PEMODELAN DAN PENGIMPLEMENTASIAN PERMAINAN *CONNECT FOUR*

Andrew Mahisa Halim[1], Frederikus Judianto[1], Samuel Lukas[1], Petrus Widjaja[2]

[1]Teknik Informatika, Universitas Pelita Harapan, Lippo Karawaci, Tangerang
[2]Matematika, Universitas Pelita Harapan, Lippo Karawaci, Tangerang
andrewmahisa@gmail.com

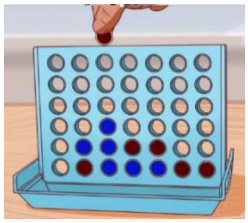
Abstrak . *Permainan Connect Four adalah salah satu permainan logika yang menggunakan papan permainan. Connect Four dimainkan oleh dua pemain yang bergerak secara bergantian. Papan permaianan terdiri dari tujuh kolom dan enam baris. Setiap pemain memasukkan disk warnanya ke salah satu kolom dan disk akan menempati satu baris diatas disk yang telah ada di kolom itu. Pemain dikatakan menang apa bila ia telah menyusun empat disknya secara berurutan baik secara horisontal, vertikal ataupun diagonal. Paper ini membahas bagaimana memodelkan permaianan ini dan mengimplementasikan model permainan itu dengan menggunakan algoritma minimax dan alpha-beta pruning dari pohon pencarian dengan kedalaman satu hingga delapan. Diakhir bahasan juga akan diujikan dengan melawan sistem cerdas yang sudah ada. System mampu mengalahkan sistem cerdas lainnya lebih baik dengan kedalaman yang semakin tinggi.*

***Kata kunci**: Connect Four, Sistem Cerdas, Algoritma Minimax, Alpha-Beta pruning.*

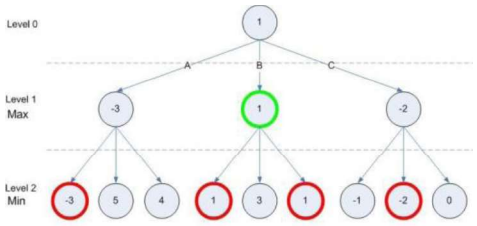
1. Pendahuluan

Four in Line adalah suatu permainan untuk beradu strategi antara 2 orang pemain dan tergolong pada permainan *zero-sum game*. Permainan pertama kali di keluarkan dan di tampilkan oleh Milton Bradley pada tahun 1974 di bawah trademark perusahaan dan nama "Connect 4". James Dow Allen dan Victor Allis pada Oktober 1988 pernah menganalisis permainan ini secara menyeluruh dan tetap menjadi permainan yang mengasyikan sampai sekarang [1]. Papan permainan berdiri dan terdiri dari 42 lubang dalam 6 baris dan 7 kolom. Papan memiliki celah diatasnya untuk menjatuhkan *disc*, Gambar 1. Setiap pemain memiliki *Disc* dengan warna berbeda. Ukurannya *disk* sedikit lebih besar dari ukuran lubang di papan, sehingga *disk* akan menutupi lubang yang ada. Masing masing pemain mempunyai 21 *disk.* Pemain akan memenangkan permainan apabila ia berhasil menempatkan 4 *disk* secara berurutan baik secara vertikal (kolom), horizontal (baris), ataupun diagonal terlebih dahulu setelah itu permainan selesai.

Banyak langkah yang mampu tercipta dalam sebuah permainan yang sempurna, tersisa 1 lubang dalam papan yang belum di isi, maka, akan ada sebanyak 4,531,985,219,092 cara yang bisa kita lakukan, terdapat 1,904,587,136,600 cara untuk kita bisa menang, dan terdapat 713,298,878 cara agar permainan menjadi seri draw [1]. Hal ini membuat tantangan tersendiri untuk dapat mengkomputerisasikan game sehingga komputer dapat mencari "jalan" terbaik dalam setiap langkah pada permainan ini.



Gambar 1. Papan Permainan connect four



Gambar 2. Pohon keputusan MinMax

Stefan and Peter [2] menyatakan bahwa meskipun permainan *Conect Four* memiliki kompleksitas tinggi namun dapat dimodelkan dengan Binary Decision Diagram (BDDs). Sementara itu Victor [3] menyimpulkan bahwa dua pemain terlatih untuk *game* ini, maka pemain pertama memiliki kecenderungan menang dibanding dengan pemain kedua. Pemain pertama mempunyai langkah

winning strategy. Mohamed F. abdelsadek [4] mencoba menggunakan algoritma genetik dalam menyelesaikan Connect four game ini. Kirk dkk [5] membuat projek yang memampukan komputer berfikir dalam permainan ini.
Makalah ini akan memodelkan dan mengimplementasikan permainan connect four dengan menerapkan algoritma minimax dan menerapkan alpha-beta pruning untuk mempercepat proses perhitungan, kemudian mengujinya dengan melawan aplikasi program lainnya diakhir bahasan ini.

Pemodelan Permainan
Permainan *connect four*, direpresentasikan sebagai sebuah matrix $A, a[i,j]$, berukuran 6×7 yang di indexkan sesuai (1) dan setiap elemen matrix, $a[i,j]$, dinyatakan pada persamaan (2)

$$A = \begin{pmatrix} a_{16} & a_{26} & a_{36} & a_{46} & a_{56} & a_{66} & a_{76} \\ a_{15} & a_{25} & a_{35} & a_{45} & a_{55} & a_{65} & a_{75} \\ a_{14} & a_{24} & a_{34} & a_{44} & a_{54} & a_{64} & a_{74} \\ a_{13} & a_{23} & a_{33} & a_{43} & a_{53} & a_{63} & a_{73} \\ a_{12} & a_{22} & a_{32} & a_{42} & a_{52} & a_{62} & a_{72} \\ a_{11} & a_{21} & a_{31} & a_{41} & a_{51} & a_{61} & a_{71} \end{pmatrix} \quad (1)$$

$$a[i,j] = f(i,j) = \begin{cases} 1 & untuk\ pemain\ 1 \\ -1 & untuk\ pemain\ 2 \\ 0 & kosong \end{cases} \quad (2)$$

Pemain pertama dikatakan menang apabila nilai $z = 4$ sedangkan pemain kedua dikatakan menang apabila $z = -4$. Nilai z adalah jumlah empat buah *disk* pada papan secara berurutan baik secara baris, kolom, diagonal kiri atau diagonal kanannya. Perhitungan nilai z diperlihatkan pada persamaan (3), (4), (5) dan (6)

$$\exists(k,y), k \in \{1,2,3,4\}, y \in \{1,2,\dots,6\} \rightarrow z = \sum_{x=k}^{k+3} a[x,y] \quad (3)$$
$$\exists(x,k), x \in \{1,2,\dots,7\}, k \in \{1,2,3\} \rightarrow z = \sum_{y=k}^{k+3} a[x,y] \quad (4)$$
$$\exists(x,y), x \in \{1,2,3,4\}, y \in \{4,5,6\} \rightarrow z = \sum_{i=0}^{3} a[x+i, y-i] \quad (5)$$
$$\exists(x,y), x \in \{1,2,3,4\}, y \in \{1,2,3\} \rightarrow z = \sum_{i=0}^{3} a[x+i, y+i] \quad (6)$$

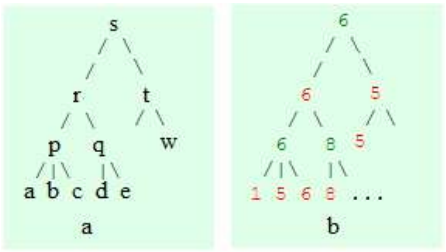
Ferguzon [6] mengatakan *Zero-sum game* adalah sebuah representasi matematis atas sebuah kondisi yang menghasilkan kemungkinkan efek yang berlawanan bagi pemain. Ini berarti keuntungan bagi satu pemain adalah kerugian yang sama bagi pemain lain. *Connect four*, adalah salah satu contoh game yang memiliki properti *zero-sum*. Ada berbagai macam metode yang dapat digunakan untuk menyelesaikan *Zero-sum game* yang terdiri dari 2 player, salah satunya adalah game-tree search dengan mengunakan algoritma minimax

Minimax Algorithm
Algoritma minimax [7] merupakan salah satu algoritma yang sering digunakan untuk *game* berbasis kecerdasan buatan yang menggunakan teknik *depth first search* (DFS) dalam pencarian-nya pada pohon dengan kedalaman terbatas. Algoritma minimax digunakan untuk memilih langkah terbaik. Algoritma ini bekerja secara rekursif dan mendeskripsikan kondisi apabila terdapat pemain yang mengalami keuntungan, pemain lain akan mengalami kerugian senilai dengan keuntungan yang diperoleh lawan dan sebaliknya.
Algoritma minimax akan melakukan pengecekan pada seluruh kemungkinan yang ada, sehingga akan menghasilkan pohon permainan yang berisi semua kemungkinan permainan tersebut. Dengan pohon permainan ini setiap pemain mengetahui langkah-langkah yang mungkin diberikan pada situasi permainan saat ini. Sehingga untuk setiap langkah dan semua langkah selanjutnya dapat diketahui. Dalam representasi pohon pada algoritma minimax, terdapat dua jenis simpul, yaitu simpul min dan simpul max, [8]. Max akan memilih langkah dengan nilai tertinggi dan min akan memilih langkah dengan nilai terendah. Dalam penentuan keputusan maxmin tersebut dibutuhkan suatu nilai yang merepresentasikan kerugian atau keuntungan untuk itu dibutuhkan sebuah fungsi heuristik.
Langkah pertama algoritma ini adalah membentuk pohon keputusan seperti diperlihatkan Gambar 2. Pemain memiliki tiga kemungkinan langkah yaitu A,B dan C. Jika pemain melangkah A, maka lawan memiliki tiga kemungkinan langkah lainnya. Pada gambar 2, Pohon keputusan memiliki kedalaman dua tingkat. Langkah ke dua Minmax adalah menentukan nilai hiuristik pada setiap ”daun” dari pohon

keputusan. Langkah ketiga adalah menentukan level max dan level min pada pohon keputusan. Level max adalah level ganjil sedangkan level min adalah level genap. Untuk gambar 2 maka level max adalah level 1 dan level min adalah level 0 dan 2.
Nilai heuristik *parent* dari seluruh *child* ditentukan dari nilai max atau min dari nilai heuristik *child*. Nilai heuristik *parent* pada level max adalah nilai minimum dari seluruh nilai heuristik *child* nya. Sedangkan nilai heuristik *parent* pada level min adalah nilai maximum dari seluruh nilai heuristik *child* nya, [9]. Untuk gambar 2 maka nilai heuristik parent pilihan A adalah -3 karena ini adalah nilai minimum dari ketiga nilai heuristik *child* nya. Nilai heuristik pilihan B adalah 1 karena ini adalah nilai minimum dari tiga nilai heuristik *child* nya. Demikian juga nilai heuristik pilihan C adalah -2. Nilai heuristuk akar adalah nilai maximum dari ketiga *child* nya pada level 1 yaitu $\{-3, 1, -2\}$. Jadi nilai heuristik akar adalah 1. Ini artinya pemain akan melangkah B dengan keuntungan sedikitnya adalah 1. Sedangkan kalau dia melangkah A maka pemain akan memiliki keuntungan -3 yang artinya dia cenderung mengalami kerugian 3.
Terlihat semakin dalam pohon pencarian maka kecerdasan sistem akan semakin tinggi namun ini membutuhkan banyak perhitungan karena ada banyak nilai heuristik daun yang harus dihitung. Kalau setiap langkah ada 7 kemungkinan jalan maka untuk kedalaman 8 akan dibutuhkan perhitungan nilai heuristik daunnya sebanyak $7^8 = 5.764.801$ perhitungan. Untuk ini dibutuhkan algoritma Alpha-Beta Pruning

Alpha Beta Pruning

Alpha Beta Pruning [8] adalah algoritma untuk mengurangi banyak perhitungan nilai heuristik daun baik baik *child* maupun *parent* dari minmax algorithm. Gambar 3 merupakan contoh pohon keputusan. Setelah dihitung $h(a) = 1$, maka $h(p) = 1$. Kemudian dihitung $h(b) = 5$ dan $h(c) = 6$ maka nilai $h(p)$ tetap yaitu $h(p) = 1$. Ini mengakibatkan $h(r) = 6$. Setelah menghitung $h(d) = 8$ ini mengakibatkan $h(q) = 8$. Ini akan tetap menjadikan $h(r) = 6$. Jadi jika $h(e) > 8$ maka $h(q) > 8$ dan ini mengakibatkan $h(r) = 8$ atau jika $h(e) < 8$ maka nilai $h(q) = 8$ dan nilai $h(r) = 6$ hal ini artinya tidak perlu menghitung nilai $h(e)$. $h(r) = 6 \rightarrow h(s) = 6$ dan jika $h(t) = 5$ maka $h(w)$ tidak perlu dihitung. Ini akan menghemat waktu perhitungan karena child dari $h(w)$ juga tidak perlu dihitung nilai heuristiknya.



Gambar 3. Pohon keputusan
Source: https://www.cs.cornell.edu/courses/cs312/2002sp/lectures/rec21.htm

Pseudocode yang mengimplementasikan Minimax Beserta Alpha-beta pruning secara bersamaan termuat dalam tabel dibawah ini [4]:

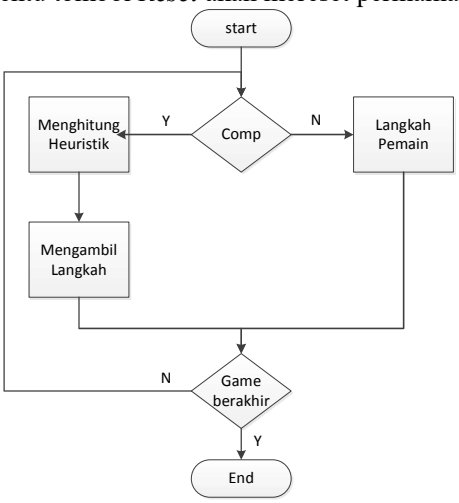
```
function alphabeta(node, depth, α, β)
02    if depth = 0 or node is a terminal node
03        return the heuristic value of node
04        v := -∞
05        for each child of node
06            v := max(v, alphabeta(child, depth – 1, α, β, FALSE))
07            α := max(α, v)
08            if β ≤ α
09                break (* β cut-off *)
10        return v
11
```

2. Pembahasan

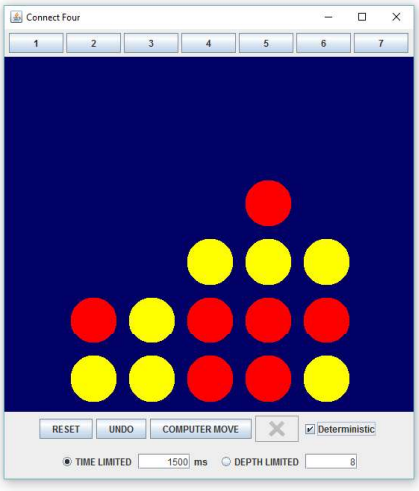
Perancangan Sistem

Sistem dirancang terdiri dari tiga bagian utama, Gambar 4, yaitu perhitungan nilai heuristik setiap daun yang terbentuk pada proses pembentukan pohon keputusan dengan teknik DFS. Kedua pegambilan langkah terbaik dan proses penerimaan input dari langkah pemain lawan. Proses perhitungan nilai heuristik para parent lainnya setelah daun dilakukan dengan menggunakan algoritma minmax yang diperlengkapi dengan algoritma alpha-beta pruning.

Tampilan user interface dari system yang dibangun diperlihatkan pada Gambar 5. Sebelum melangkah, sistem akan menerima masukan berupa lama waktu untuk setiap langkah yang akan dilakukan komputer yang dinyatakan dalam *time limited* kemudian *depth limited* untuk membatasi pencarian kedalaman maksimum hingga ke delapan. Apabila tombol *computer move* di *click* maka ini mengindikasikan komputer sebagai pemain pertama. Ini menandakan komputer akan melakukan langkah awal. Jika tombol *computer move* tidak di *click* maka komputer bertindak sebagai pemain ke dua. Jika *deterministic* di centang maka langkah komputer selalu pada nilai heuristik optimum pada kolom terkecil sedangkan jika tidak maka langkah komputer adalah menurut nilai heuristik optimum yang diacak diantara nilai heuristik optimum itu. Jika tombol *Undo* di *click* maka pemain lawan bisa membatalkan langkah terakhir yang dipilih dan kembali kekeadaan sebelum pemain melangkah. Sudah tentu tombol *Reset* akan mereset permainan ke permainan awal



Gambar 4: Perancangan umum sistem yang dibangun



Gambar 5: Tampilan implementasi sistem

Perhitungan nilai heuristik setiap daun dilakukan dengan persamaan (7)

$$f(x_1, x_2, x_3, x_4) = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 \quad (7)$$

Variabel x_1 hingga x_4 masing-masing menyatakan banyaknya garis kemenangan, banyaknya disc yang ada pada garis kemenangan, banyaknya disk lawan yang ada pada garis kemenangan lawan dan langkah kemenangan atau kekalahan. Sedangkan a_1 sampai a_4 adalah konstanta yang dipilih dimana $a_1 = 1$, $a_2 = 200$, $a_3 = 100$ dan $a_4 = 1000$. Besaran konstanta ini juga menentukan karakteristik kecerdasan sistem yang dirancang. Jika nilai $a_2 < a_3$ sistem cenderung bermain bertahan.

Proses pengambilan langkah dilakukan dengan algoritma minimax yang diperlengkapi dengan alpha-beta pruning. Apabila nilai heuristik *root* sama dengan beberapa nilai heuristik *child* satu level dibawahnya maka pengambilan langkah dilakukan pada kolom terkecil apabila pilihan *deterministic* dipilih jika pilihan *deterministic* tidak dipilih maka kolom yang dipilih adalah kolom dari hasil acak kolom dengan nilai heuristik child yang sama dengan nilai heuristik *root*.

Langkah pemain kedua dilakukan dengan memilih salah satu tombol 1 sampai dengan 7 yang ada diatas *background* biru kemudian *disk* kuning akan meluncur dari atas ke kolom yang dipilih dan berhenti tepat diatas disk sebelumnya yang ada pada kolom itu.

Pengujian Sistem

Untuk melakukan Analisis keandalan sistem, dilakukan percobaan dengan menandingkan beberapa kali dengan game yang sama yang sesuai dengan kedalaman pencarian yang sama yang dipublikasikan pada website https://rmarcus.info/blog/2014/12/23/connect4.html. Setiap permainan selalu dimulai

dengan sistem yang dirancang dikarenakan keterbatasan dari game connect four milik lawan yang tidak bisa melakukan gerakan pertama. Hasil pengujian ditampilkan pada Tabel 1.
Setiap baris menandakan depth dari A.I. lawan, dan setiap kolom menandakan depth dari sistem. Kotak yang berwarna merah menandakan kemenangan dari sistem, sedangkan kotak berwarna kuning menandakan kemenangan lawan. Kotak berawarna hijau menandakan permainan berakhir dengan keadaan seri. Setiap kotak berisi huruf W untuk (menang), L untuk (kalah) atau D (seri), diikuti dengan jumlah putaran yang diambil untuk menyelesaikan permainan.

Tabel 1: Hasil pengujian sistem

Depth	1	2	3	4	5	6	7	8
1	W (7)	W (7)	W (7)	W (7)	W (7)	W (7)	W (7)	W (7)
2	D (42)	L (28)	L (40)	L (40)	L (38)	W (17)	W (7)	W (7)
3	L (14)	L (38)	L (30)	L (22)	L (30)	W (39)	W (7)	W (7)
4	L (22)	L (34)	D (42)	W (35)	W (33)	W (37)	W (35)	D (42)
5	L (30)	W (35)	L (42)	W (33)	W (35)	W (37)	W (39)	W (19)
6	L (26)	D (42)	L (42)	L (40)	W (31)	W (31)	W (27)	W (29)
7	L (32)	L (30)	L (36)	L (34)	L (34)	W (31)	D (42)	W (35)
8	L (22)	W (35)	L (42)	W (33)	W (35)	W (31)	W (33)	W (31)

Terlihat depth = 1 semua kotak berwarna merah dan dengan label W(7). ini mengindikasikan bahwa untuk setiap kedalaman sistem yang dirancang dapat memenangkan permainan hanya dalam 7 langkah. Jadi sistem mendapatkan kemenangan sebanyak 36 kali permainan, kekalahan sebanyak 23 kali, dan permainan berakhir dalam keadaan seri sebanyak 5 kali.
Banyak kasus dimana sistem mengalahkan lawan pada depth yang lebih tinggi. Ini adalah hal yang wajar terjadi, dikarenakan kecerdasan sistem yang dirancang sebanding dengan jumlah depth yang digunakan. Rata-rata langkah kemenangan sistem dari 36 kali kemenangan adalah 23.89 langkah. Namun terdapat juga kasus dimana sistem dapat dikalahkan, meskipun dengan depth lawan yang lebih rendah. Akan tetapi dari 23 kekalahan sistem, rata-rata langkah kemenangan lawan adalah 32.30. Ini mengindikasikan sistem cenderung melakukan langkah penyerangan dibandingkan bertahan sementara sistem lain mungkin cenderung bertahan. Sistem mampu mengalahkan lawan secara lebih konsisten pada depth yang lebih tinggi, yaitu depth 6 s/d 8
Terdapat pula faktor lain yang menyebabkan lebih banyak kemenangan yang didapatkan dari sistem dibandingkan dengan lawan, yaitu urutan gerakan. Dalam semua data yang kami ambil, sistem selalu mengambil putaran pertama. Dan putaran pertama menjadi kunci untuk memenangkan sebuah permainan *connect four*.

5. Simpulan

Dari hasil penelitian dan eksperimen, dapat ditarik kesimpulan beberapa hal berikut, yaitu:

1. Semakin dalam pohon pencarian, semakin besar juga kemungkinan untuk menang. Ini dikarenakan evaluasi minimax dalam DFS nya jauh lebih akurat dibandingkan dengan yang menggunakan depth rendah.
2. Depth bukan satu-satunya faktor yang dapat menentukan cerdas tidaknya suatu sistem, ini juga tergantung dari fungsi heuristik yang dipakai.

Daftar Pustaka

[1] https://en.wikipedia.org/wiki/Connect_Four, diakses 15 Desember 2016
[2] Stefan Edelkamp and Peter Kissmann, 2011, "*On the Complexity of BDDs for State Space Search: A Case Study in Connect Four*", Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence.
[3] Victor Allis, 1988, A Knowledge-based Approach of Connect-Four, The Game is Solved: White Wins, Master thesis, Department of Mathematics and Computer Science, Vrije Universiteit Amsterdam, Netherlands.
[4] Mohamed F. Abdelsadek, Using Genetic Programming to Evolve a Connect-4 game player, Department of Computer Science, Columbia University.
[5] Kirk Baly, Andrew Freeman, Andrew Jarratt, Kyle Kling, Owen Prough, Greg Hume, 2012, *Teaching Computers to Think: Automated Analysis of Connect Four*, Research in Computer Science (CS 492).
[6] Ferguson, T. S., & Games, P. I. I. T. Z. (n.d.). Game theory.
[7] Kusumadewi Sri, 2003, *Artificial Intelligence (Teknik dan Aplikasinya)*. Graha Ilmu. Yogyakarta.
[8] B. Berwick, 2011, "Game trees, minimax, & alpha-beta search", Massachusetts Institute Of Technology, pp. 2-3, 2011.
[9] MiniMax search and Alpha-Beta Pruning", *cs.cornell.edu*, 2002. [Online]. Available: https://www.cs.cornell.edu/courses/cs312/2002sp/lectures/rec21.htm. Diakses 18 /11/ 2016