Pemodelan Permainan *Connect Four* menggunakan Algorithma Genetik dengan Algoritma Minimax

Andrew Mahisa Halim, Frederikus Judianto, Andre Wirawan, Samuel Lukas, Petrus Widjaja
Teknik Informatika, Fakultas Ilmu Komputer, Universitas Pelita Harapan
UPH Tower, Lippo Karawaci, Tangerang, 15811,
Samuel.lukas@uph.edu

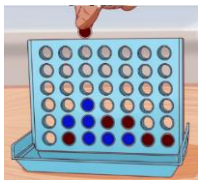
Abstrak

Connect Four adalah sebuah permainan logika menggunakan papan sederhana yang populer. Paper ini mengungkapkan pemodelkan kecerdasan buatan yang mampu bertindak sebagai pemain lawan dalam permainan Connect Four. Kecerdasan buatan diimplementasikan menggunakan Genetic Algorithm (GA), dengan metode heuristik menerapkan Algoritma Minimax. Proses seleksi dalam GA menggunakan 2 metode, yaitu metode better half dan metode stochastic universal sampling. Penelitian juga membandingkan kecerdasan algoritman hibrid ini dengan kecerdasan yang hanya menggunakan Algoritma Minimax. Hasilnya memperlihatkan untuk kedalaman pencarian yang sama, kecerdasan keduanya relatif sama namun algoritma yang dikembangkan memberikan variasi yang lebih baik. Sehingga dapat dikatakan pemodelan yang dihasilkan mampu bertindak sebagai pemain lawan dalam connect four yang baik, yang memiliki variabilitas yang cukup tinggi dikarenakan sifat acak dari Genetic Algorithm.

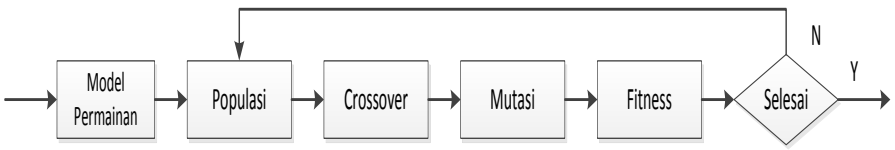
Kata kunci: *Algortima Genetik, Algoritma Minimax, Connect Four*

1. Pendahuluan

Connect Four merupakan sebuah permainan logika menggunakan papan sederhana yang dimainkan oleh 2 orang pemain. Papan permainan berdiri dan terdiri dari 42 lubang dalam 6 baris dan 7 kolom. Disetiap kolom pada papan ada celah untuk menjatuhkan *disc*, Gambar 1. Setiap pemain memiliki 21 *Disc* dengan warna sama namun setiap pemain warnanya berbeda. Ukurannya *disk* sedikit lebih besar dari ukuran lubang di papan, sehingga *disk* akan menutupi lubang yang ada. Dalam permainan ini, kedua pemain secara bergantian memasukkan satu *disk* ke lubang pada papan. 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.



Gambar 1.
Papan Permainan



Gambar 2. Blok Diagram Sistem

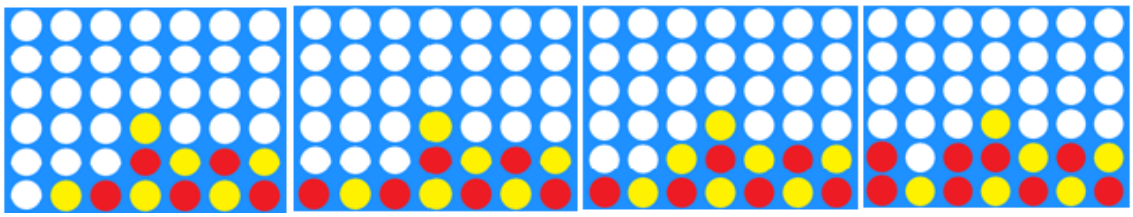
Permainan termasuk *zero-sum game*. Algoritma *Minimax* dalam *Game Tree* seringkali digunakan untuk memecahkan permasalahan ini. Stefan and Peter [1] menyatakan bahwa meskipun permainan *Conect Four* memiliki kompleksitas tinggi namun dapat dimodelkan dengan Binary Decision Diagram (BDDs). Sementara Victor [2] 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 [3] mencoba menggunakan algoritma genetik dalam menyelesaikan Connect four game ini. Kirk dkk [4] membuat projek yang memampukan komputer berfikir dalam permainan ini. Paper ini akan membahas mengenai implementasi kecerdasan buatan dalam permainan *Connect Four* menggunakan *Genetic Algorithm*, dengan algoritma *Minimax* sebagai metode Heuristik untuk mengatasi sifat acak yang muncul.

2. Metode Penelitian

Model permainan seperti yang telah dirancang pada [5]. Namun ditambahkan dengan menerapkan Algoritma Genetika dan Algoritma Minimax, yang digambarkan pada blok diagram sistem, Gambar 2.

2.1. Perancangan Kromosom pada Algoritma Genetika

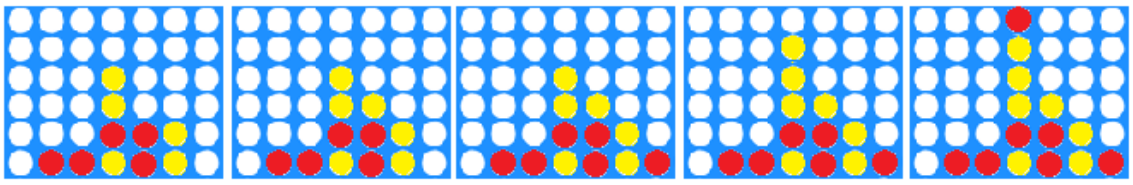
Kromosom adalah sederetan angka inklusif dari 1 sampai dengan 7 dengan panjangnya sama dengan kedalaman level tree yang akan digunakan. Sistem dirancang dapat menerima kedalaman tree sebanyak 13 level. Setiap digit angka merepresentasikan peletakan *disk* pada kolom. Kolom paling kiri dilambangkan dengan 1 sedangkan angka 7 merepresentasikan peletakan *disk* pada kolom paling kanan. Digit ganjil dari suatu kromosom menyatakan langkah pemain komputer yang dilakukan secara acak, sedangkan digit genap adalah langkah lawan. Langkah lawan dihasilkan secara heuristik sesuai algoritma minimax dengan level=1. Gambar 3 memperlihatkan langkah suatu kromosom yang terdiri dari 3 level dengan kromosom 131.



(a) (b) (c) (d)

Gambar 3. Peletakan disk pada kromosom 131. Gambar (a) keadaan awal sebelum peletakan *disk* pertama komputer. Gambar (b), *disk* merah (komputer) diletakan pada kolom 1 diikuti *disk* lawan (kuning) pada kolom 3, pada Gambar (c) dan *disk* kedua komputer pada kolom ke 1, Gambar (d)

Pembentukan lengkap suatu kromosom dilakukan dengan algoritma heurisik berikut. Generate digit ganjil pertama yang menjadi gen pertama kromosom secara random, kemudian secara heuristik komputer menentukan langkah terbaik musuh dengan menggunakan algoritma minimax, ini menjadi digit kedua dari kromosom, lalu komputer merandom digit ganjil kedua yang menjadi gen ke 3 bagi kromosom, digit keempat yang merupakan langkah musuh ditentukan berdasarkan langkah terbaik kondisi itu dst. Gambar 4 memperlihatkan bagaimana pembentukan kromosom 5744.



(a) (b) (c) (d) (e)

Gambar 4. Peletakan disk pada kromosom 5744. Gambar (a) keadaan awal sebelum peletakan *disk* pertama komputer. Gambar (b), *disk* kuning (komputer) diletakan pada kolom 5 diikuti *disk* lawan (merah) pada kolom 7, pada Gambar (c) dan *disk* kedua komputer pada kolom ke 4, Gambar (d) dan gambar (e) pemain meletakannya pada kolom 4.

Terlihat bahwa digit genap yang dilakukan langkah pemain lawan dihasilkan komputer tidak secara random melainkan melakukan perhitungan heuristik terbaik saat itu dengan algoritma minimax level=1. Namun jelas digit ganjil dihasilkan komputer secara random setelah digit genap sebelumnya terbentuk.

2.2. Populasi dan Proses pemilihan

Populasi terdiri dari 150 buah kromosom yang panjang kromosomnya terdiri dari *k* digit yang menyatakan banyaknya level heuristik yang akan digunakan. Setiap digit ganjil adalah bilangan-angka acak dari 1 sampai dengan 7. Sedangkan digit genapnya diisi dengan menggunakan metode heuristik menggunakan algoritma minimax satu level. Proses pemilihan menentukan kromosom mana yang akan tetap bertahan pada generasi selanjutnya. Proses ini dilakukan dengan *Better Half*. Metode ini menggunakan setengah kromosom terbaik dari generasi sebelumnya berdasarkan nilai *fitness* akan bertahan dan setengah lainnya adalah hasil *crossover* pada kromosom-kromosom terpilih, sehingga dihasilkan sebuah generasi baru sejumlah generasi yang sebelumnya. Metode ini akan menghasilkan

generasi yang cenderung stabil, tidak banyak memiliki *outlier* pada kromosom-kromosom yang dihasilkan [6].

2.3. Proses *Crossover*

Dua kromosom dipilih secara acak dari kromosom terpilih, kemudian dipilih sebuah nilai *border-point* $c < k$, $n = 2n+1$. Nilai c menandakan pada digit ganjil keberapa, pemotongan kromosom terjadi. Misalkan, ada sepasang kromosom X dan Y dengan nilai $X = X_1X_2X_3...X_k$ dan $Y = Y_1Y_2Y_3...Y_k$ dengan $c = 3$ maka kromosom anak adalah $X = X_1X_2Y_3...Y_k$ dan $Y = Y_1Y_2X_3... X_k$. Perlu dipahami, bahwa nilai-nilai dari *bit* genap dari kromosom anak belum tentu sama dengan bit yang bersesuaian dengan gen induknya hasil *crossover*. Karena nilai bit genap didapat dari proses algoritma minimax satu level pada kondisi itu. Namun nilai bit ganjil kromosom anak sama dengan nilai gen kromosom induk yang bersesuai. Maka, sesungguhnya proses mating/crossover ini bertujuan untuk mengkawinkan dua sampel langkah pemain komputer.

2.4. Proses Mutasi

Proses mutasi dilakukan dengan algoritma yang sederhana, untuk suatu kromosom. Diambil nilai acak $m < k$, $m = 2n+1$, kemudian ganti nilai X_m dengan sebuah nilai acak antara 1 sampai dengan 7. Setelah itu ubah digit genap selanjutnya sebagaimana pembentukan kromosom namun digit ganjil setelahnya nilainya tetap. Meski terlihat sederhana, proses mutasi sangat penting untuk mencegah generasi menjadi "stagnan" karena kromosom-kromosom pada generasi tersebut suboptimal, sehingga perkawinannya pun menghasilkan generasi suboptimal [7].

2.5. Fitness

Algoritma minimax memerlukan sebuah fungsi yang dapat digunakan untuk "menilai" setiap daun-daun pada pohon keputusan. Fungsi yang digunakan untuk menilai keputusan tersebut, untuk sebuah matriks board *Ai* diperlihatkan pada persamaan (1)

$$f(A_i) = \frac{x_1 + 200x_2 + 100x_3 + 1000x_4}{1913} \quad \ldots(1)$$

Nilai 1913 diatas didapatkan dari nilai maksimum fungsi $f(A_i)$ untuk suatu kondisi board *Ai*. Tujuannya adalah menginterpolasi nilai-nilai $f(A_i)$ sehingga nilainya berada diantara $[0,1]$. Variabel x_1 hingga x_4 masing-masing menyatakan banyaknya garis kemenangan maksimum 13, banyaknya disc kawan maksimum yang ada pada garis kemenangan, banyaknya disk lawan maksimum yang ada pada garis kemenangan lawan dan langkah kemenangan atau kekalahan, untuk keadaaan board *Ai*. Board *Ai* adalah kondisi board setelah chromosom ke c_i diterapkan pada board A_{i-1}. Untuk kromosom c_i, nilai fitness dihitung dengan sebuah fungsi $F:\mathbb{R}^7 \rightarrow \mathbb{R}$ yang dinyatakan pada (2)

$$fit(c_i) = \begin{cases} 1 & Jika\ c_i\ adalah\ langkah\ kemenangan \\ 0 & Jika\ c_i\ adalah\ langkah\ kekalahan \\ \sum_{k=1}^{7} \left(\frac{1}{7}\right)^i f(A_i|x_i) & lainnya \end{cases} \quad \ldots(2)$$

Dimana:

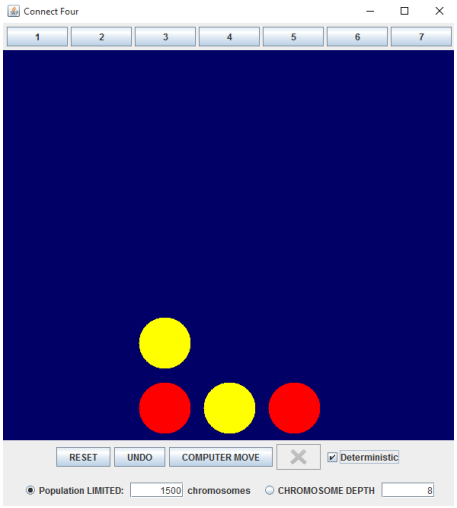
1. $X \in \mathbb{R}^7$ dimana X_i melambangkan nilai bit ke i pada kromosom X.
2. $f(A|X_i)$ melambangkan nilai heuristic board A jika langkah pada kromosom X_i dijalankan.
3. Jika pada suatu kondisi, langkah X_i menyebabkan player kalah, maka nilai kromosom langsung bernilai 0, dan jika menyebabkan kemenangan, nilai kromosom diberi angka 1.

2.6. Pemilihan kromosom terbaik

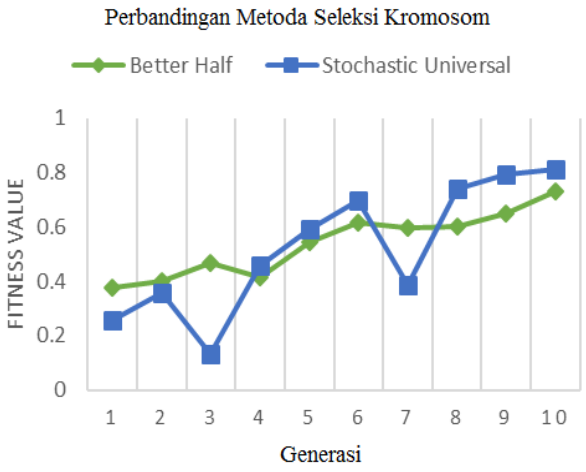
Setelah semua proses genetis dilakukan pada suatu populasi, dan nilai *fitness* setiap kromosom dihitung, maka kromosom paling baik adalah kromosom dengan nilai fitness tertinggi, pada generasi terakhir. Menentukan kapan generasi terakhir, aplikasi menyediakan 2 metode. Pertama adalah *time based* (digunakan 1500 msec) untuk setiap proses, sehingga pengambilan keputusan AI tidak dilakukan terlalu lama. Metode kedua adalah metode generasi, yaitu ketika sejumlah generasi telah selesai diproses. Setelah proses genetis berhenti, kromosom dengan fitness tertinggi dijalankan dengan hanya mengambil langkah sesuai dengan digit pertama dari kromosom tersebut.

3. Implementasi dan Pembahasan

Aplikasi menggunakan bahasa pemrograman java sebagai GUI application, dengan basis GUI Swing, Gambar 5. Pemain dapat berinteraksi dengan system dengan meng *click* angka 1-7 yang ada pada atas panel utama, yang berarti user ingin meletakkan kepingan pada kolom tersebut. Apabila pemain ingin mengulang langkah terakhir, tekan tombol UNDO. Jika pemain bermain sebagai pemain ke-dua maka tekan tombol COMPUTER MOVE. Jika pilihan deterministik di centang artinya langkah komputer adalah langkah digit pertama kromosom terbaik dengan kolom terendah sedangkan jika tidak maka langkah komputer adalah langkah digit pertama dari perandoman kromosom terbaik. Kecerdasan aplikasi ditentukan dari nilai isian dari input pupulation LIMITED dan CHROMOSOM DEPTH. Semakin besar nilai kedua input maka semakin cerdas aplikasi meaminkan permainan ini.



Gambar 5. User interface aplikasi permainan



Gambar 6. Perbandingan metoda seleksi kromosom

Kecerdasan sistem dibandingkan dengan kecerdasan dari algoritma minimax yang dibuat dengan fungsi heuristic yang sama [5]. Hasil perbandingan diperlihatkan 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 (algoritma minimax). Kotak berwarna 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 perbandingan kecerdasan sistem dengan [5]

Depth	4	5	6	7	8
4	L (35)	W (33)	W (37)	W (35)	D (42)
5	W (33)	L (35)	L (37)	W (39)	L (19)
6	L (40)	W (31)	W (31)	W (27)	W (29)
7	L (34)	L (34)	L (31)	D (42)	W (35)
8	L (33)	L (35)	L (31)	L (33)	W (31)

Berdasrkan hasil diatas, Secara umum kedua algoritma memiliki kemampuan yang hampir sama, dimana mereka saling mengalahkan ketika memiliki depth yang lebih dalam dari algoritma lawannya. Disini, terlihat keunikan dari algoritma genetik, karena algoritma genetic menggunakan konsep -konsep stokastik (banyak kemungkinan, nilai-nilai acak) terdapat beberapa *outlier* pada hasil uji coba diatas, terlihat bahwa algoritma genetic dengan depth (8) kalah dengan algoritma minimax dengan depth (5). Ini bisa terjadi jika suatu generasi nilainya tidak membaik setelah melalui mutasi, seleksi, dan crossover. Begitu juga algoritma genetic bisa menang walaupun memiliki depth lebih kecil (4 vs 5 dan 5 vs 6),

sangat mungkin terjadi jika kita mendapat generasi yang sangat bagus, jika dibandingkan dengan metode minimax yang cenderung tidak memiliki proses stokastik.

Percobaan kedua dilakukan dengan mengubah proses seleksi kromosom yang bertahan hidup pada generas selanjutnya yaitu antara Better Half dengan Stochastik Universal. Terlihat bahwa seleksi better half menghasilkan individu yang lebih stabil, rendah outlier, namun generasinya cenderung stagnan (tidak banyak berubah). Sedangkan pada proses stochastic universal sampling, nilainya lebih bervariasi, ini karena sebuah kromosom dengan nilai *fitness* yang kecil dapat tetap ikut dalam generasi berikutnya, sehingga meningkatkan variabilitas dari kromosom. Variabilitas ini juga yang memungkinkan kromosom berkembang lebih pesat, dan tidak stagnan pada suatu titik.

4. Simpulan

Ada dua kesimpulan yang dapat diambil dari sistem yaitu

- Sistem telah berhasil dirancang dan diimplementasikan dengan baik dengan kecerdasan yang baik.
- Kecerdasan AI yang dibuat dengan algoritma genetic dan minimax dibandingkan dengan kecerdasan AI yang hanya menggunakan algoritma minimax dengan formula heuristik yang sama relative sama. Namun algoritma genetic memiliki variabilitas yang lebih tinggi, sehingga lebih menarik untuk dimainkan.

Daftar Pustaka

[1] Stefan Edelkamp and Peter Kissmann, "*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, 2011

[2] Victor Allis, *"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, 1988.

[3] Mohamed F. Abdelsadek, *"Using Genetic Programming to Evolve a Connect-4 game player"*, Department of Computer Science, Columbia University.

[4] Kirk Baly, Andrew Freeman, Andrew Jarratt, Kyle Kling, Owen Prough, Greg Hume, *"Teaching Computers to Think: Automated Analysis of Connect Four"*, Research in Computer Science, 2012.

[5] Halim Andrew Mahisa, Frederikus Judianto, Samuel Lukas, Petrus Widjaja, *"Pemodelan dan Pengimplementasian Permainan Connect Four*" Seminar Nasional Inovasi Dan Aplikasi Teknologi Di Industri, ITN Malang, 2017

[6] R. Sivaraj, T. Ravichandran, *A Review of Selection Methods in Genetic Algorithms*. Journal of Engineering Science and Technology, 2011.

[7]. S.Forrest, Genetic Algorithms: Principles of Natural Selection Applied to Computation. In Science, New Series, Volume 261, Issue 5123, 1993