascout didapatkan dengan menggunakan rumus sebagai berikut :

Jika evaluasi nilai dilakukan pada giliran komputer(level ganjil) maka

E = - (mangkukK – mangkukM)

atau

E = mangkukM – mangkukK

Jika evaluasi nilai dilakukan pada giliran manusia(level genap) maka

E = mangkukK – mangkukM

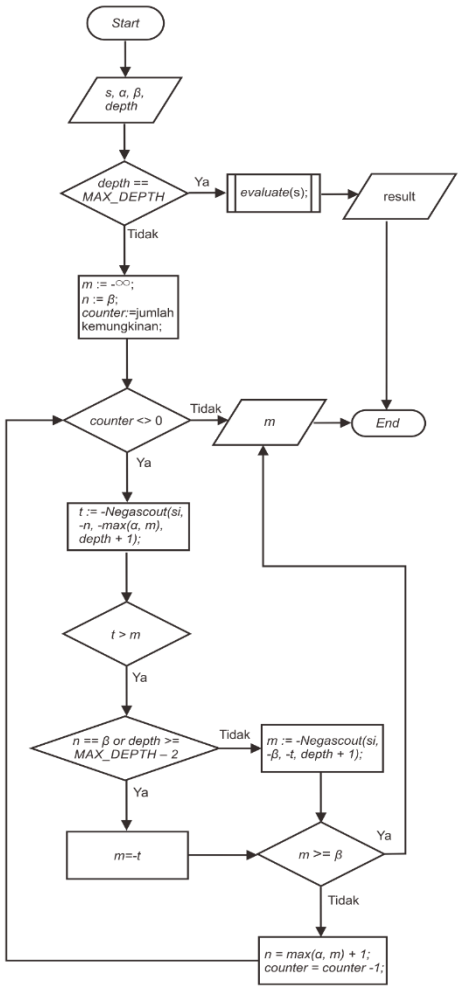
Keterangan :

mangkukK = jumlah biji dalam lubang besar komputer

mangkukM = jumlah biji dalam lubang besar manusia

E = nilai evaluasi

Tahap pencarian solusi menggunakan algoritma negascout yang bisa dilihat alur nya secara umum dalam *flowchart* pada gambar 9. Dalam pencariannya diasumsikan bahwa langkah pertama (*node* pertama pada level 1) yang diambil merupakan langkah terbaik, sedangkan sisanya merupakan langkah terburuk. Namun jika ternyata ada langkah yang lebih baik dari langkah pertama, maka akan terjadi proses *research* atau proses pencarian ulang.



Gambar 9. Flowchart Algoritma Negascout

Pada algoritma Negascout dari gambar 9, syarat untuk melakukan research adalah jika nilai dari t lebih besar dari m, lebih kecil dari *beta*, Syarat t lebih kecil dari beta diperlukan karena jika t lebih besar dari *beta*, maka *beta pruning* (pemotongan beta) akan diproses sehingga proses *research* tidak akan terjadi.

2.5 Pengujian Sistem

2.5.1 Pengujian Akurasi

Pengujian akurasi dilakukan dengan tujuan untuk mengetahui apakah perhitungan manual menghasilkan nilai yang sama dengan perhitungan pada program. Hasil pengujiannya dapat dilihat pada tabel 1.

Tabel 1. Pengujian Akurasi

No	Kondisi Papan									Hasil Perhitungan Manual		Hasil Perhitungan Program	
										Nilai Evaluasi	Lubang yang Dipilih	Nilai Evaluasi	Lubang yang Dipilih
1	28	3	3	3	3	3	3	3	28	8	[0,2]	8	[0,2]
		3	3	3	3	3	3	3					
2	34	0	4	0	0	4	4	0	30	4	[0,1]	4	[0,1]
		4	4	0	4	5	5	0					
3	37	3	2	3	2	0	1	2	37	2	[0,1]	2	[0,1]
		1	2	3	0	2	1	2					
4	39	0	3	4	0	1	2	0	38	2	[0,1]	2	[0,1]
		1	0	3	0	2	2	3					
5	42	0	0	5	0	0	3	0	42	2	[0,2]	2	[0,2]
		0	0	3	0	3	0	0					

Dari hasil pengujian akurasi didapatkan bahwa nilai evaluasi dan lubang yang dipilih dari setiap hasil perhitungan manual sesuai dengan hasil perhitungan pada program. Maka dapat disimpulkan bahwa nilai akurasi dari pengujian ini adalah 100%

2.5.2 Pengujian Kinerja

Pengujian kinerja dilakukan dengan tujuan untuk mengetahui sejauh mana kualitas dari perangkat lunak yang dibangun, apakah sudah sesuai dengan harapan atau belum. Dalam pengujian kinerja akan ada dua tahap yaitu pengujian kinerja 1 dan pengujian kinerja 2. Pada pengujian kinerja 1 pengujian akan dilakukan dalam 21 kali pertandingan dengan 3 tingkatan level (kedalaman pencarian). Tiap satu level akan digunakan untuk 7 kali pertandingan. Pada pengujian kinerja 2 pengujian akan dilakukan untuk melihat waktu proses pencarian dan jumlah langkah per satu kali giliran permainan.

a. Pengujian Kinerja 1

Hasil pengujian pada pengujian kinerja 1 adalah sebagai sebagai berikut:

1. Pengujian pada tingkat kedalaman pencarian sampai level ke-2

Hasil pengujian pada tingkat kedalaman pencarian sampai level ke-2 dapat dilihat pada tabel 2.

Tabel 2. Hasil Pengujian pada Tingkat Kedalaman Pencarian Level 2

No	Jumlah Langkah Pencarian	Jumlah Waktu Proses Pencarian (dalam milidetik)	Giliran Pertama	Pemenang	Skor (Komputer-Manusia)
1	253	3	Komputer	Manusia	44-54
2	278	4	Manusia	Manusia	32-66
3	235	3	Komputer	Seri	49-49
4	415	5	Manusia	Manusia	43-55
5	301	4	Komputer	Komputer	54-44
6	351	5	Manusia	Komputer	50-48
7	275	4	Komputer	Komputer	64-34

2. Pengujian pada tingkat kedalaman pencarian sampai level ke-3

Hasil pengujian pada tingkat kedalaman pencarian sampai level ke-3 dapat dilihat pada tabel 3.

Tabel 3. Hasil Pengujian pada Tingkat Kedalaman Pencarian Level 3

No	Jumlah Langkah Pencarian	Jumlah Waktu Proses Pencarian (dalam milidetik)	Giliran Pertama	Pemenang	Skor (Komputer-Manusia)
1	1327	28	Komputer	Komputer	62-36
2	558	11	Manusia	Manusia	38-60
3	1402	42	Komputer	Komputer	56-42
4	1742	29	Manusia	Manusia	45-53
5	1329	19	Komputer	Komputer	66-32
6	1229	26	Manusia	Komputer	58-40
7	931	24	Komputer	Komputer	53-45

3. Pengujian pada tingkat kedalaman pencarian sampai level ke-4

Hasil pengujian pada tingkat kedalaman pencarian sampai level ke-4 dapat dilihat pada tabel 4.

Tabel 4. Hasil Pengujian pada Tingkat Kedalaman Pencarian Level 4

No	Jumlah Langkah Pencarian	Jumlah Waktu Proses Pencarian (dalam milidetik)	Giliran Pertama	Pemenang	Skor (Komputer-Manusia)
1	3162	47	Komputer	Komputer	58-40
2	2864	49	Manusia	Manusia	48-50
3	2255	48	Komputer	Seri	49-49
4	2622	33	Manusia	Komputer	50-48
5	2905	53	Komputer	Komputer	53-45
6	1342	27	Manusia	Komputer	58-40
7	3909	85	Komputer	Manusia	48-50

Setelah seluruh pengujian kinerja 1 didapatkan persentase tingkat kemenangan keseluruhan pada tabel 5 berikut:

Tabel 5. Presentasi Tingkat Kemenangan

Level	Komputer	Manusia	Seri
2	3	3	1
3	5	2	0
4	4	2	1
Jumlah	12	7	2
Persentase	57%	33%	10%

Kesimpulan dari pengujian kinerja satu adalah penggunaan algoritma negascout pada game congklak memiliki tingkat kemenangan dari 21 pertandingan adalah sebesar 57%. Pada penelitian Citra Anindya [2] tingkat kemenangan algoritma minimax pada game congklak dari 21 pertandingan adalah sebesar 71%. Hal ini menunjukan bahwa algoritma minimax lebih baik dari algoritma negascout dalam tingkat kemenangan.

b. Pengujian Kinerja 2

Hasil pengujian pada kinerja 2 dapat dilihat pada tabel 6.

Tabel 6. Hasil Pengujian Kinerja 2

No	Kondisi Papan									Minimax		Negascout	
										Waktu Proses (detik)	Jumlah Langkah	Waktu Proses (milidetik)	Jumlah Langkah
1	28	3	3	3	3	3	3	3	28	27,8301	430	1,6729	162
		3	3	3	3	3	3	3					
2	34	0	4	0	0	4	4	0	30	3,4367	120	0,489	43
		4	4	0	4	5	5	0					
3	37	3	2	3	2	0	1	2	37	9,976	296	1,3771	126
		1	2	3	0	2	1	2					
4	39	0	3	4	0	1	2	0	38	3,4988	124	0,5922	61
		1	0	3	0	2	2	3					
5	42	0	0	5	0	0	3	0	42	1,1727	36	0,1633	21
		0	0	3	0	3	0	0					
Jumlah										45,9143	1006	4,2945	413

Keterangan :

Waktu proses dan jumlah langkah pencarian algoritma minimax didapatkan dari hasil pengujian pada penelitian Sarah Nurhasanah [3].

Dari hasil pengujian kinerja 2 didapatkan bahwa waktu proses pencarian dan jumlah langkah pencarian algoritma negascout pada setiap pengujian lebih kecil daripada algoritma minimax. Perbandingan jumlah keseluruhan langkah pencariannya adalah 2:5 dan perbandingan keseluruhan waktu proses pencariannya adalah 1:10691,42. Hal ini menunjukan bahwa algoritma negascout lebih cepat daripada algoritma minimax.

c. Kesimpulan Pengujian Kinerja

Dari hasil pengujian jika dilihat dari segi tingkat kemenangan algoritma minimax lebih unggul dari algorima negascout dengan presentasi kemenangan 71% untuk minimax dan 57% untuk negascout, sedangkan jika dilihat dari segi kecepatan algoritma negascout lebih baik dari algoritma minimax dengan perbandingan jumlah keseluruhan langkah pencariannya 2:5 dan perbandingan keseluruhan waktu proses pencariannya 1:10691,42.

3. PENUTUP

3.1 Kesimpulan

Berdasarkan analisis yang telah dilakukan terhadap pembangunan aplikasi game congklak yang menerapkan algoritma Negascout, maka dapat disimpulkan bahwa kinerja algoritma Negascout dapat diketahui dengan hasil sebagai berikut:

a. Akurasi dari perhitungan manual dan perhitungan oleh program adalah 100%.
b. Tingkat kemenangan algoritma Minimax lebih unggul dari algorima Negascout dengan presentasi kemenangan 71% untuk Minimax dan 57% untuk Negascout.
c. Tingkat kecepatan algoritma Negascout lebih baik dari algoritma Minimax dengan perbandingan jumlah keseluruhan langkah pencariannya 2:5 dan perbandingan keseluruhan waktu proses pencariannya 1:10691,42.

3.2 Saran

Agar aplikasi ini dapat berkembang ke depannya, penulis menyarankan beberapa hal sebagai berikut yaitu:

a. Menambahkan parameter evaluasi selain selisih jumlah biji pada lubang besar untuk menambahkan kemungkinan tingkat keakuratan proses pencarian solusi.
b. Menggunakan algoritma yang lebih baik dari algoritma Minimax dan Negascout.

DAFTAR PUSTAKA

[1] G. Hermawan, "Implementasi Algoritma Greedy Best First Search pada Aplikasi Permainan Congklak untuk Optimasi Pemilihan Lubang dengan Pola Berfikir Dinamis," dalam *Seminar Nasional Teknologi Informasi dan Multimedia*, Surabaya, 2012.

[2] C. Anindya, "Penerapan Kecerdasan Buatan pada Permainan Tradisional Congklak Menggunakan Optimasi Algoritma Minimax," Skripsi. Universitas Pendidikan Indonesia, Bandung, 2011.

[3] S. Nurhasanah, "Analisis Algoritma Mimimax Optimasi Alpha-Beta Pruning terhadap Waktu Komputasi Pada Game Congklak," Skripsi. Universitas Komputer Indonesia, Bandung, 2013.

[4] J. Mandziuk, Knowledge-Free and Learning-Based Methods in Intelligent Game Playing, Springer Science & Business Media, 2010.

[5] "Congklak, A Traditional Game of Indonesia," Expat Web Site Association Jakarta, [Online]. Available: http://www.expat.or.id/info/congklak.html. [Diakses 12 02 2015].

[6] Suyanto, Artificial Intelligence: Searching, Reasoning, Planning, Bandung: Informatika, 2007.

[7] M. Jones, Artificial Intelligence : A System Approach, Massachusetts: Infinity Science Press LLC, 2008.